RZ/T2H, RZ/N2H Group

OpenPLC Runtime Installation and Connectivity

Introduction

This document provides a guide to install OpenPLC and OpenPLC Runtime on the Windows PC and the evaluation board.

This guide provides the following information:

- Building procedure
- Preparation for the test environment for OpenPLC

Please be advised that any GitHub repositories, libraries, or code referenced within this document may be governed by specific licenses and terms as defined by their respective authors or copyright holders. Users are solely responsible for reviewing, understanding, and adhering to all applicable licenses and terms of use associated with such content. Our company assumes no responsibility or liability for compliance with these third-party licenses.

Please also refer to the documents below to build VLP and boot the evaluation board.

- r01us0820ej0xxx-rz(Release_Note).pdf
- r01us0821ej0xxx-rz(Linux_Start-up_Guide_RZ/T2H_N2H).pdf
- r01us0848ej0xxx-rz-t2h-n2h(OpenPLC_Release_Note).pdf

"xxx" is the revision number.

Target Reference Board

- RZ/T2H Evaluation Board
- RZ/N2H Evaluation Board

Target Software

- RZ/T2H, RZ/N2H VLP version 5.0.0 or later.
- OpenPLC

Contents

1. Environmental Requirement 3
2. Build and boot Instructions 4
3. Install the OpenPLC Runtime to the evaluation board 6
4. Appendix 10
4.1 Modbus TCP communication with Windows PC 10
4.2 OpenPLC Editor Setup and Communication Procedure 24
5. Revision History 38
Website and Support 39

1. Environmental Requirement

The environment for preparing OpenPLC environment is listed in **Table 1**. Refer to the below documents for details about setting up the environment:

Figure 1 shows the recommended environment for this package. The picture of RZ/T2H is shown as an example.

A Windows PC can be used as a serial terminal interface with software such as TeraTerm.

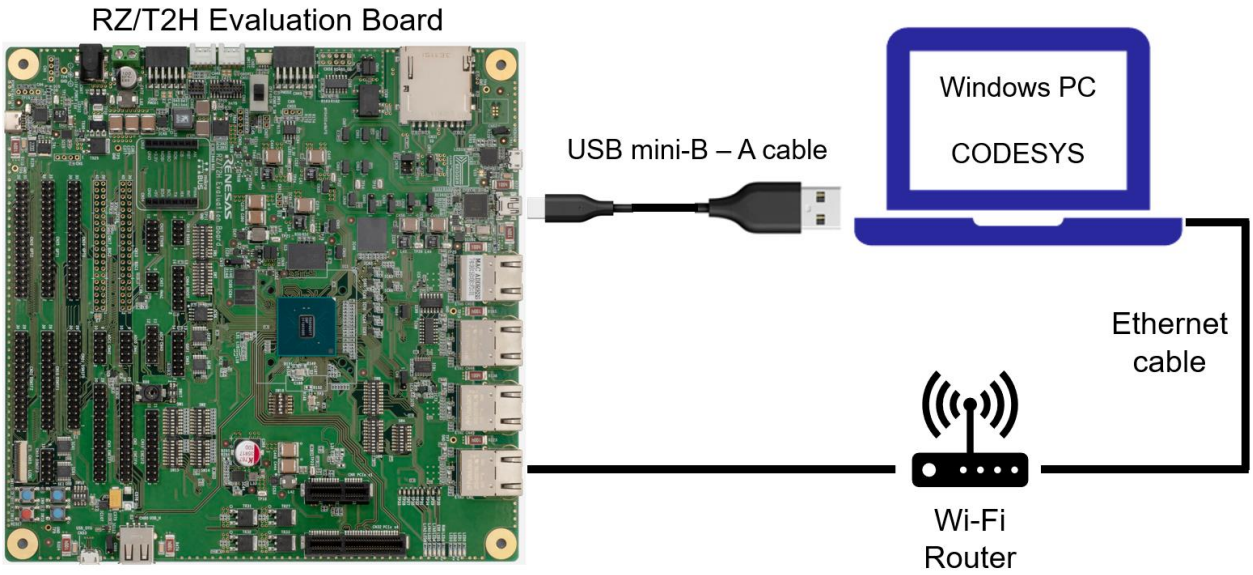


Figure 1. Recommend environment.

Table 1. Equipment and Software for Developing Environments of Linux Platform

Equipment		Description
Linux Host PC		Used as build/debug environment 100GB free space on HDD or SSD is necessary
	OS	Ubuntu 22.04 LTS 64 bit OS must be used. 22.04 inside a docker container also OK.
Windows Host PC		Used as debug environment, controlling with terminal software
	OS	Windows 11
	Terminal software	Used for controling serial console of the target board Tera Term (latest version) is recommended Available at https://ttssh2.osdn.jp/index.html.en
	VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: http://www.ftdichip.com/Drivers/VCP.htm
USB serial to Mini–USB Cable		Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is Mini USB type B.
micro–SD Card		Use to boot the system, and store applications.

2. Build and boot Instructions

This section describes the instructions to build the Board Support Package.

Before starting the build, run the command below on the Linux Host PC to install packages used for building the VLP.

```
$ sudo apt-get update
$ sudo apt install build-essential chrpath cpio debianutils diffstat file gawk \
gcc git iputils-ping libacl1 liblz4-tool locales python3 python3-git \
python3-jinja2 python3-pexpect python3-pip python3-subunit socat texinfo unzip \
wget xz-utils zstd bmap-tools
```

Please refer to the URL below for detailed information:

- https://docs.yoctoproject.org/5.0.11/brief-yoctoprojectqs/index.html

Run the commands below and set the user name and email address before starting the build procedure. Without this setting, an error occurs when building procedure runs git command to apply patches.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

Copy all files obtained from Renesas into your Linux Host PC prior to the steps below. The directory which you put the files in is described as <package download directory> in the build instructions.

(1) Build VLP with the Linux Start-up Guide

Read the below document and build VLP normally. After that, proceed to the next step. “xxx” means the document revision.

- r01us0821ej0xxx-rz(Linux_Start-up_Guide_RZ/T2H_N2H).pdf

(2) Build Initialize

Initialize a build using the 'oe-init-build-env' script in Poky and point TEMPLATECONF to platform conf path.

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rz-distro/conf/templates/vlp-v5-conf/ source \
poky/oe-init-build-env build
```

(3) Edit local.conf

Enable the below packages to install OpenPLC Runtime. Add the below lines in “~/rz_vlp_v${PACKAGE_VERSION}/build/conf/local.conf”.

```
# Build openplc
EXTRA_IMAGE_FEATURES ?= "tools-sdk dev-pkgs "
HOSTTOOLS += " fallocate pkg-config"
IMAGE_INSTALL:append = " pkgconfig util-linux libtool python3 python3-pymodbus \
python3-flask python3-flask-login python3-pyserial python3-sqlite3 sqlite3 \
libmodbus cmake make gcc-dev autoconf-dev binutils-dev libbfd cpp cpp-symlinks gcc \
gcc-symlinks binutils binutils-symlinks autoconf g++ g++-symlinks automake git \
bison bison-dev flex "
INCOMPATIBLE_LICENSE_EXCEPTIONS:append = " libtool:GPL-3.0-only gcc-dev:GPL-3.0-only \
autoconf-dev:GPL-3.0-only libbfd:GPL-3.0-only cpp:GPL-3.0-only \
cpp-symlinks:GPL-3.0-only gcc:GPL-3.0-only gcc-symlinks:GPL-3.0-only \
autoconf:GPL-3.0-only g++:GPL-3.0-only g++-symlinks:GPL-3.0-only \
automake:GPL-3.0-only "
PREFERRED_VERSION_bison = "3.8.2"
# Set rootfs over 2.1G
IMAGE_ROOTFS_SIZE = "2202009"
```

(4) Start a build

Run the commands below to start a build. Building an image can take up to a few hours depending on the user's host system performance.

Build the target file system image using bitbake.

```
$ MACHINE=<board> bitbake core-image-minimal
```

<board> can be selected by referring to the table below.

Table 2. List of platforms and boards

Renesas MPU	<board>
RZ/T2H	rzt2h-dev
RZ/N2H	rzn2h-dev

After the building is successfully completed, a similar output will be seen, and the command prompt will return. The following log is an example.

```
NOTE: Tasks Summary: Attempted 3512 tasks of which 8 didn't need to be rerun and all
succeeded.
```

All necessary files listed in the following table will be generated by the bitbake command and will be in the "**~/rz_vlp_v${PACKAGE_VERSION}/build/tmp/deploy/images**" directory.

Table 3. Image files

RZ/T2H	**Linux kernel**	Image-rzt2h-dev.bin
	Device tree file	r9a09g077m44-dev.dtb
	root filesystem	core-image-minimal-rzt2h-dev.rootfs.tar.bz2
	Boot loader	bl2_bp_xspi0-rzt2h-dev.srec fip-rzt2h-dev.srec
	SD image	core-image-minimal-rzt2h-dev.rootfs.wic.gz core-image-minimal-rzt2h-dev.rootfs.wic.bmap
RZ/N2H	**Linux kernel**	Image-rzn2h-dev.bin
	Device tree file	r9a09g087m44-dev.dtb
	root filesystem	core-image-minimal-rzn2h-dev.rootfs.tar.bz2
	Boot loader	bl2_bp_xspi0-rzn2h-dev.srec fip-rzn2h-dev.srec
	SD image	core-image-minimal-rzn2h-dev.rootfs.wic.gz core-image-minimal-rzn2h-dev.rootfs.wic.bmap

If you want to know the components installed to the root filesystem, please check the manifest file. The manifest file is created to the following path after building the images:

~/rz_vlp_v${PACKAGE_VERSION}/build/tmp/deploy/images/rzt2h-dev/core-image-minimal-rzt2h-dev.manifest

~/rz_vlp_v${PACKAGE_VERSION}/build/tmp/deploy/images/rzn2h-dev/core-image-minimal-rzn2h-dev.manifest

(5) Boot the evaluation board with the Linux Start-up Guide

Read the below document and boot the evaluation board. Please ignore the building instructions in this guide because the building was completed in this application note.

- r01us0821ejxxxx-rz(Linux_Start-up_Guide_RZT2H_N2H).pdf

The steps to boot the evaluation board are completed, so proceed with the next chapter.

3. Install the OpenPLC Runtime to the evaluation board

This section describes how to install the OpenPLC Runtime to the evaluation board from the OpenPLC application on the Windows PC.

(1) Install OpenPLC Runtime to the evaluation board

Run the commands below to install OpenPLC Runtime.

```
root@rzt2h-dev:~# git clone https://github.com/thiagoralves/OpenPLC_v3.git
root@rzt2h-dev:~# cd OpenPLC_v3/
root@rzt2h-dev:~/OpenPLC_v3# git checkout -b tmp a283d0dbac813116babdf81c53144c4d45\
5edf49
root@rzt2h-dev:~/OpenPLC_v3# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
root@rzt2h-dev:~/OpenPLC_v3# export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
root@rzt2h-dev:~/OpenPLC_v3# ldconfig
root@rzt2h-dev:~/OpenPLC_v3# ./install.sh custom
```

After that the below log can be seen.

```
Installing
OpenPLC on Custom Platform
[MATIEC COMPILER]
…
checking for gcc ... gcc
checking whether the C compiler works... Yes
…
[ 90%] Built target asiopal
[100%] Built target asiodnp3 Install the project...
--
Install configuration: "Release"
…
Moving Files...
Compiling for Linux
Generating object files... Generating glueVars
Compiling main program...
Compilation finished successfully!
```

Notice: Sometimes this installation could fail in the “libmodbus” stage, in this case, you can also run step (2) to run the OpenPLC Runtime.

(2) Run the webserver

Moidfy the file "~/OpenPLC_v3/webserver/monitoring.py" on the evaluation board as below:

```
@@ -1,6 +1,6 @@
 import time, threading
 from struct import *
-from pymodbus.client.sync import ModbusTcpClient
+from pymodbus.client import ModbusTcpClient

 class debug_var():
     name = ''
```

Run the commands below to start the webserver.

```
root@rzt2h-dev:~/OpenPLC_v3# cd webserver/
root@rzt2h-dev:~/OpenPLC_v3/webserver# python3 webserver.py
```

Please be sure to change the directory into "webserver".

The below error occurs when you execute commands not under directory "webserver".

```
* Serving Flask app "webserver" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

(3) Check the OpenPLC Runtime on the browser application in Windows PC

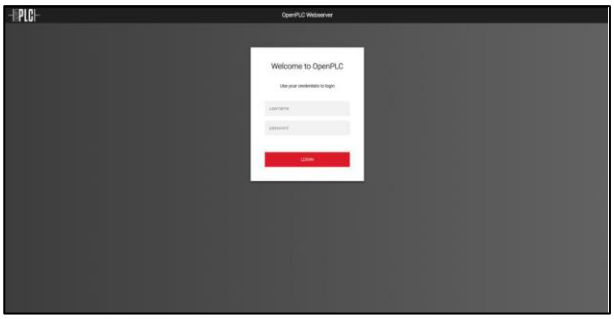
The main runtime webserver is available at your target IP address on port 8080. For example, if you installed

OpenPLC Runtime on a RZT2H/RZN2H, and your RZT2H/RZN2H IP address is 10.166.29.238, then you can access OpenPLC.

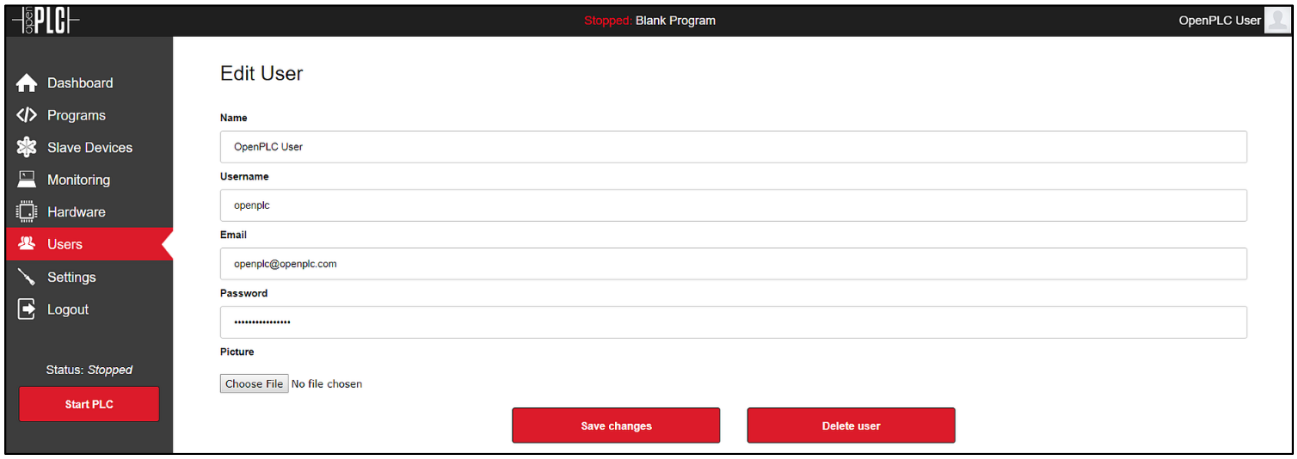
Runtime by opening your web browser and pointing it to http://10.166.29.238:8080.



Once you access OpenPLC webserver, you should see in your browser window a login page like this:



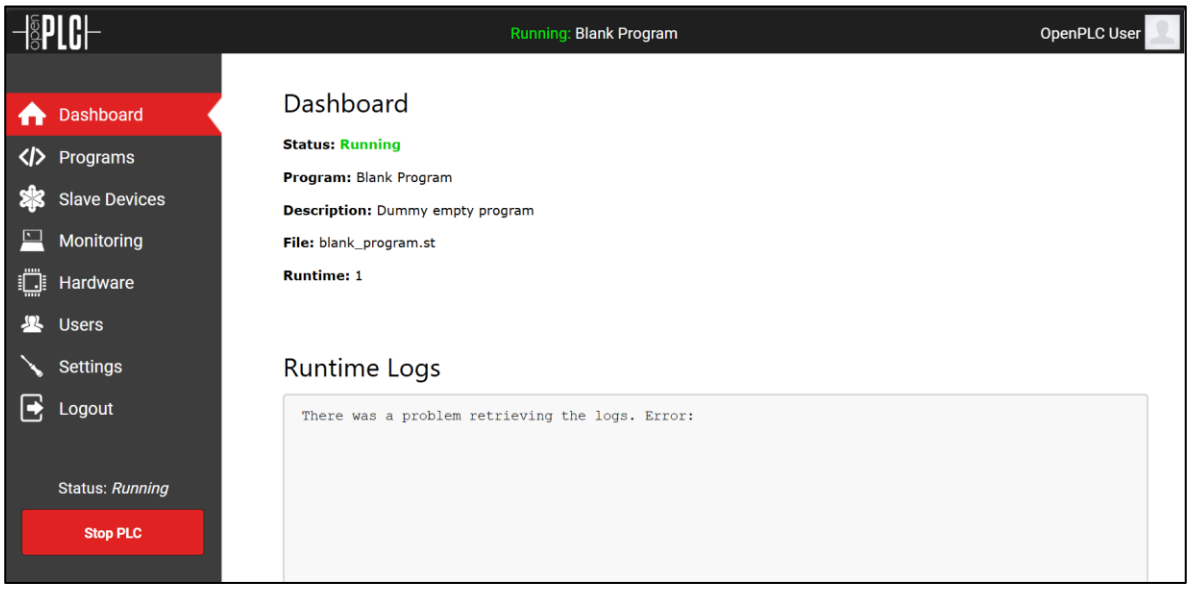
The default username and password is openplc (Username) and openplc (Password).




```
Log on board when Login OK:
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /dashboard HTTP/1.1" 302 -
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /login HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:26] "GET /static/openplc_logo.gif HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "POST /login HTTP/1.1" 302 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /dashboard HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/openplc_logo.gif HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/arrow.png HTTP/1.1" 304 -
```

```
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/programs-icon-64x64.png HTTP/1.1"
 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/users-icon-64x64.png HTTP/1.1" 30
4 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/logout-icon-64x64.png HTTP/1.1" 3
04 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/modbus-icon-512x512.png HTTP/1.1"
 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/settings-icon-64x64.png HTTP/1.1"
 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/monitoring-icon-64x64.png HTTP/1.
1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/hardware-icon-980x974.png HTTP/1.
1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/default-user.png HTTP/1.1" 304 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /static/home-icon-64x64.png HTTP/1.1" 304
 -
10.231.33.40 - - [09/Jun/2026 10:33:31] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:32] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:33] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:34] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:35] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:36] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:37] "GET /runtime_logs HTTP/1.1" 200 -
10.231.33.40 - - [09/Jun/2026 10:33:38] "GET /runtime_logs HTTP/1.1" 200 -
```

Can **start PLC** with default **Blank Program** successfully




```
Log on board when Starting PLC:
OpenPLC Runtime starting...
Skipping configuration of Slave Devices (mbconfig.cfg file not found)
Interactive Server: Listening on port 43628
Interactive Server: waiting for new client...
Warning: Persistent Storage file not found
Setting main thread priority to RT
Locking main thread memory
Getting current time
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
```

```
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued runtime_logs() command
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
10.231.33.40 - - [09/Jun/2026 10:33:39] "GET /runtime_logs HTTP/1.1" 200 -
Openning database
Enabling Modbus on port 502
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued start_modbus() command to start on port: 502
Server: Listening on port 502
Server: waiting for new client...
```

4. Appendix

This chapter provides guidance on utilizing the OpenPLC Runtime on the evaluation board. The explanation in this chapter is based on RZ/T2H as an example.

OpenPLC Runtime is an open-source PLC (Programmable Logic Controller) runtime that runs on standard Linux-based hardware. OpenPLC supports IEC 61131-3 standard programming languages including Ladder Diagram and Structured Text, and provides industrial communication protocols such as Modbus TCP, DNP3, and EtherNet/IP. This makes it suitable for industrial automation prototyping and evaluation on embedded platforms.

Please follow the following description to know how to prepare the test environment and test it on the evaluation board.

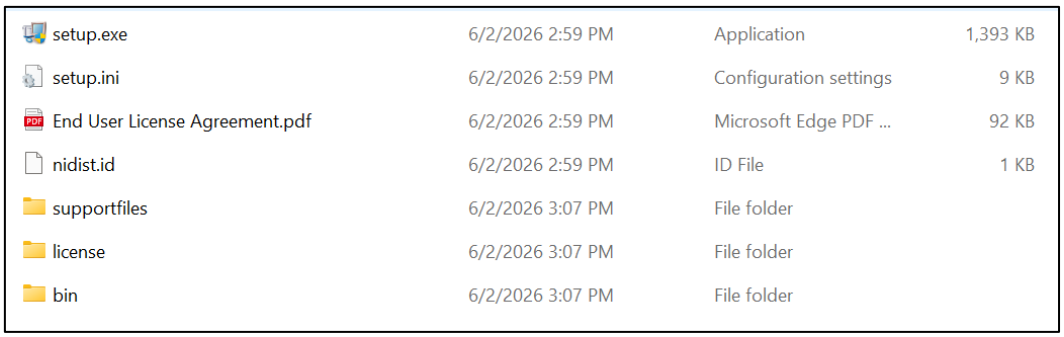
4.1 Modbus TCP communication with Windows PC

(1) Download Simply Modbus TCP Client Tool

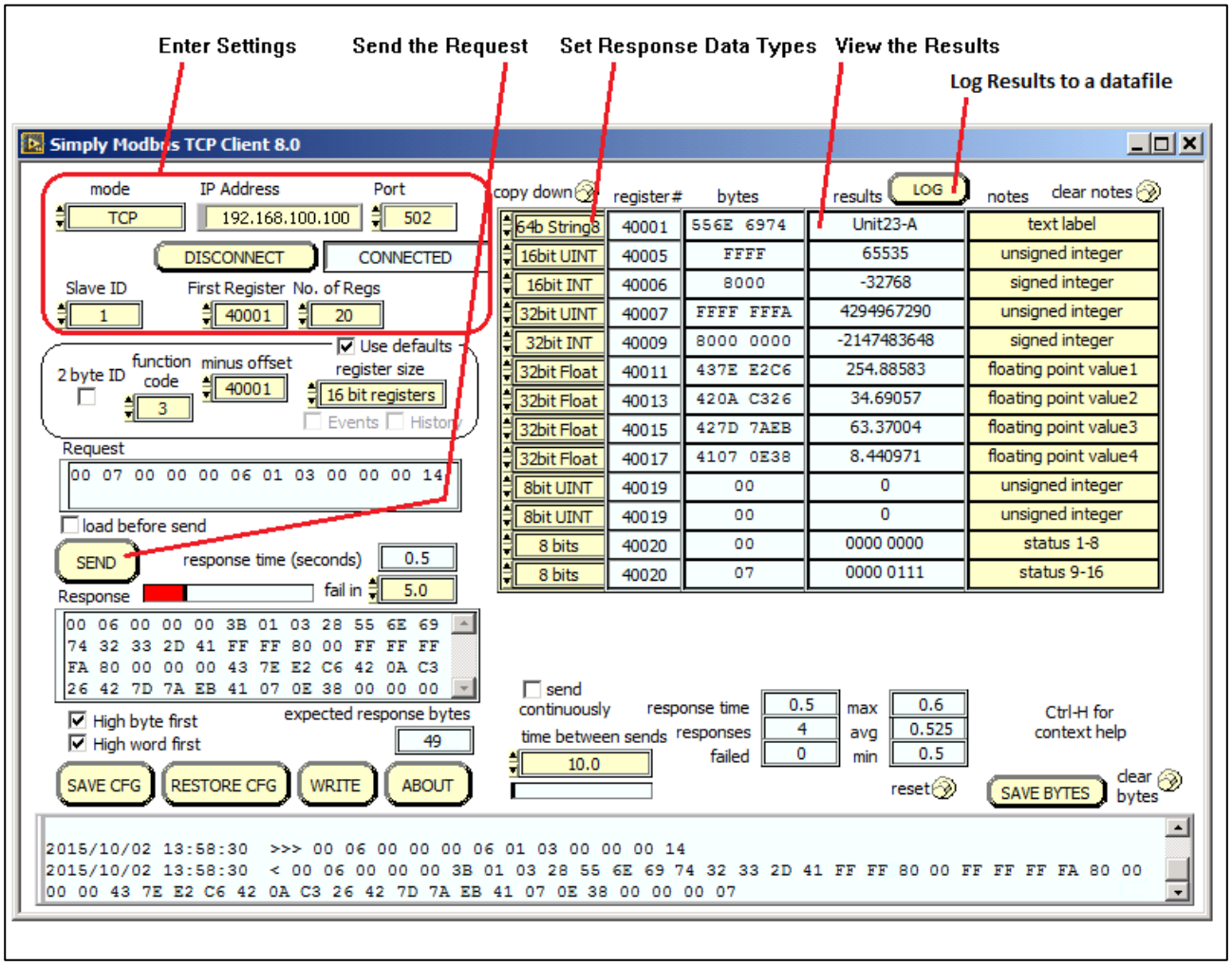
This test tool turns your Windows PC into a Modbus TCP Client or Modbus RTU-over-TCP Client. Connect to Modbus TCP or server (slave) devices to test, troubleshoot, and monitor Modbus TCP communication.

Notes: This is a demo version. It allows a limited number of requests per session. Restart the application to reset the counter if the demo popup appears.

- Access the download page: https://www.simplymodbus.ca/TCPclient.htm
- Scroll to Free Download and choose one Compiler:
  Ex: Compiler 14 - SimplyModbusTCPclient8.1.2Install.zip
- Download the installer and install the application:
  A) Extract the downloaded ZIP file, then run setup.exe and follow the installation wizard (Next → Next → Finish). Navigate to the extracted folder STM812 (note: the folder name may vary depending on the version downloaded).



B) Once the installation is complete, open "Simply Modbus TCP Client 8.1.2". The following window will appear:



(2) Set the IP addresses

Set the IP address of the evaluation board and the opposite device. The below settings are the examples. Note that the evaluation board and the opposite device are in the same network, or they are connected directly:

A) Evaluation Board:

```
$ ifconfig eth0 192.168.15.127 netmask 255.255.255.0 up
$ route add default gw 192.168.15.1
```

B) Windows PC : The requested operation requires elevation (Run as administrator).

```
$ netsh interface ip set address "Ethernet interface" static 192.168.5.20 \
255.255.255.0
```

(3) Check connection

Check connection between board and Windows PC by ping command:

A) Evaluation Board:

```
$ ping -c 10 192.168.5.20
```

B) Windows PC :

```
$ ping -c 10 192.168.5.2
```

(4) Start OpenPLC webserver on the board

Run the commands below to start the webserver.

```
root@rzt2h-dev:~/OpenPLC_v3# cd webserver/
root@rzt2h-dev:~/OpenPLC_v3/webserver# python3 webserver.py
```

(5) Access OpenPLC Web UI from PC browser

Open a browser on the Windows PC that guide at step "Check the OpenPLC Runtime on the browser application in Windows PC"

(6) Create the ST program file

Create and upload a Structured Text (ST) program that reads a Modbus Holding Register input, multiplies it by 2, and writes the result to another register.

- Create the ST program file on the board or somewhere in your Windows File Explorer
- Ex: vi /tmp/test_modbus.st
- Paste the following program save file:

```
PROGRAM test_modbus
  VAR
    input_val  AT %QW0 : INT;
    output_val AT %QW1 : INT;
  END_VAR

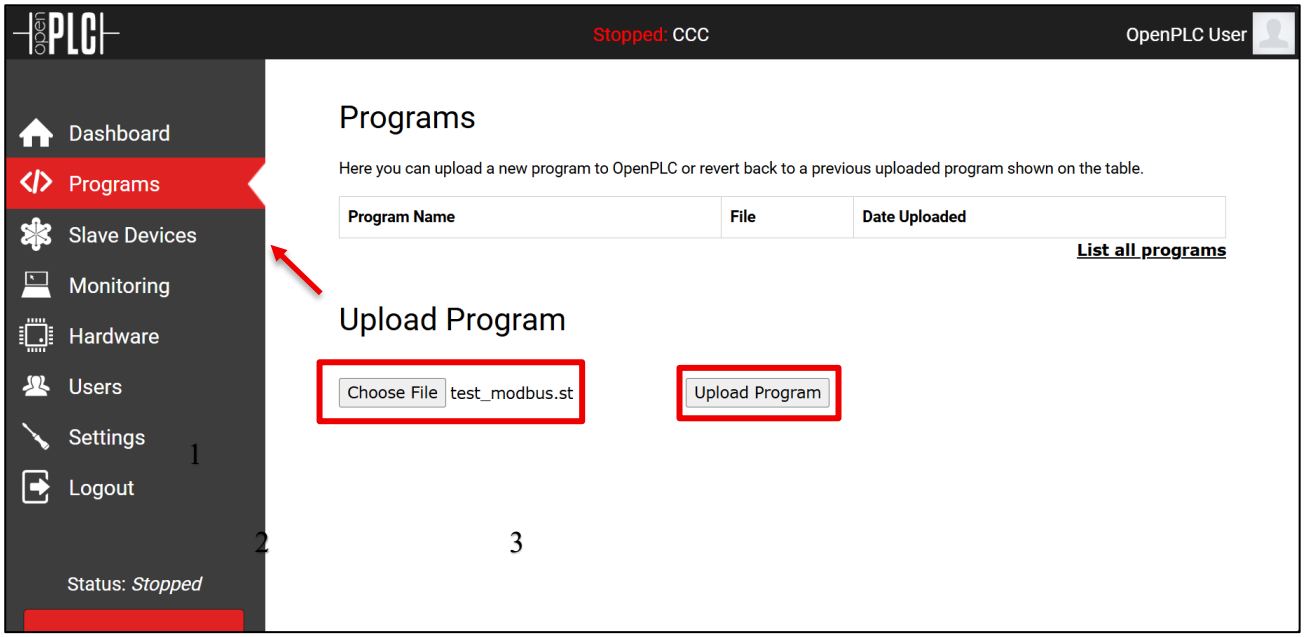
  output_val := input_val * 2;

END_PROGRAM

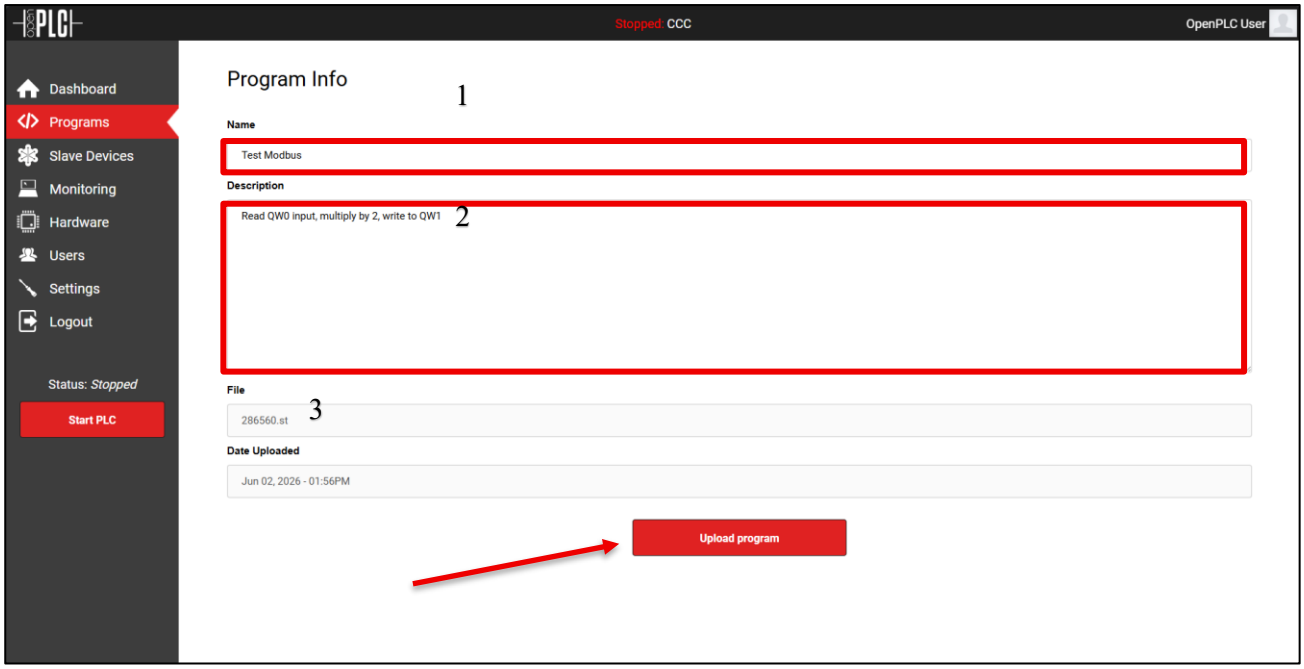
CONFIGURATION Config0
  RESOURCE Res0 ON PLC
    TASK task0(INTERVAL := T#100ms, PRIORITY := 0);
    PROGRAM instance0 WITH task0 : test_modbus;
  END_RESOURCE
END_CONFIGURATION
```

(7) Upload program via Web UI

- In the browser, navigate to Programs → click Choose File "test_modbus.st" has done in (6) → click Upload Program:

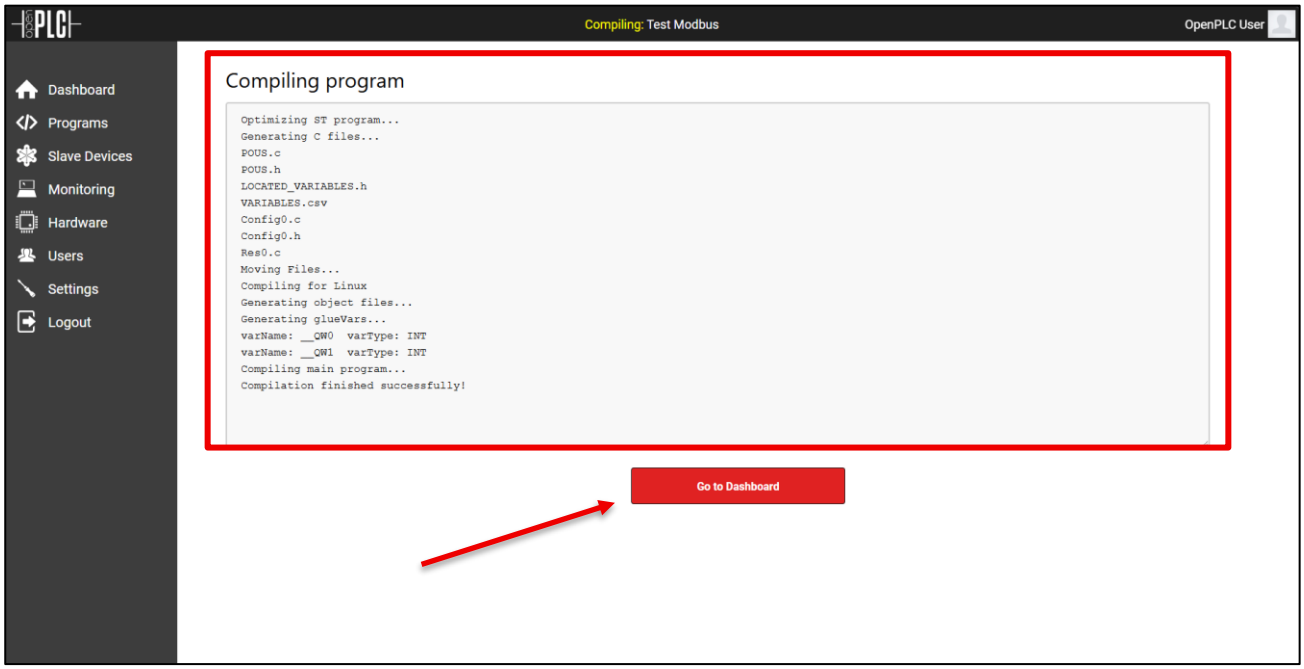


- Edit "Name" and "Description" → click "Upload program".
  Ex: the name is "Test Modbus", description is "Read QW0 input, multiply by 2, write to QW1"



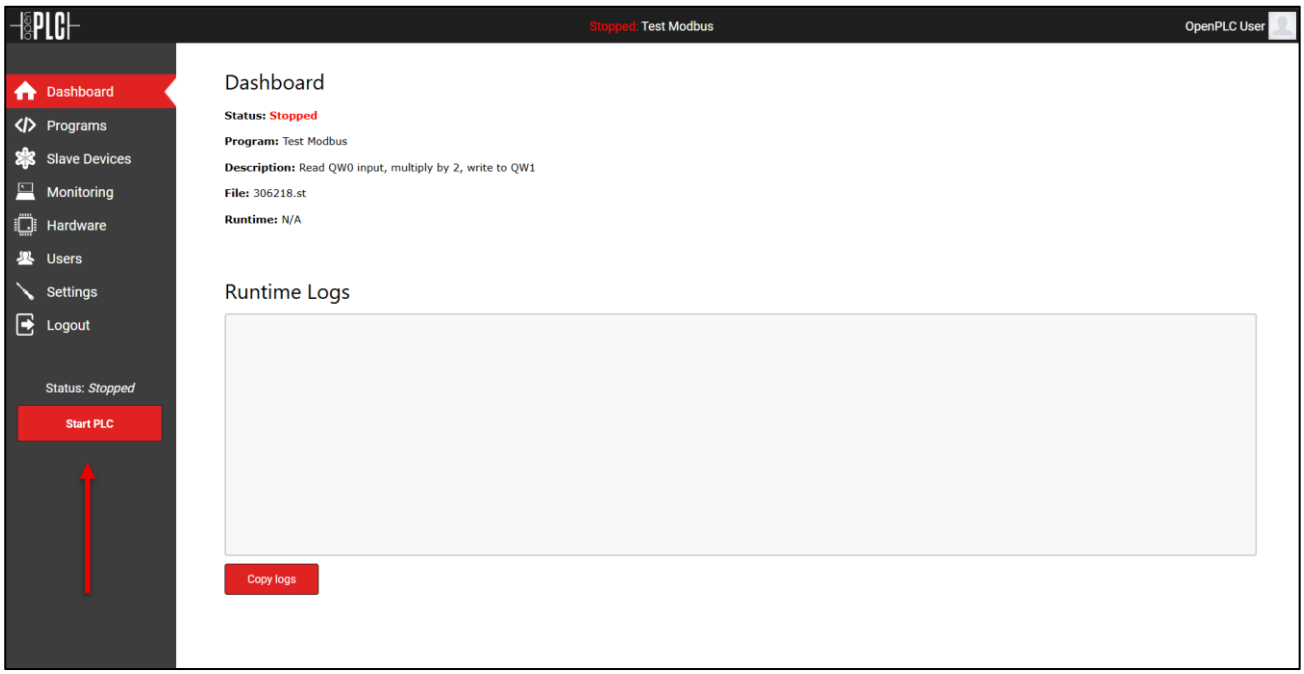
(8) Compile and launch program

After successful compilation, click **Launch program**. Once the "Go to Dashboard" button appears, click it to proceed to the dashboard.

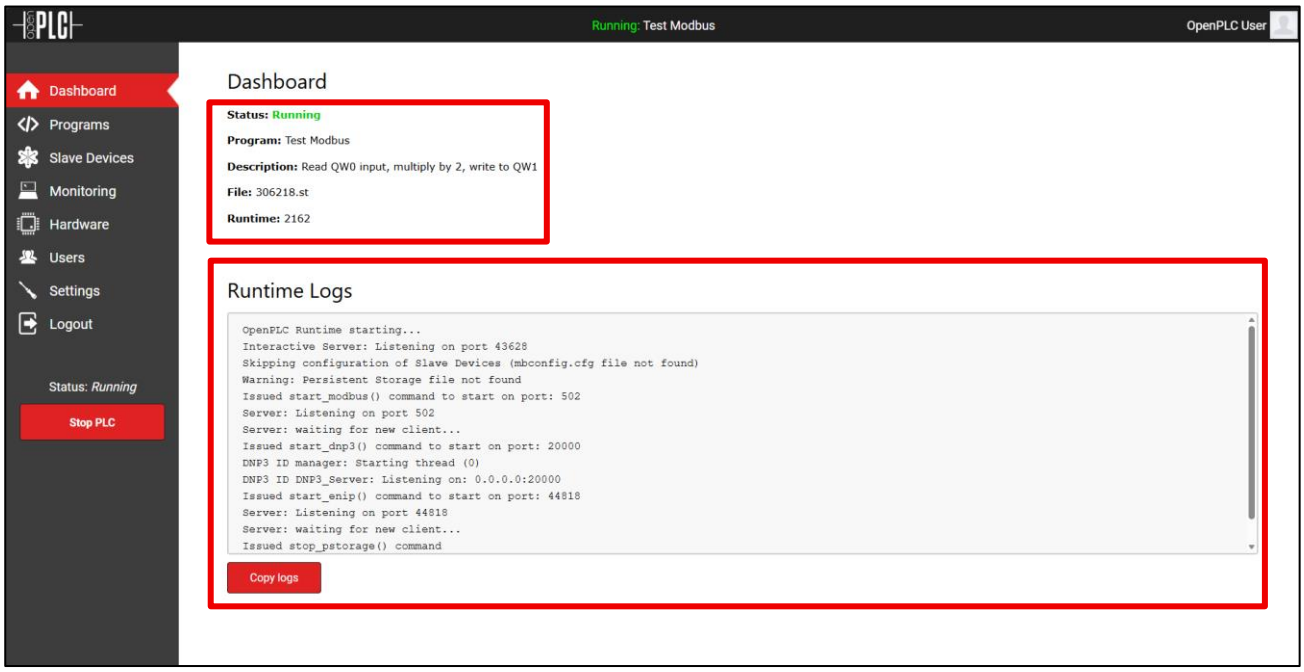


(9) Start PLC

- Click **Start PLC**.



- Dashboard should show: **Status: Running | Program: Test Modbus**



- Check log on RZT2H:

```
192.168.5.20 - - [02/Jun/2026 14:21:10] "GET /runtime_logs HTTP/1.1" 200 -
192.168.5.20 - - [02/Jun/2026 14:21:11] "GET /runtime_logs HTTP/1.1" 200 -
OpenPLC Runtime starting...
Skipping configuration of Slave Devices (mbconfig.cfg file not found)
Interactive Server: Listening on port 43628
Interactive Server: waiting for new client...
Warning: Persistent Storage file not found
Setting main thread priority to RT
Locking main thread memory
Getting current time
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued runtime_logs() command
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
192.168.5.20 - - [02/Jun/2026 14:21:12] "GET /runtime_logs HTTP/1.1" 200 -
Openning database
Enabling Modbus on port 502
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued start_modbus() command to start on port: 502
Server: Listening on port 502
Server: waiting for new client...
Interactive Server: client ID: 4 has closed the connection
Enabling DNP3 on port 20000
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 4...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 4
Issued start_dnp3() command to start on port: 20000
```

```
Interactive Server: client ID: 4 has closed the connection
Terminating interactive server connections
Enabling EtherNet/IP on port 44818
DNP3 ID manager: Starting thread (0)
channel state change: OPENING
DNP3 ID DNP3_Server: Listening on: 0.0.0.0:20000
DNP3 Enabled
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Issued start_enip() command to start on port: 44818
Server: Listening on port 44818
Server: waiting for new client...
Interactive Server: client ID: 9 has closed the connection
Disabling Persistent Storage
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Issued stop_pstorage() command
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
persistent.file removed!
Name: input_val
Location: AT
Type: :

Name: output_val
Location: %QW1
Type: INT

Name: display_in
Location: %QW2
Type: INT

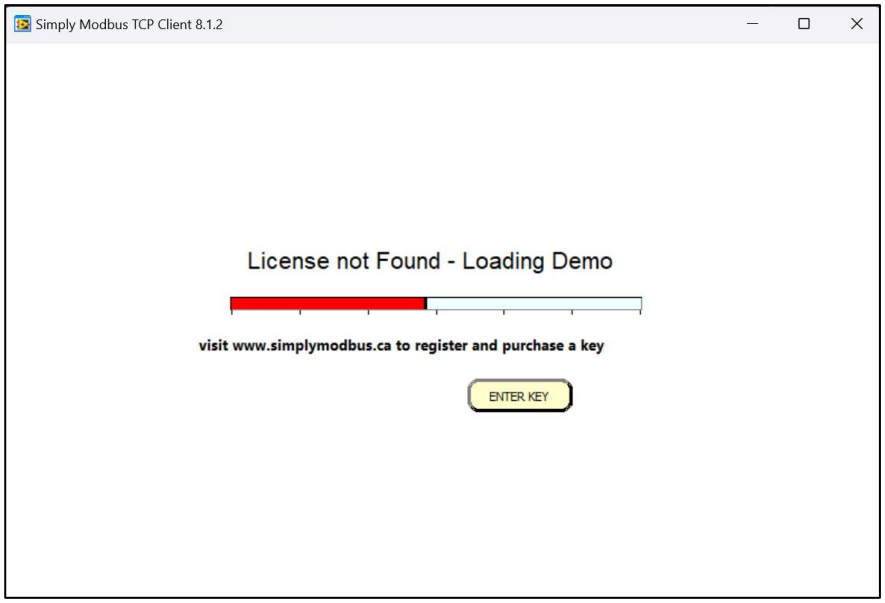
192.168.5.20 - - [02/Jun/2026 14:21:13] "GET /start_plc HTTP/1.1" 302 -
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: Client accepted! Creating thread for the new client ID: 11...
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 11
Interactive Server: Client accepted! Creating thread for the new client ID: 9...
Issued runtime_logs() command
Interactive Server: waiting for new client...
Interactive Server: Thread created for client ID: 9
Interactive Server: client ID: 11 has closed the connection
Terminating interactive server connections
Interactive Server: client ID: 9 has closed the connection
Terminating interactive server connections
…
```

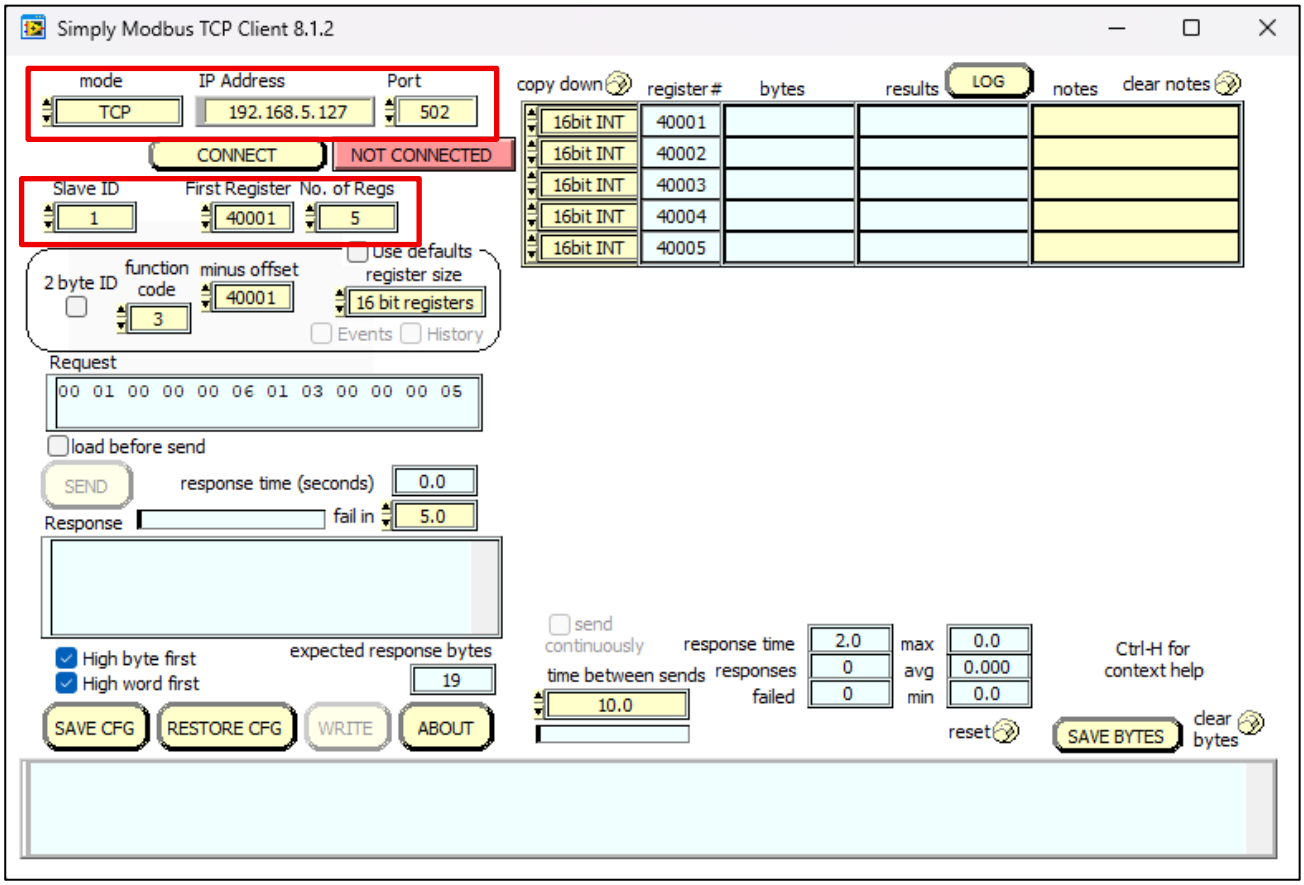
- Verfiy Modbus port is listening on the board

```
root@rzt2h-dev:# netstat -tln | grep 502
tcp        0      0 0.0.0.0:502             0.0.0.0:*               LISTEN
root@rzt2h-dev:#  cat OpenPLC_v3/webserver/core/LOCATED_VARIABLES.h
__LOCATED_VAR(INT,__QW0,Q,W,0)
__LOCATED_VAR(INT,__QW1,Q,W,1)
```

(10) Connect Simply Modbus to Board

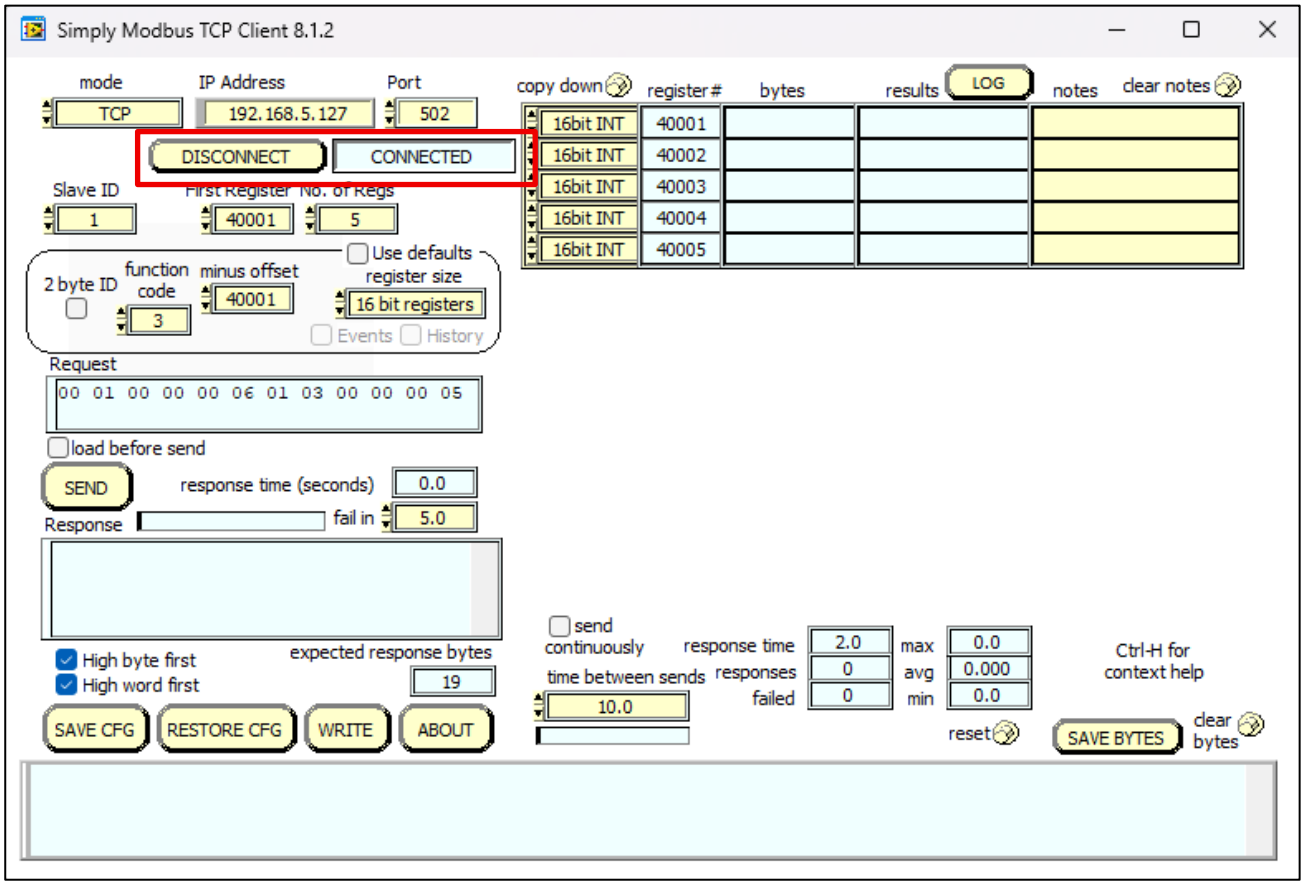
- Open Simply Modbus TCP Client (Launch the application from the Start menu or desktop shortcut.)





Field	Value	Description
mode	TCP	Modbus TCP protocol
IP Address	192.168.5.127	Board ETH3 (interface eth0) address
Port	502	Modbus TCP standard port
Slave ID	1	OpenPLC default slave ID
function code	3	Read Holding Registers
First Register	40001	Start from register 1
No. of Regs	5	Read 5 registers

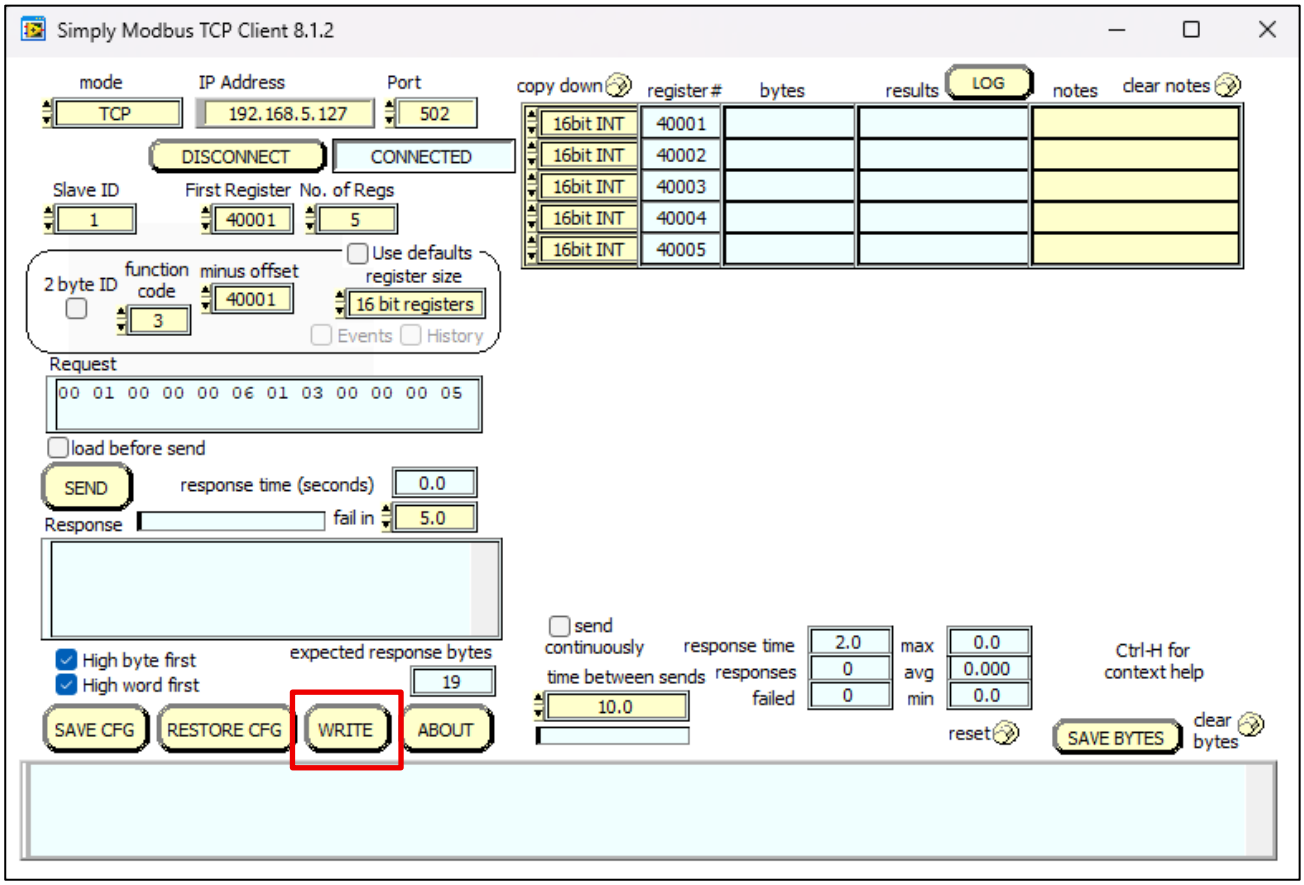
- Click the CONNECT button. The status indicator should change from NOT CONNECTED to CONNECTED. If connection fails, verify PLC is Running on dashboard and port 502 is listening on the board.



(11) Write Register Value

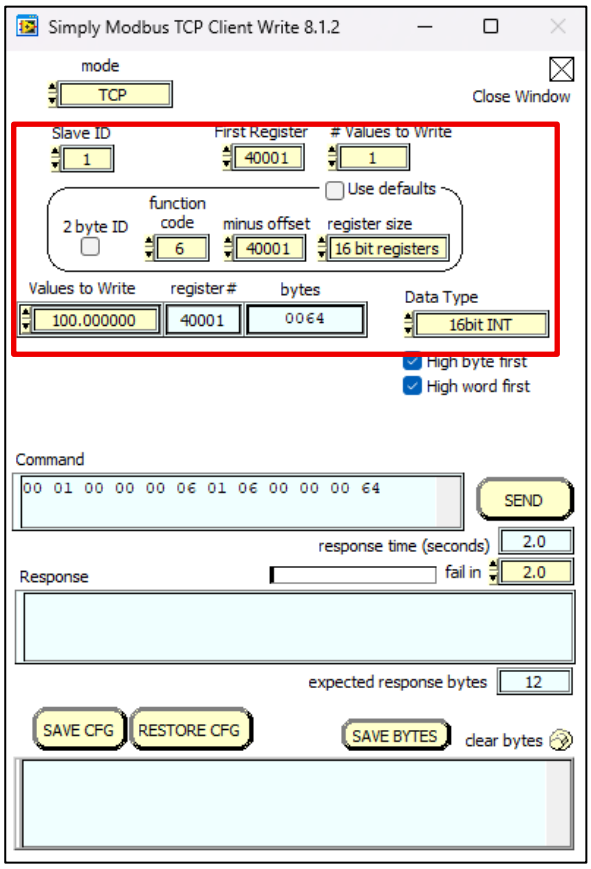
Write a value to Holding Register %QW0 (mapped to Modbus register 40001). The PLC program will read this as input_val and compute output_val = input_val × 2.

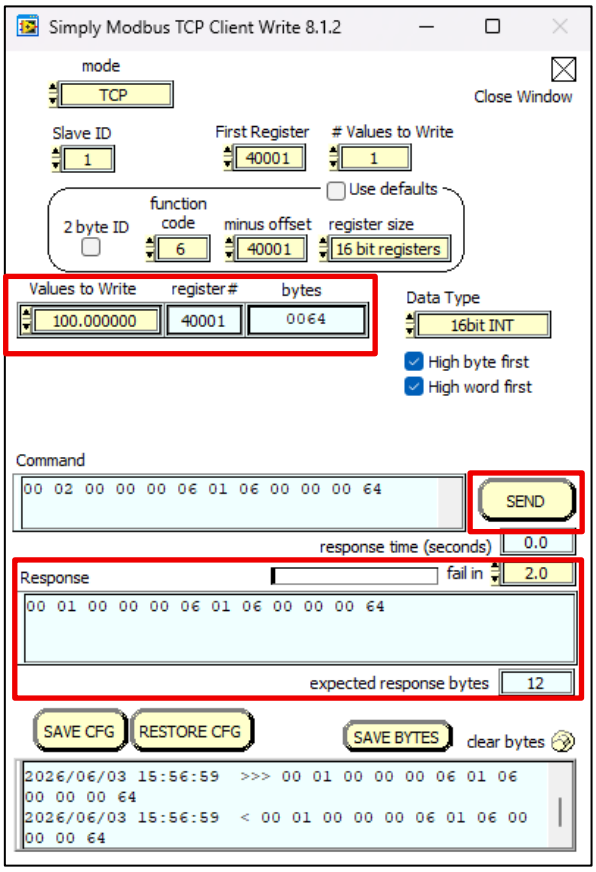
- Open the **Write window**: In the main Simply Modbus window, click the **WRITE** button. A new window will appear.



- Configure the following parameters and click SEND in the write window. Check the Response field:

Field	Value	Description
function code	6	Write Single Register (FC06)
First Register	40001	Maps to %QW0 (input_val)
# Values to Write	1	Write 1 register
Values to Write	100	Test value to write





- Expected response (hex): 0x64 = 100 decimal. Response containing 00 64 (hex for 100) confirms the write was successful.

(12) Read Register Value & Verify

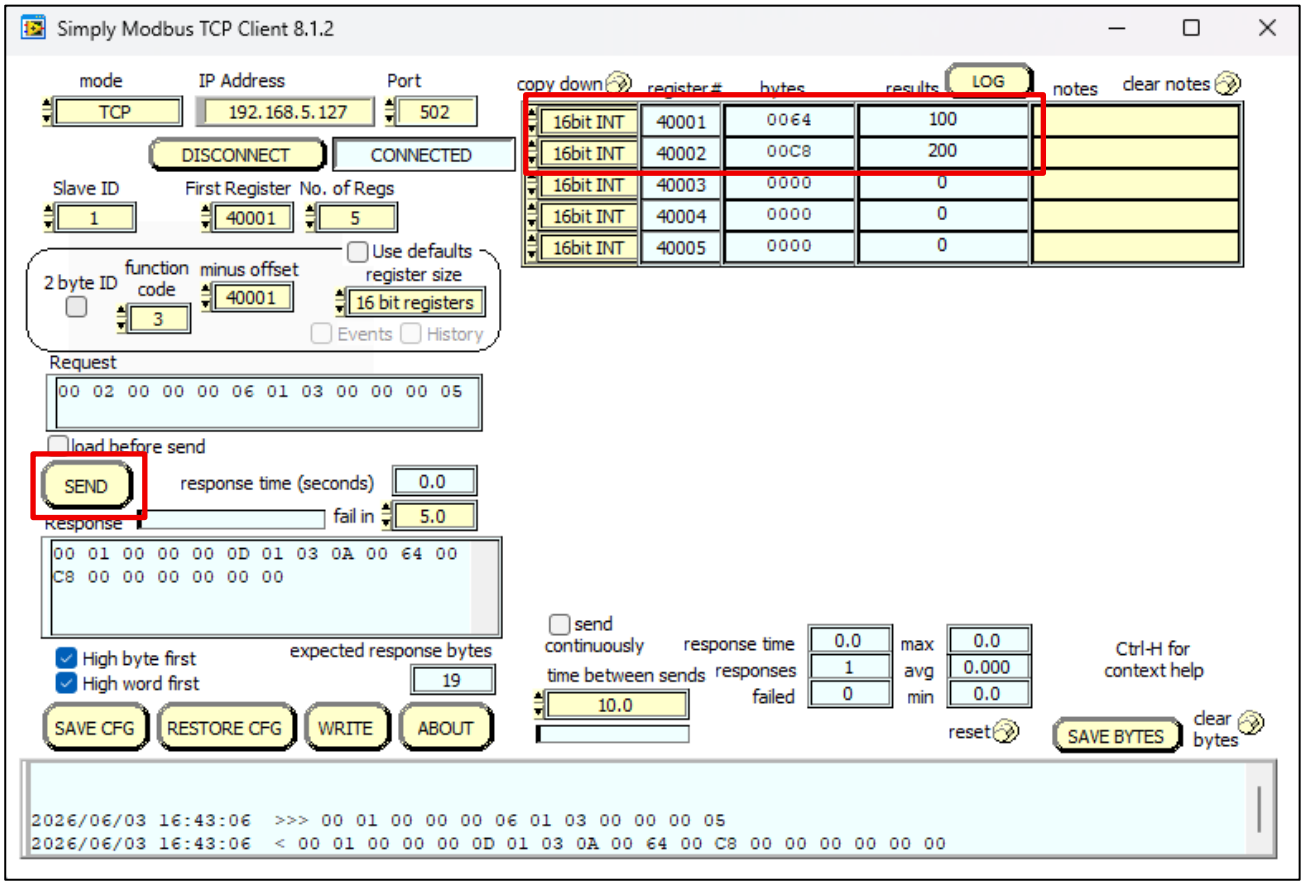
- Return to the main Simply Modbus window.
- Configure reading parameters:

Field	Value	Description
function code	3	Read Holding Registers (FC03)
First Register	40001	Start from register 1
No. of Regs	5	Read 5 registers

- Click **SEND** and verify results

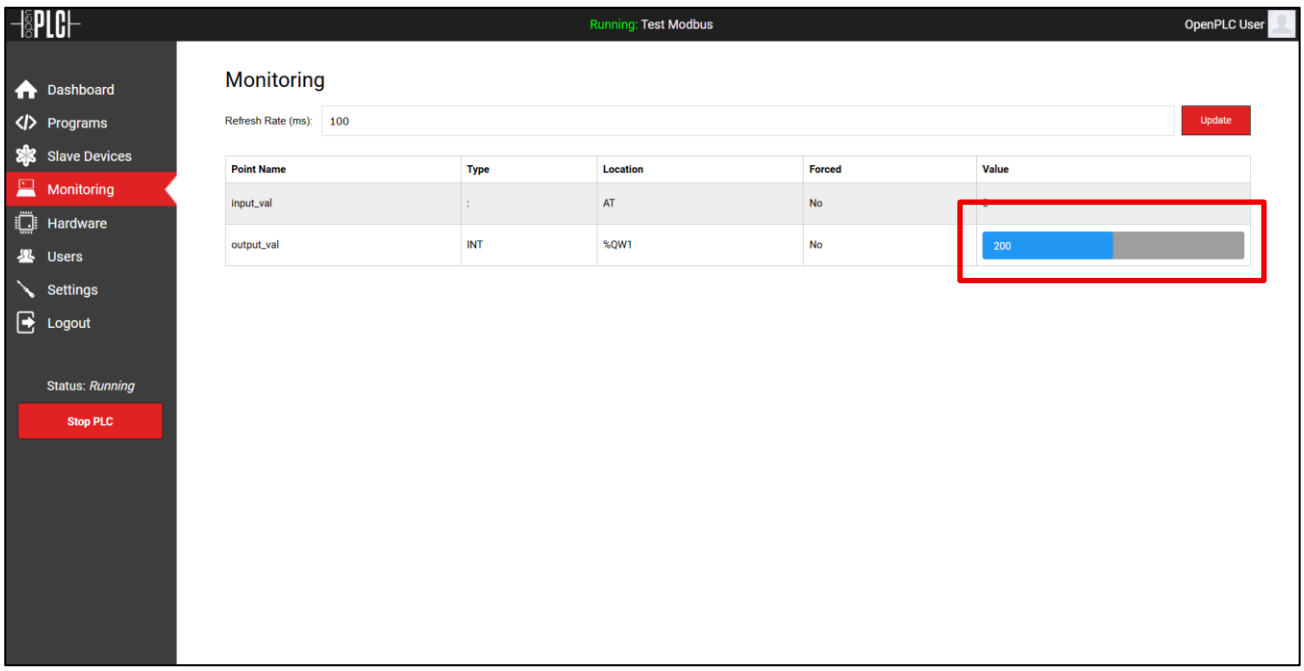
Register	Variable	Expected Value
40001	%QW0 (input_val)	100
40002	%QW1 (output_val)	200
40003–40005	—	0

- Register 40002 showing **200** confirms the PLC program is executing correctly — reading input from Modbus, computing the result, and writing back to the Modbus register.



(13) Monitor Variables on OpenPLC Web UI

- Navigate to Monitoring page: In the browser, click **Monitoring** in the left sidebar:



- Tests with different values:

Write to 40001	**Expected 40002**
50	100
100	200
500	1000
1000	2000

Note: In OpenPLC v3, variables declared with AT addressing (e.g. input_val AT %QW0) may display "AT" in the Location column on the Monitoring page instead of the actual address. This is a known UI display issue in OpenPLC v3 and does not affect the actual program execution. The variable is correctly mapped to its Modbus register and operates as expected.

- Kill webserver v3

```
$ pkill -f "webserver.py"
$ pkill -f "openplc"
```

4.2 OpenPLC Editor Setup and Communication Procedure

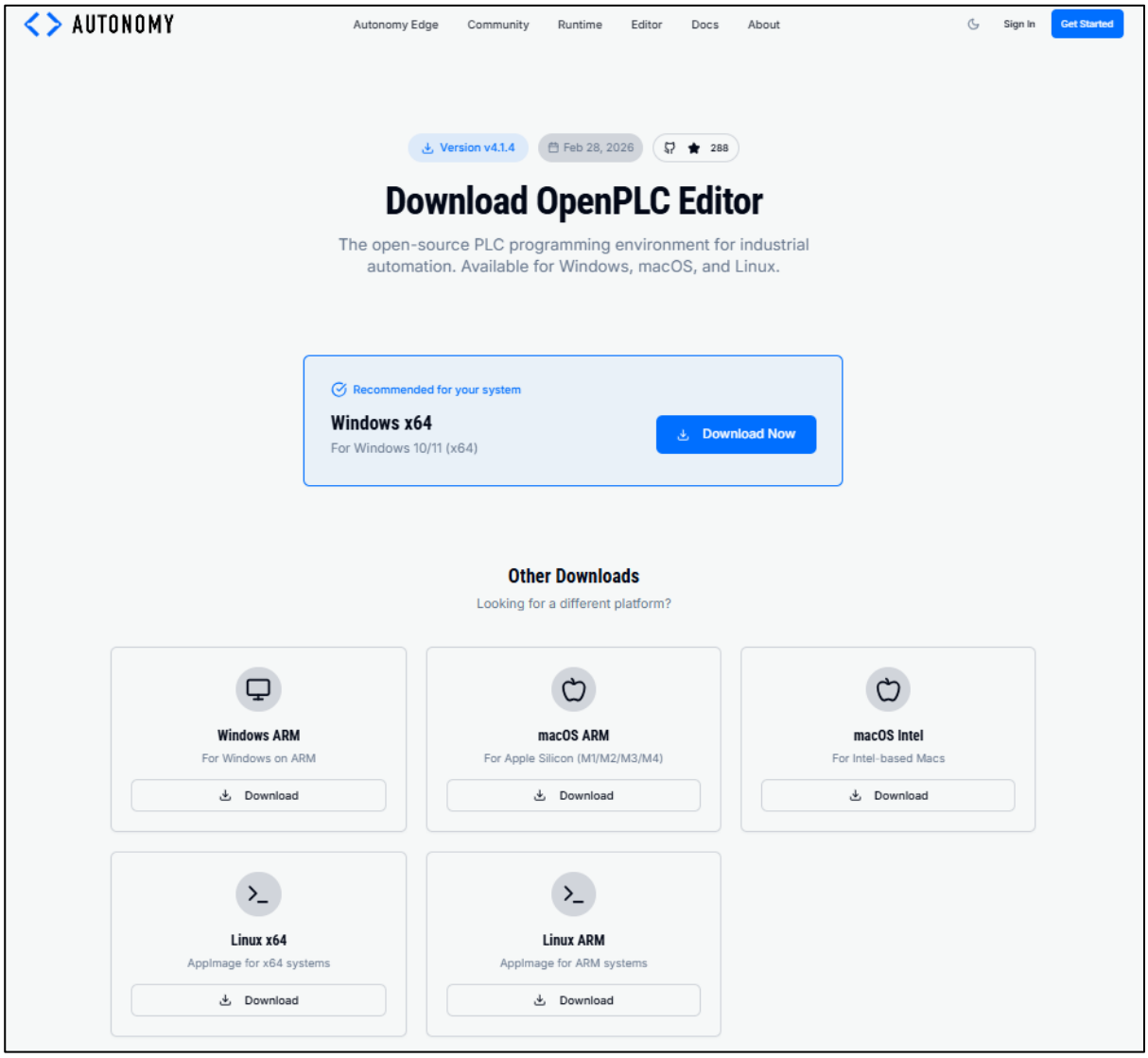
This section describes how to use OpenPLC Editor to create and upload a PLC program to the RZ/T2H evaluation board and verify the communication between the Windows PC and the board via Modbus TCP protocol.

Note: OpenPLC Editor v4 requires OpenPLC Runtime v4 to be installed on the evaluation board. OpenPLC Runtime v3 (installed in the previous section in chapter 3 and chapter 4.1) is not compatible with OpenPLC Editor v4 due to a protocol mismatch — Runtime v3 uses plain HTTP without authentication, while Editor v4 requires HTTPS/TLS with JWT-based authentication on port 8443. This incompatibility exists regardless of which port is configured. Therefore, OpenPLC Runtime v4 must be installed separately on the board to use OpenPLC Editor v4.

(1) Download OpenPLC

The steps may be changed by the vendor, so please follow the instructions of the official site. Visit the site below and download the installer:

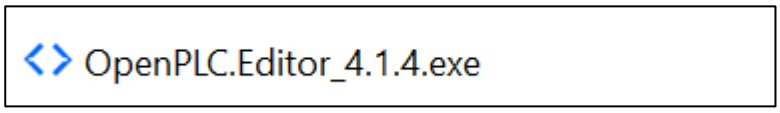
OPENPLC Development System V4 | OPENPLC Store International

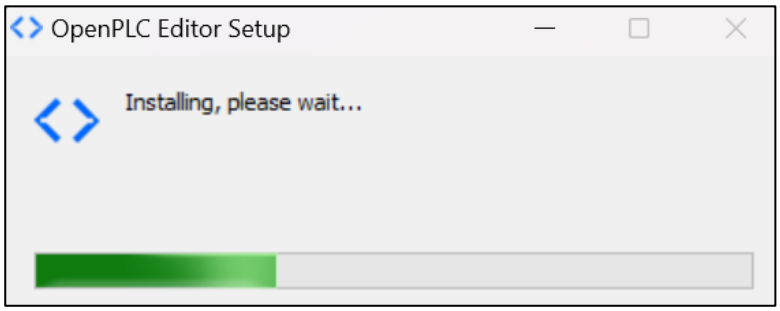


(2) Run the installer and install the OpenPLC

Please follow the installer, there is no problem with the installation if you press "Next>" to proceed. Once the installation is complete, press "Finish". After that, an icon will appear on your desktop. When you start it up, "OpenPLC" will start up.

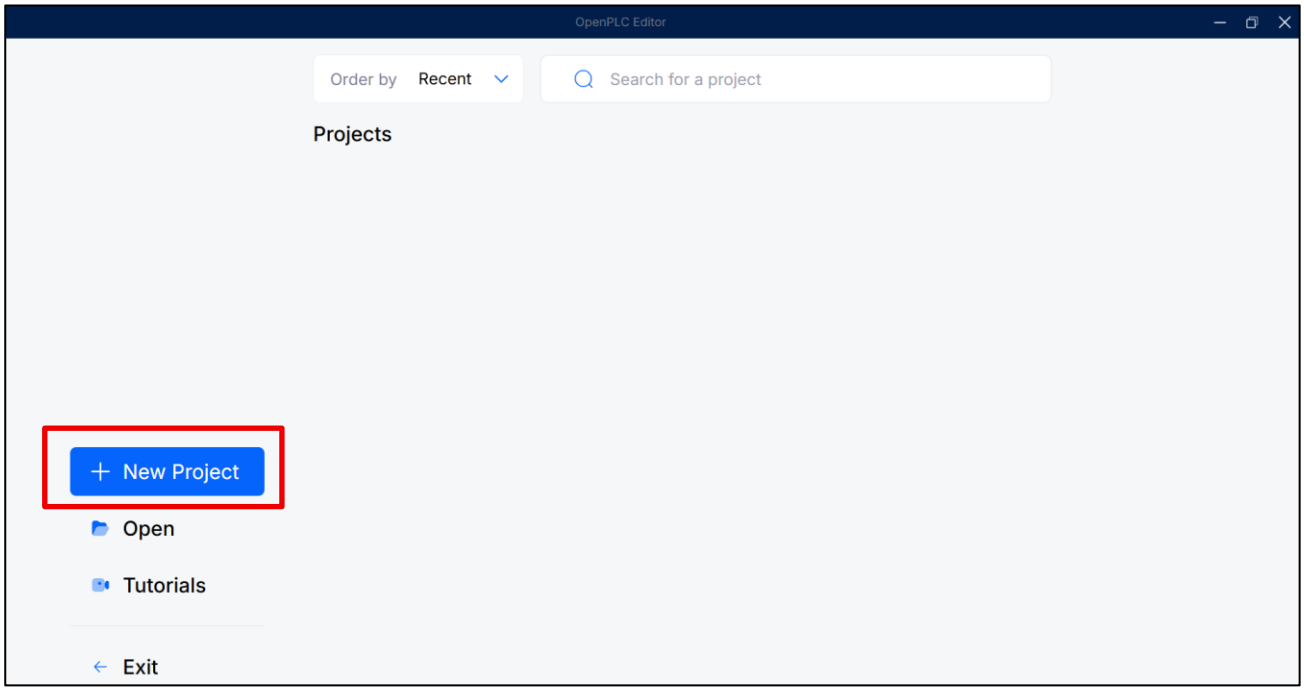
- Double click to install exe file and waiting to finish:



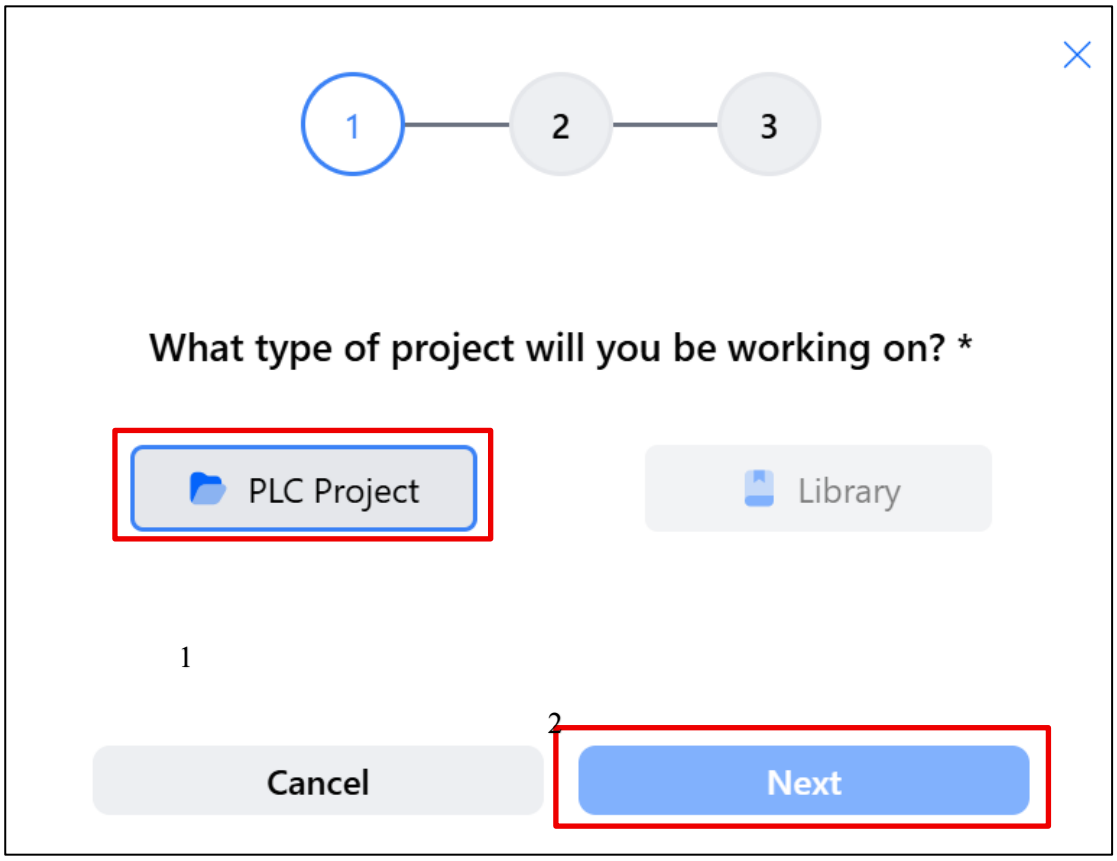


(3) Create new project

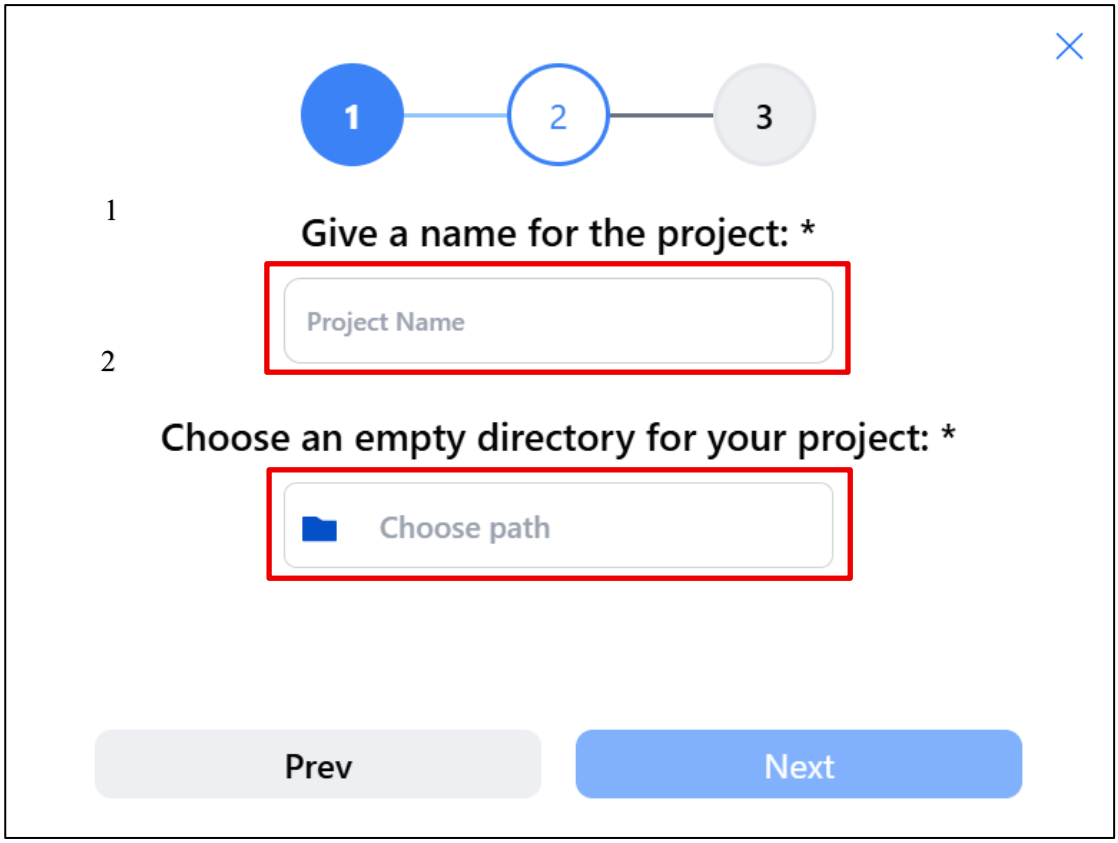
- Click **New Project.**



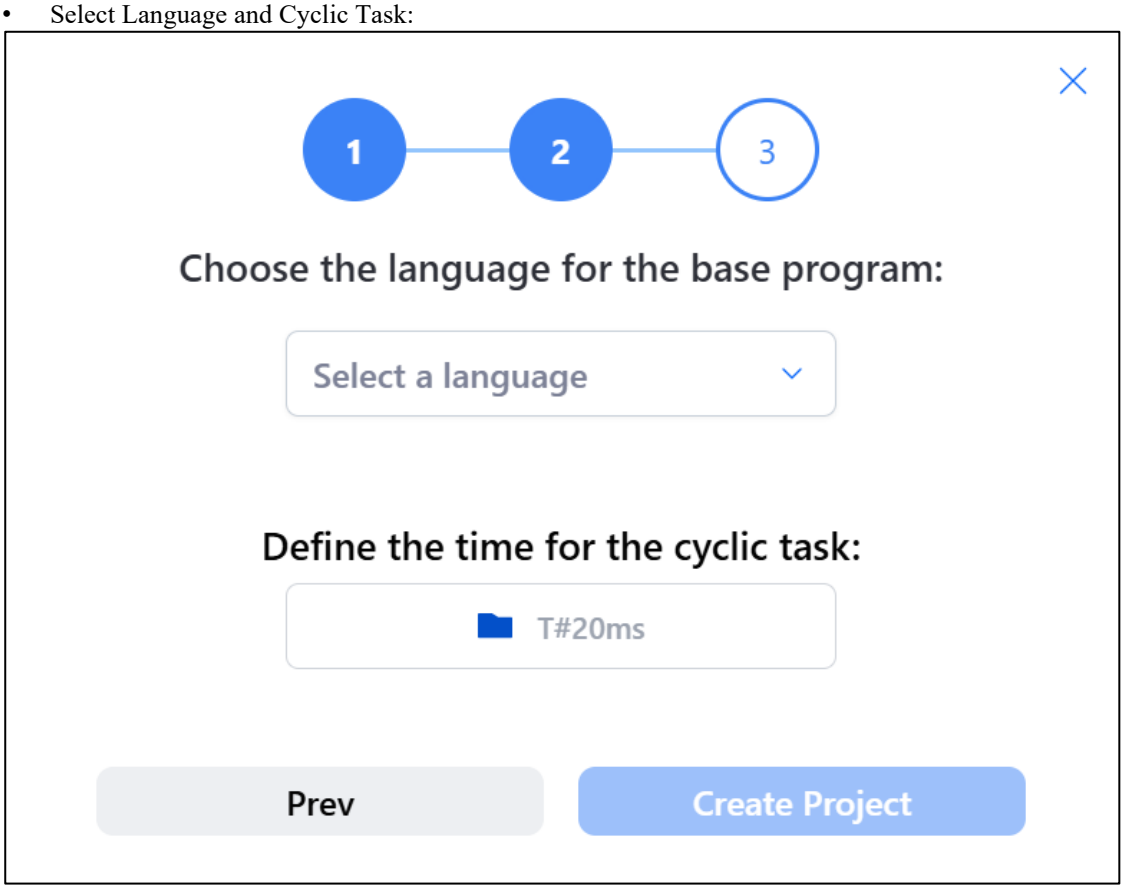
- Click **PLC Project** → **Next**



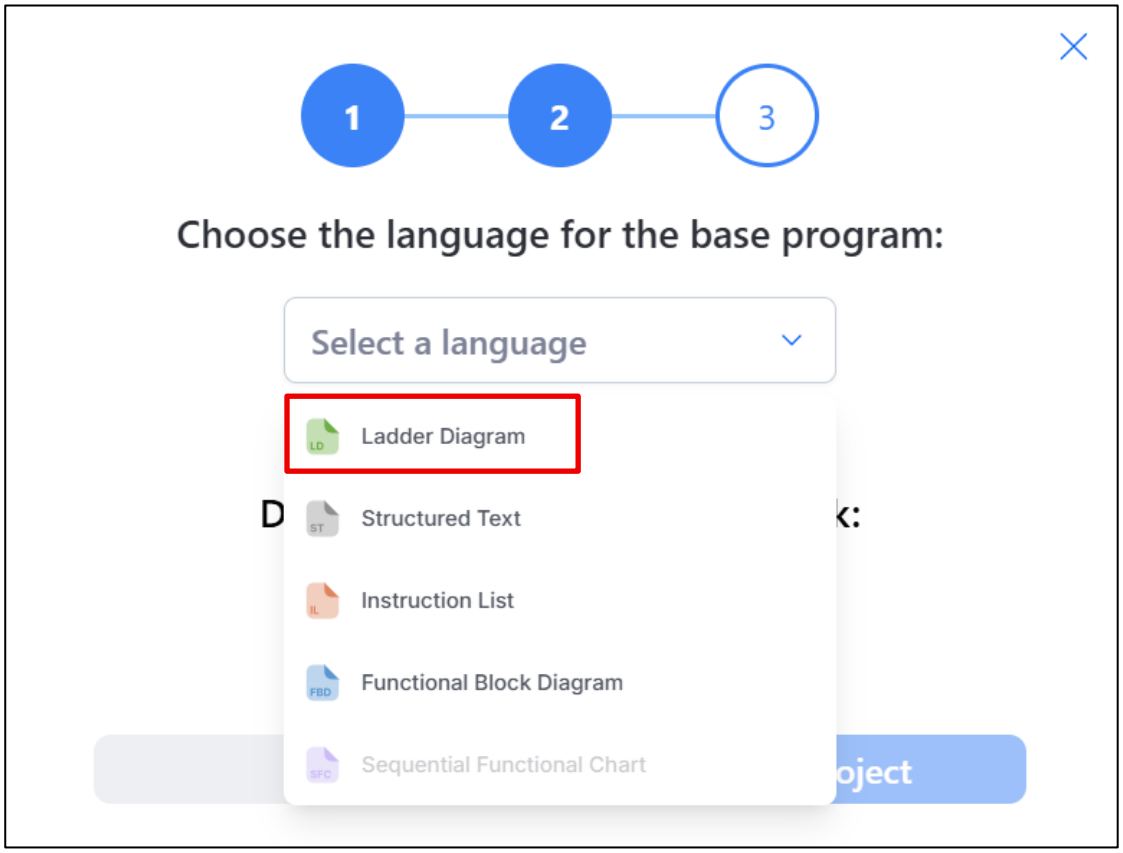
- Give a name and choose empty for project



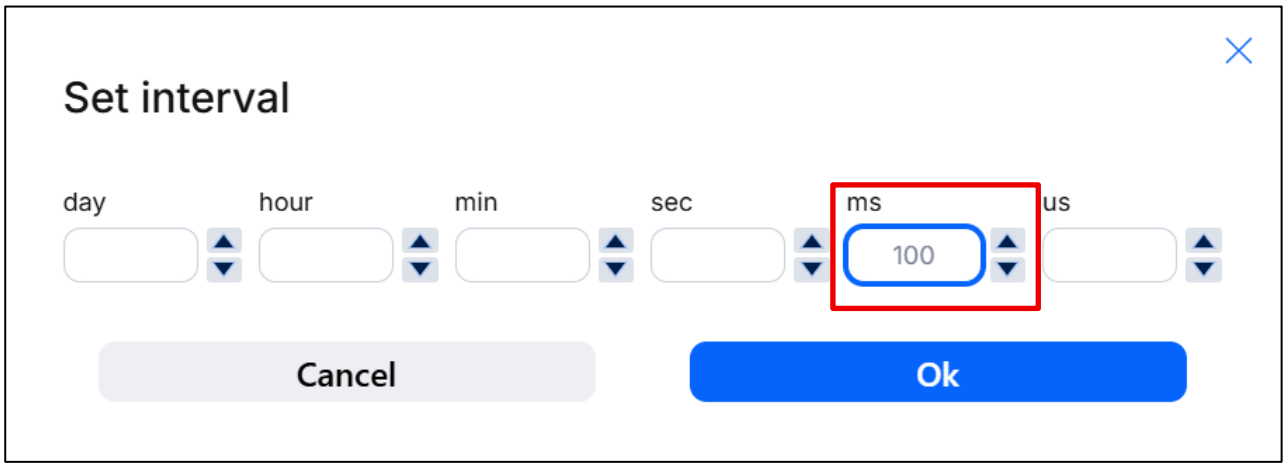
- Select Language and Cyclic Task:



- Select language: **Ladder Diagram**:



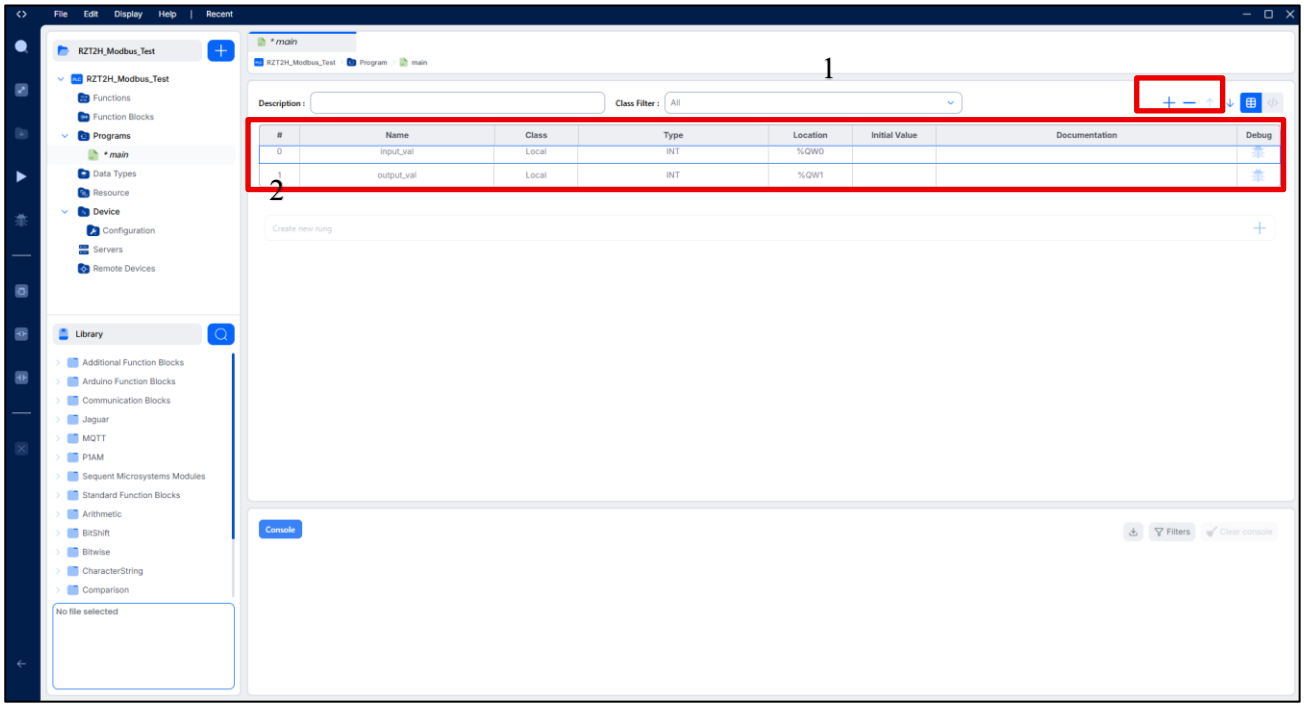
- Define Cyclic task time → OK → Create Project



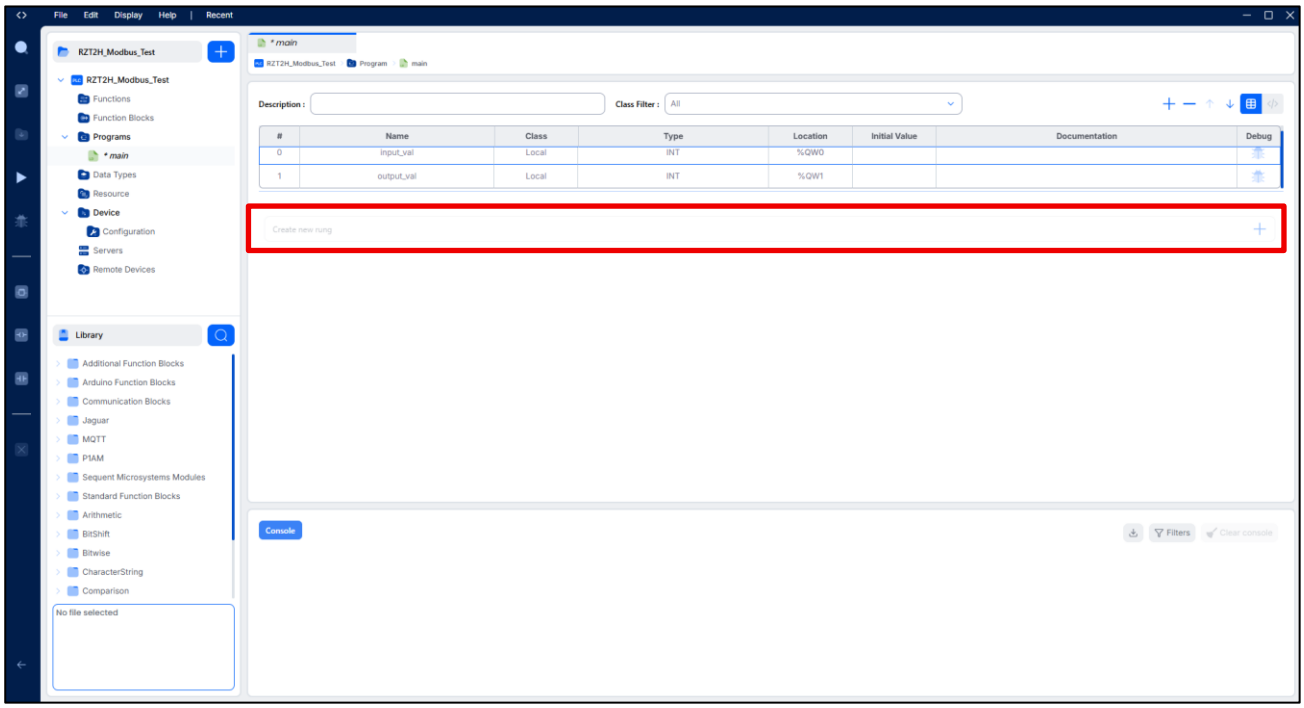
(4) Define Variables

- Click “Plus Icon” → Add **input_val** and **output_val**

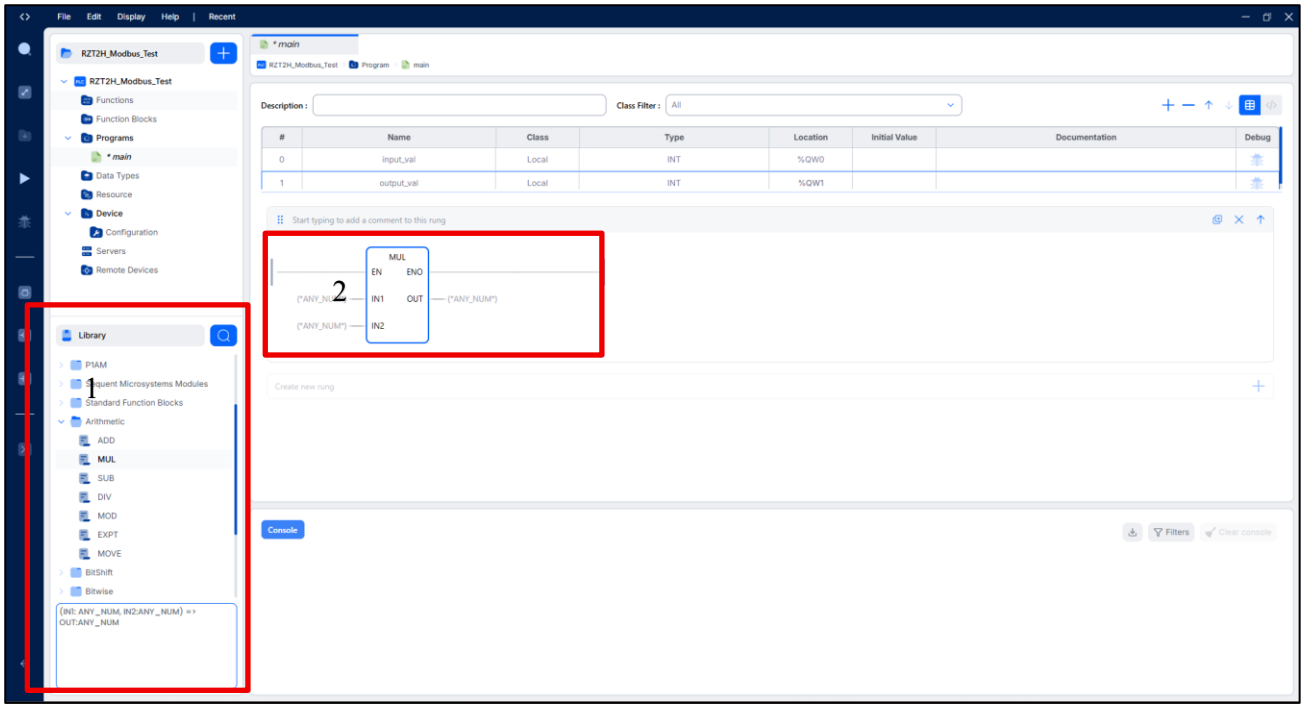
#	Name	Class	Type	Location
0	input_val	Local	INT	%QW0
1	output_val	Local	INT	%QW1



- Create new rung: In the Ladder Diagram editor, click the **"+"** button on the right side of the **"Create new rung"** text to add a new rung. A new rung will appear with a left rail and right rail as shown below.



- Add Function Block **MUL: Library in left side bar → Arithmetic → MUL** → drag and drop it into the center of the rung.

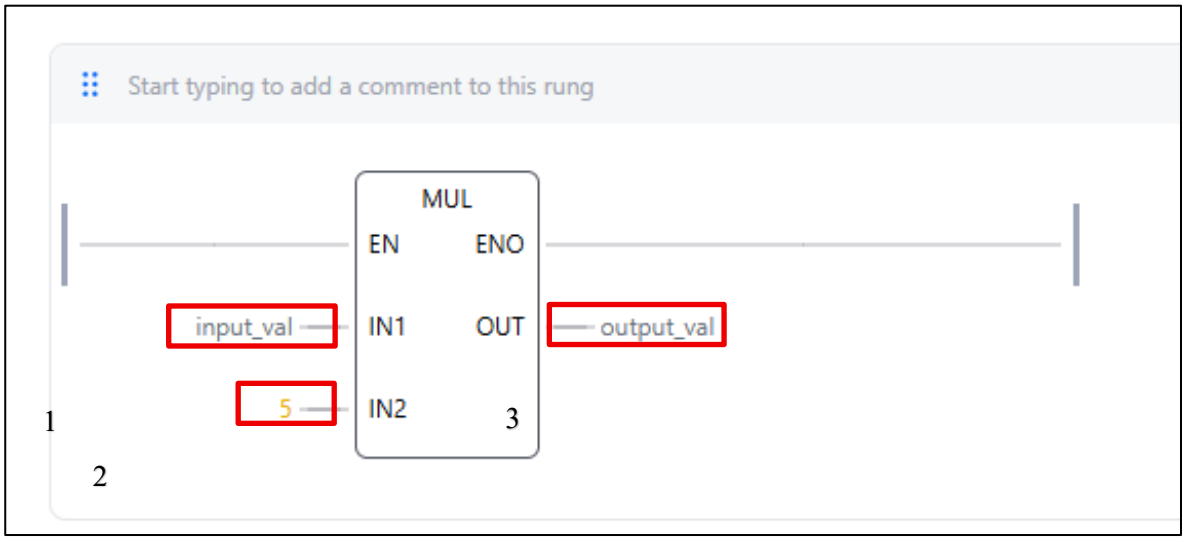


- Assign Variables to MUL Block:

```
Assign IN1 → input_val:
1. Click on (*ANY_NUM*) on the left side of IN1
2. Type input_val → Press Enter
Assign IN2 → value 2:
```

```
1. Click on (*ANY_NUM*) on the left side of IN2
2. Type 2 → Press Enter
Assign OUT → output_val:
1. Click on (*ANY_NUM*) on the right side of OUT
2. Type output_val → Press Enter

Press Ctrl + S to save the project.
```



(5) Install OpenPLC Runtime v4 on the Board

Note: OpenPLC Runtime v4 must be checked out at tag v4.0.9 to ensure compatibility with OpenPLC Editor v4. Using a newer version (e.g. 4.1.0-RC3) will result in a "Runtime Version Mismatch" error when connecting from the Editor.

Run the following commands on the board terminal to install OpenPLC Runtime v4.

- Clone the repository

```
$ git clone https://github.com/Autonomy-Logic/openplc-runtime.git
$ cd openplc-runtime
$ git checkout v4.0.9
```

- Create Python virtual environment and install main dependencies

```
$ python3 -m venv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
$ deactivate
```

- Install plugin virtual environments

```
# modbus_master plugin
$ python3 -m venv venvs/modbus_master
$ source venvs/modbus_master/bin/activate
$ pip install pymodbus==3.11.2
$ deactivate

# modbus_slave plugin
$ python3 -m venv venvs/modbus_slave
$ source venvs/modbus_slave/bin/activate
$ pip install pymodbus==3.11.2
$ deactivate
```

```
# opcua plugin
$ python3 -m venv venvs/opcua
$ source venvs/opcua/bin/activate
$ pip install asyncua
$ deactivate

# runtime venv
python3 -m venv venvs/runtime
source venvs/runtime/bin/activate
pip install -r requirements.txt
deactivate
```

- Build the C core

```
$ mkdir -p build && cd build
$ cmake ..
$ make
$ cd ..
```

- Bypass installation check

```
$ touch /openplc-runtime/.installed
```

(6) Configure Modbus TCP Plugin on the Board

- Create the Modbus slave configuration file:

```
$ cat > /openplc-runtime/core/src/drivers/plugins/python/modbus_slave/modbus_slave_\
config.json << 'EOF'
{
  "network_configuration": {
    "host": "0.0.0.0",
    "port": 502
  }
}
EOF

$ cp /openplc-runtime/plugins_default.conf /openplc-runtime/plugins.conf
```

- Enable the Modbus slave plugin:

```
$ sed -i 's/^modbus_slave,\(.*\),0,0,/modbus_slave,\1,1,0,/' /openplc-runtime/\
plugins.conf
```

- Lock the plugin configuration to prevent it from being overwritten by the Editor on each upload:

```
$ chmod 444 /00_OpenPLC/openplc-runtime/plugins.conf
```

(7) Start OpenPLC Runtime v4

```
$ source .venv/bin/activate
$ ./start_openplc.sh &
$ sleep 8
```

- Create admin account (first time only)

```
$ curl -k -X POST https://127.0.0.1:8443/api/create-user \
  -H "Content-Type: application/json" \
  -d '{"username":"openplc","password":"openplc","role":"admin"}'
```

Note: The create-user step is required only on the first run. If the account already exists, the response will be {"msg":"User already created!"} — this can be safely ignored.

- Verify Runtime v4 is listening on port 8443:

```
$ netstat -tln | grep 8443
```

Expected output:

```
tcp 0 0 0.0.0.0:8443 0.0.0.0:* LISTEN
```

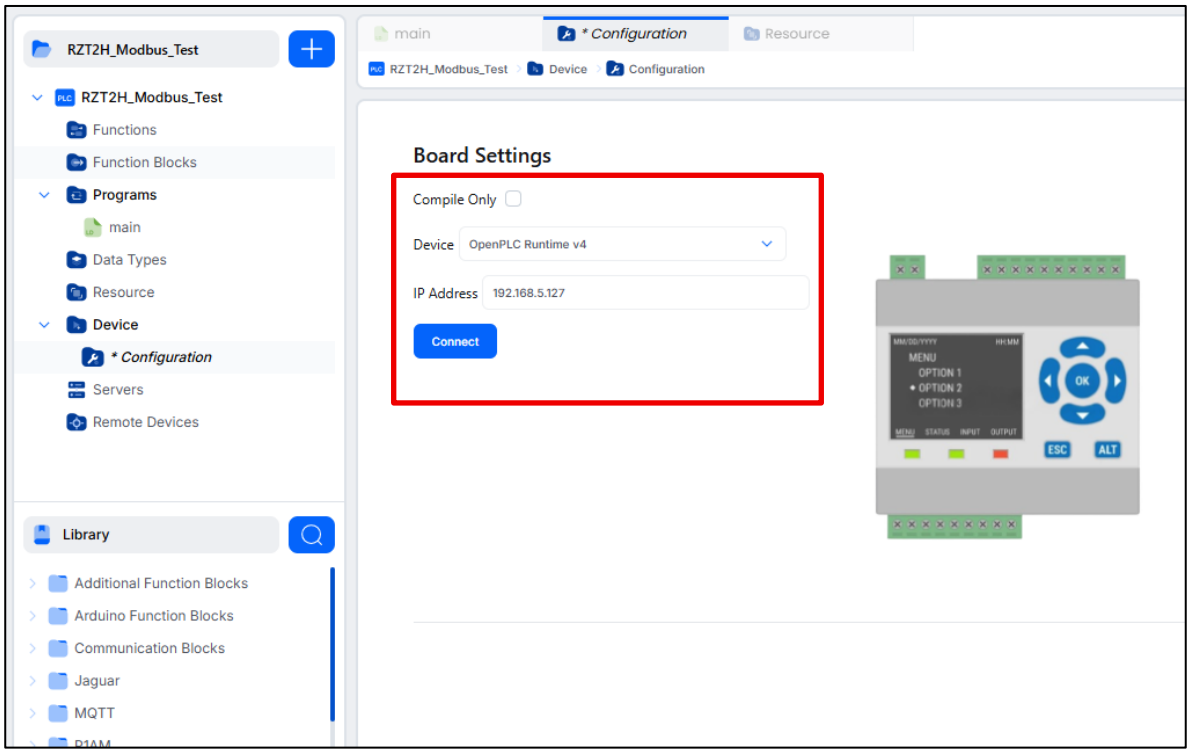
(8) Connect OpenPLC Editor to the Board

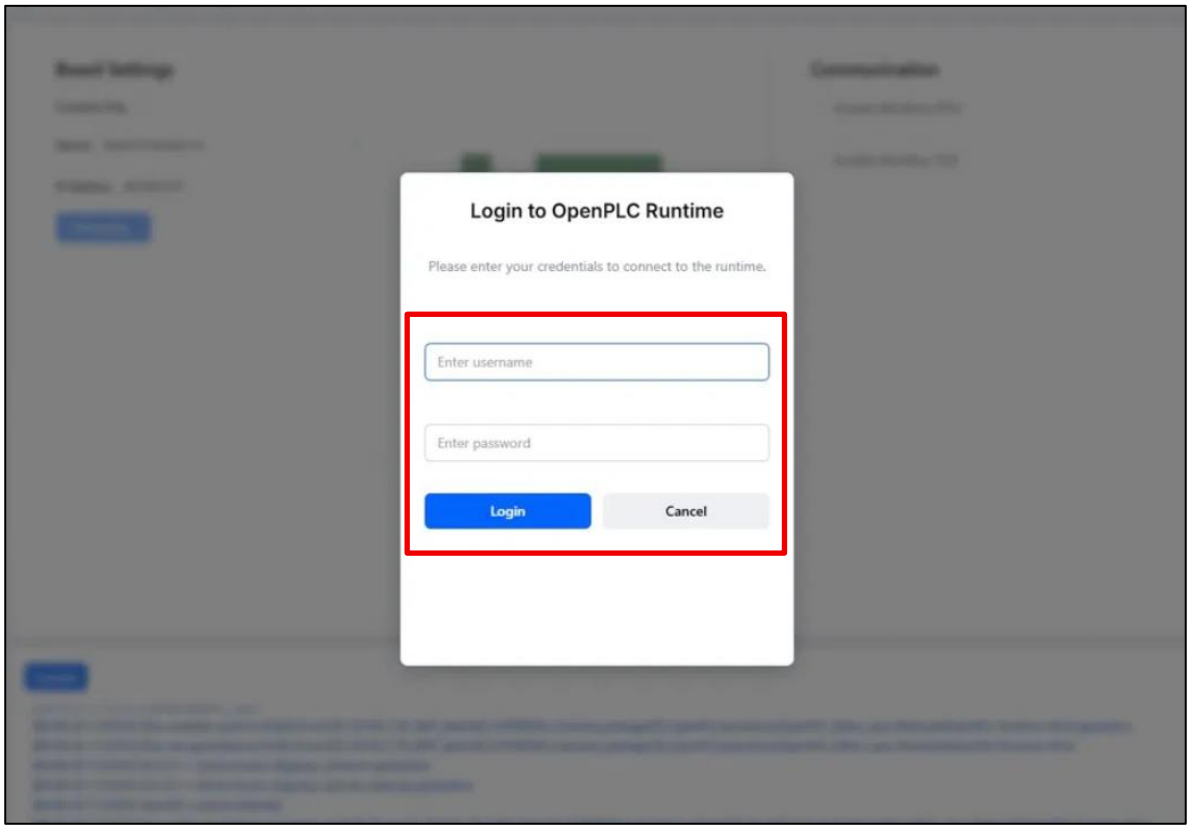
- In OpenPLC Editor, navigate to Device > Configuration and configure as follows:

Device	OpenPLC Runtime v4
IP Address	Your IP Address

- Click Connect. A login dialog will appear. Enter the following credentials:

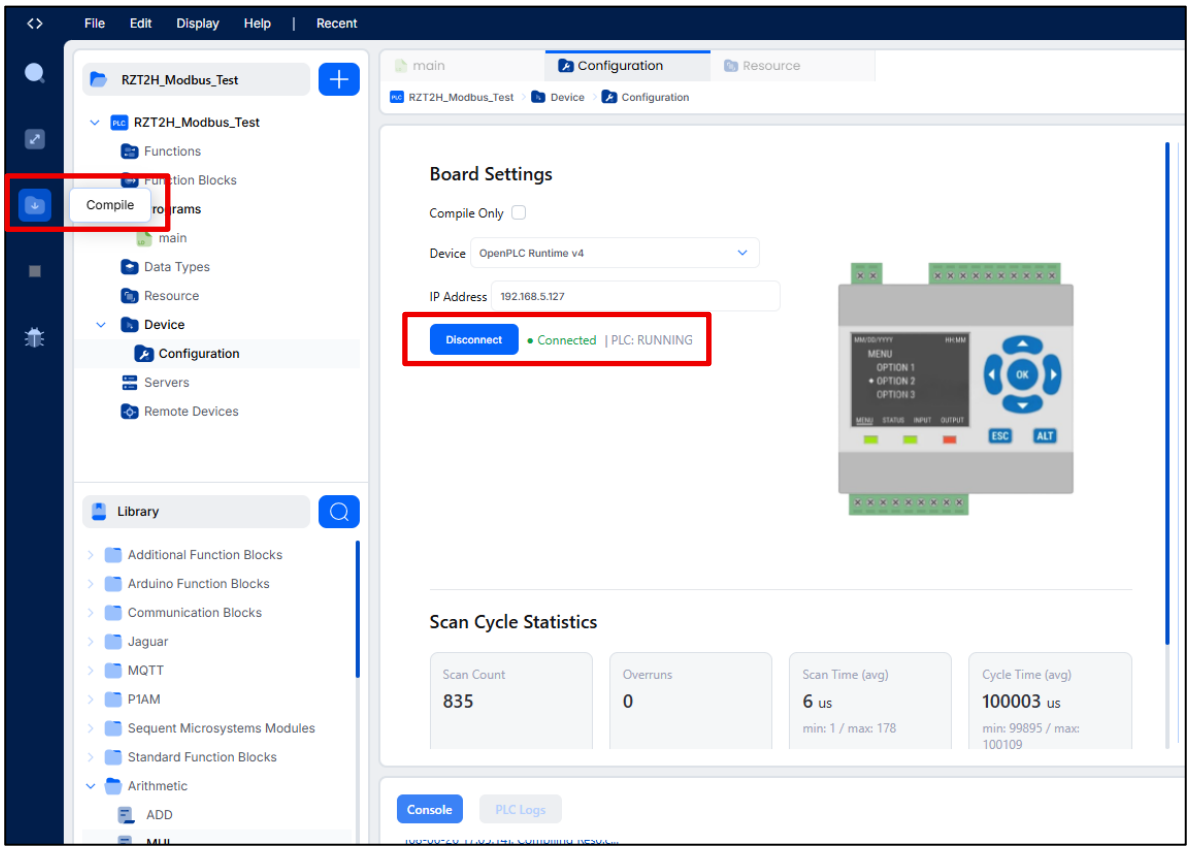
Username	openplc
Password	openplc


RZT2H_Modbus_Test
RZT2H_Modbus_Test
Functions
Function Blocks
Programs
main
Data Types
Resource
Device
* Configuration
Servers
Remote Devices
Library
Additional Function Blocks
Arduino Function Blocks
Communication Blocks
Jaguar
MQTT
main
* Configuration
Resource
RZT2H_Modbus_Test
Device
Configuration
Board Settings
Compile Only
Device
OpenPLC Runtime v4
IP Address
192.168.5.127
Connect
MENU
OPTION 1
OPTION 2
OPTION 3
OK
ESC
ALT


Login to OpenPLC Runtime
Please enter your credentials to connect to the runtime.
Enter username
Enter password
Login
Cancel

(9) Upload and Run the Program

- Now connected OK but PLC will show **EMPTY**
- Click the Run **Compile** button in the left sidebar. The Editor will compile the Ladder Diagram, upload it to the board, and automatically start the PLC. After the process completes, the status should show **Connected | PLC: RUNNING** with Scan Cycle Statistics displayed.



(10) Re-enable Modbus Plugin After Each Upload:

- Each time a new program is uploaded from the Editor, the Modbus slave plugin must be re-enabled on the board. Run the following commands:

```
$ chmod 644 /openplc-runtime/plugins.conf
$ sed -i 's/^modbus_slave,\(.*\),0,0,/modbus_slave,\1,1,0,/' /openplc-runtime/plugins\
.conf
$ chmod 444 /openplc-runtime/plugins.conf
```

- Then two time click Run (triangle play) button in the left sidebar again in the Editor (below compile button) to stop and start the PLC again. Verify port 502 is listening:

```
$ netstat -tln | grep 502
```

Expected output:

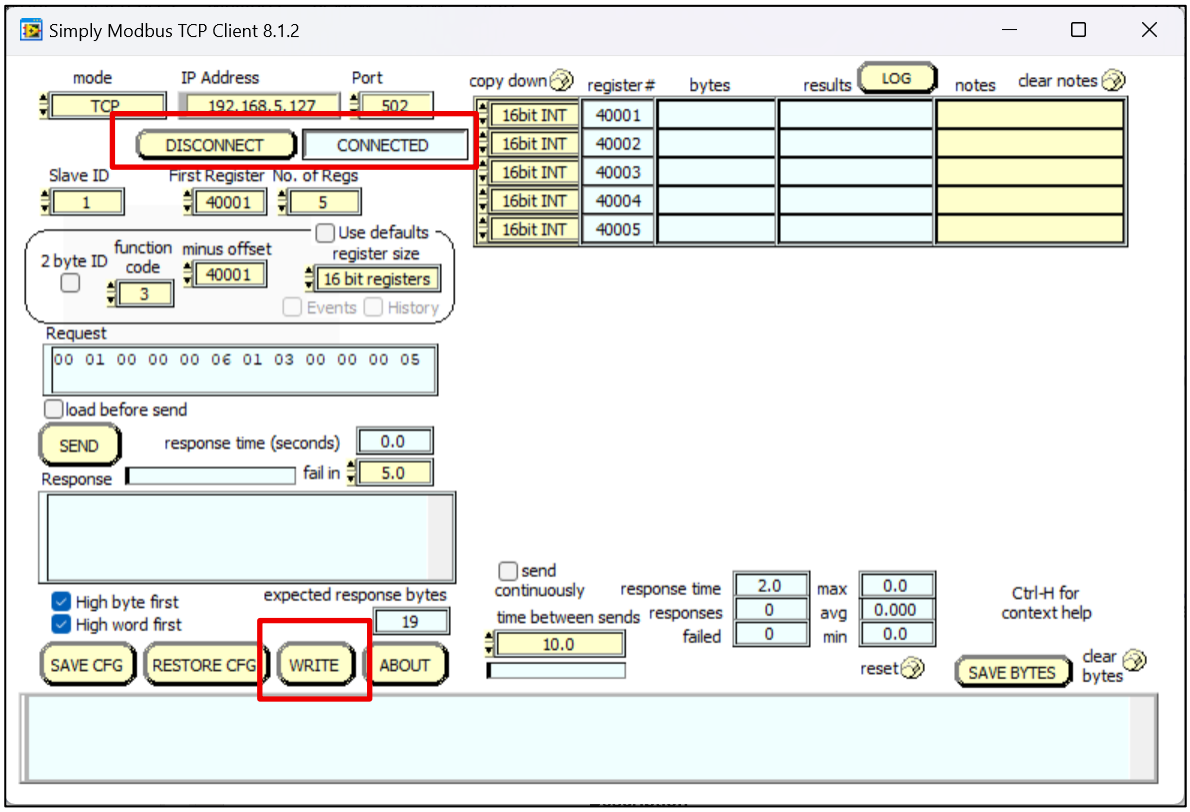
```
tcp  0  0  0.0.0.0:502  0.0.0.0:*  LISTEN
```

(11) Verify Communication with Simply Modbus TCP Client

- Open Simply Modbus TCP Client 8.1.2 on the Windows PC and configure as follows:

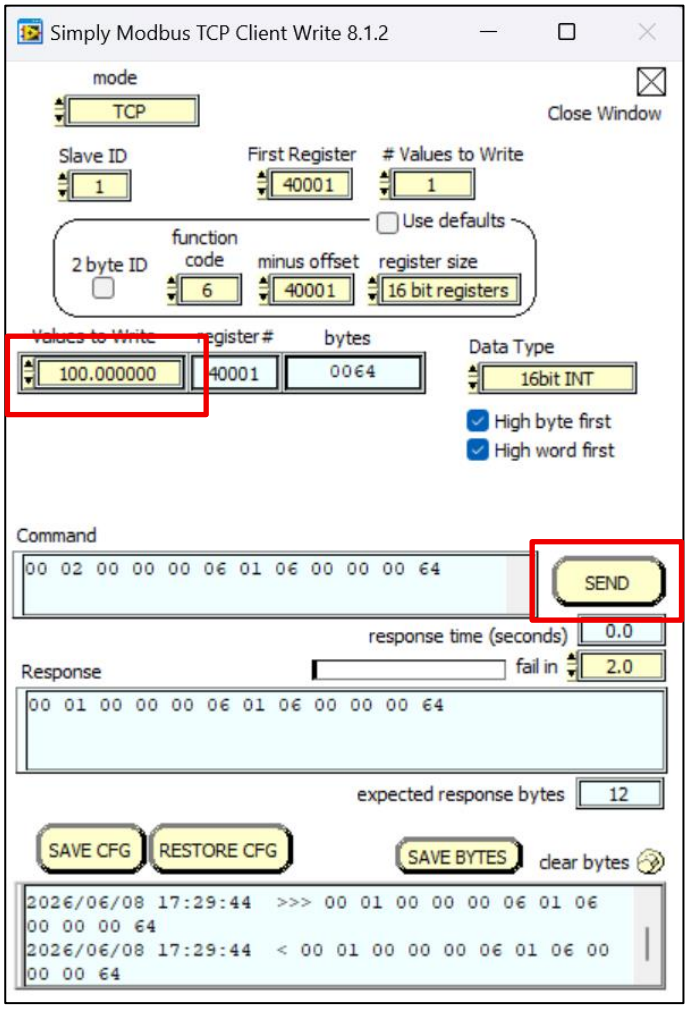
IP Address	192.168.5.127
Port	502
Function code	3 (Read Holding Registers)
First Register	40001
No. of Regs	5

- Click CONNECT, status will change from “NOT CONNECTED” to “CONNECTED



- Then click WRITE to write value 100 to register 40001:

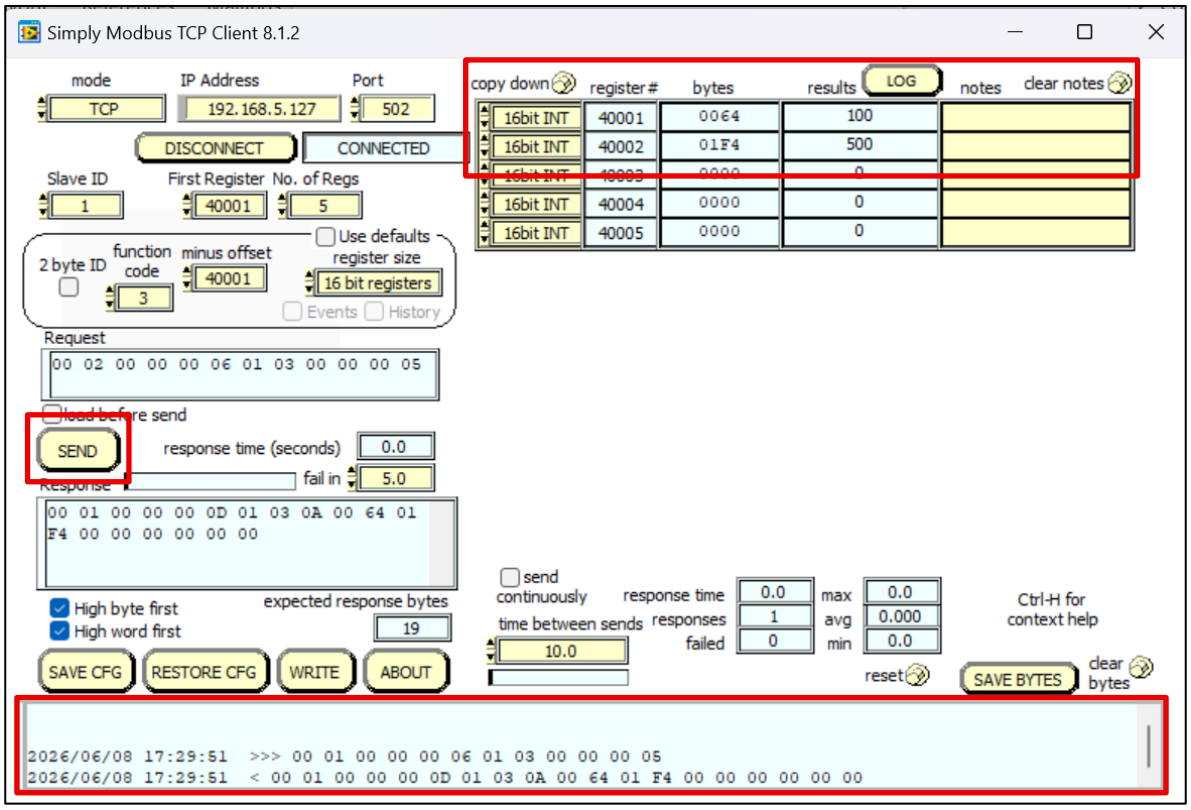
Function code	6 (Write Single Register)
First Register	40001
Values to Write	100



- Click SEND. Then switch back to FC3 and read the registers. Expected results:

Register	Variable	Expected Value
40001	%QW0 (input_val)	100
40002	%QW1 (output_val)	500

Register 40002 showing 500 (= 100 x 5) confirms the Ladder Diagram program uploaded from OpenPLC Editor is executing correctly on the RZ/T2H board and communicating via Modbus TCP.



NOTE: After each program upload from OpenPLC Editor, the Modbus slave plugin is disabled automatically. Always run the re-enable commands in step 4.2 (10) before testing with Simply Modbus TCP Client.

(12) Summary

Component	Role	Protocol	Port
OpenPLC Editor v4	Write Ladder Diagram, compile and upload PLC program, start/stop PLC	HTTPS	8443
OpenPLC Runtime v4	Execute PLC program on board, manage plugins, expose REST API	HTTPS	8443
Modbus Slave Plugin	Map PLC variables (%QW0, %QW1) to Modbus holding registers (40001, 40002)	Modbus TCP	502
Simply Modbus TCP Client	Write input value to register 40001, read output value from register 40002 to verify	Modbus TCP	502

- Complete flow:

1. Editor upload program → Runtime compile → PLC RUNNING
2. Simply Modbus write 100 → register 40001 → %QW0 = 100
3. PLC caculate: output = input × 5 → %QW1 = 500
4. Simply Modbus read → register 40002 → hiện 500

5. Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec. 24, 2025	-	First edition issued.
1.01	Jul. 31, 2026	-	Update Chapter 2 and Chapter 4.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.