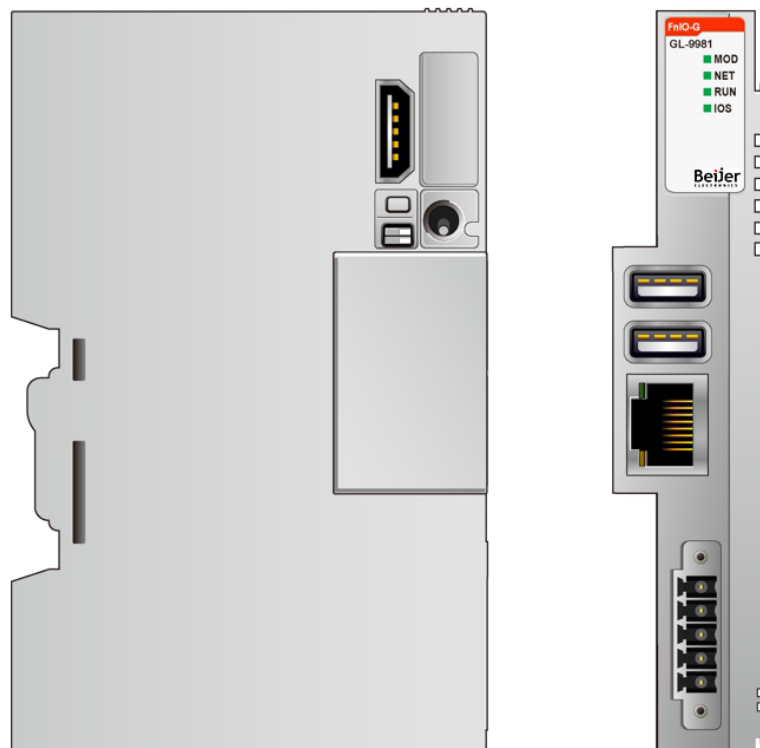


# User Manual

## GL-9981-C-1, GL-9981-L Programmable IO Module

Embedded PIO, Open Linux (Debian), Docker support

Doc ID: 53954  
2025-09-17



The information in this document is subject to change without notice and is provided as available at the time of printing. Beijer Electronics AB reserves the right to change any information without updating this publication. Beijer Electronics AB assumes no responsibility for any errors that may appear in this document. All examples in this document are only intended to improve understanding of the functionality and handling of the equipment. Beijer Electronics AB cannot assume any liability if these examples are used in real applications. In view of the wide range of applications for this software, users must acquire sufficient knowledge themselves in order to ensure that it is correctly used in their specific application. Persons responsible for the application and the equipment must themselves ensure that each application is in compliance with all relevant requirements, standards, and legislation in respect to configuration and safety. Beijer Electronics AB will accept no liability for any damage incurred during the installation or use of equipment mentioned in this document. Beijer Electronics AB prohibits all modification, changes, or conversion of the equipment.

**Head Office**

Beijer Electronics AB

Box 426

201 24 Malmö, Sweden

[www.beijerelectronics.com](http://www.beijerelectronics.com) / +46 40 358600

# Table of Contents

1. About This Manual	5
1.1. Symbols Used in This Manual	5
1.2. Modules Covered in This Manual	5
2. Safety	6
2.1. Product Certifications	6
2.2. General Safety Requirements	6
3. About the G-series System	7
3.1. IO Process Data Mapping	8
4. Specifications	9
4.1. Enviromental Specifications	9
4.2. General Specifications	9
4.3. Interface Specifications	10
4.4. GL-9981-L Specifications	10
4.5. GL-9981-C-1 Specifications	10
5. Connectors and Ports	12
5.1. Wiring Diagram	12
5.2. RJ-45 Socket	12
5.3. USB 2.0 Port	13
5.4. Monitor Port	13
6. Buttons and Switches	15
6.1. Button	15
6.2. Toggle Switch	15
6.3. DIP Switch	15
7. LED Indicator	16
7.1. MOD (Module Status)	16
7.2. NET (Network Status)	17
7.3. RUN	17
7.4. IOS (Expansion Module Status)	17
8. Hardware Setup	19
8.1. Enviromental Requirements	19
8.2. Space Requirements	20
8.3. Mount Module to DIN Rail	21
8.4. Mount Removable Terminal Block	23
8.5. Connect Cables to Removable Terminal Block	24
9. How to Start the Module	25
10. IP Settings	26
11. Configuration Diagrams	27
11.1. Programming Configuration Diagram	27
11.2. System Configuration Diagram	28
12. CODESYS	29
12.1. Download CODESYS Template Project	29
12.2. Create a New CODESYS Project	29
12.3. Example: Activate CODESYS License	31
12.4. Add a Module XML to CODESYS	31
12.5. I/O Control Functions in CODESYS Project Example	32
12.6. Add the Persistence Manager	34
13. System Setup (GL-9981-L)	36
13.1. Install the OS	36
13.2. Install Docker	38
13.3. Install CODESYS Runtime	44
13.4. Delete CODESYS Runtime	45
13.5. Login Credentials	46
13.6. Controll I/O with C Language	46

---

13.7. Functions Used in PibusloTest.c .....	46
13.8. Compile PibusloTest.c in the Terminal .....	47
13.9. Use the Geany Editor on Raspberry Pi .....	48

# 1. About This Manual

This manual contains information on the software and hardware features of the Beijer Electronics GL-9981 Programmable IO Module. It provides in-depth specifications, guidance on installation, setup, and usage of the product.

## 1.1. Symbols Used in This Manual

This publication includes Warning, Caution, Note and Important icons where appropriate, to point out safety-related, or other important information. The corresponding symbols should be interpreted as follows:



### WARNING

Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury, and major damage to the product.



### CAUTION

Indicates a potentially hazardous situation which, if not avoided, could result in minor or moderate injury, and moderate damage to the product.



### IMPORTANT

Highlights key information.



### NOTE

Points out relevant facts and conditions.



### TIP

Provides useful, non-essential information to assist you.

## 1.2. Modules Covered in This Manual

Module	Type	Other
GL-9981-C-1	Programmable IO	Open Linux (Debian), Docker support, CODESYS, OPC UA server, 16 GB (eMMC), 2 GB (DDR4), RJ-45, USB 2.0, max. 16 slices.
GL-9981-L	Programmable IO	Open Linux (Debian), Docker support, 16 GB (eMMC), 2 GB (DDR4), RJ-45, USB 2.0.

## 2. Safety

Before using this product, please read this manual and other relevant manuals carefully. Pay full attention to safety instructions!

In no event will Beijer Electronics be responsible or liable for damages resulting from the use of this product.

The images, examples and diagrams in this manual are included for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Beijer Electronics cannot take responsibility or liability for actual use based on the examples and diagrams.

### 2.1. Product Certifications

The product has the following product certifications.



### 2.2. General Safety Requirements



#### WARNING

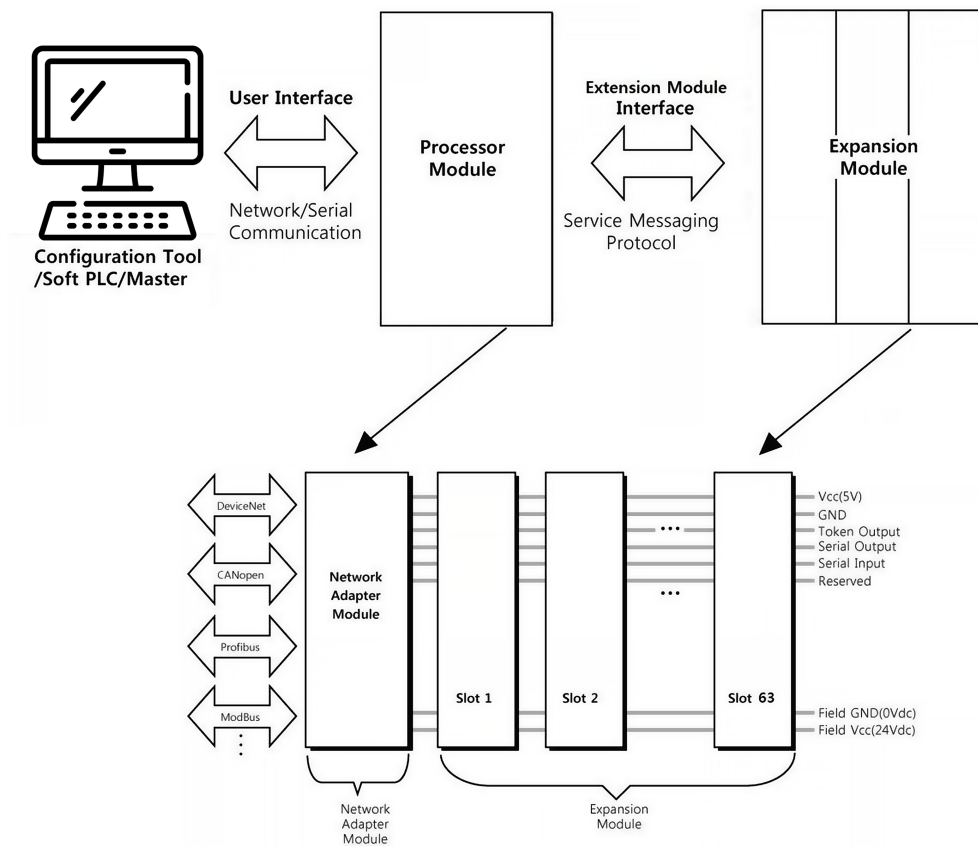
- Do not assemble the products and wires with power connected to the system. Doing so cause an "arc flash", which can result in unexpected dangerous events (burns, fire, flying objects, blast pressure, sound blast, heat).
- Do not touch terminal blocks or IO modules when the system is running. Doing so may cause electric shock, short circuit or malfunction of the device.
- Never let external metallic objects touch the product when the system is running. Doing so may cause electric shock, short circuit or malfunction of the device.
- Do not place the product near inflammable material. Doing so may cause a fire.
- All wiring work should be performed by an electrical engineer.
- When handling the modules, ensure that all persons, the workplace and the packing are well grounded. Avoid touching conductive components, the modules contain electronic components that may be destroyed by electrostatic discharge.



#### CAUTION

- Never use the product in environments with temperature over 60°C. Avoid placing the product in direct sunlight.
- Never use the product in environments with over 90% humidity.
- Always use the product in environments with pollution degree 1 or 2.
- Use standard cables for wiring.

### 3. About the G-series System

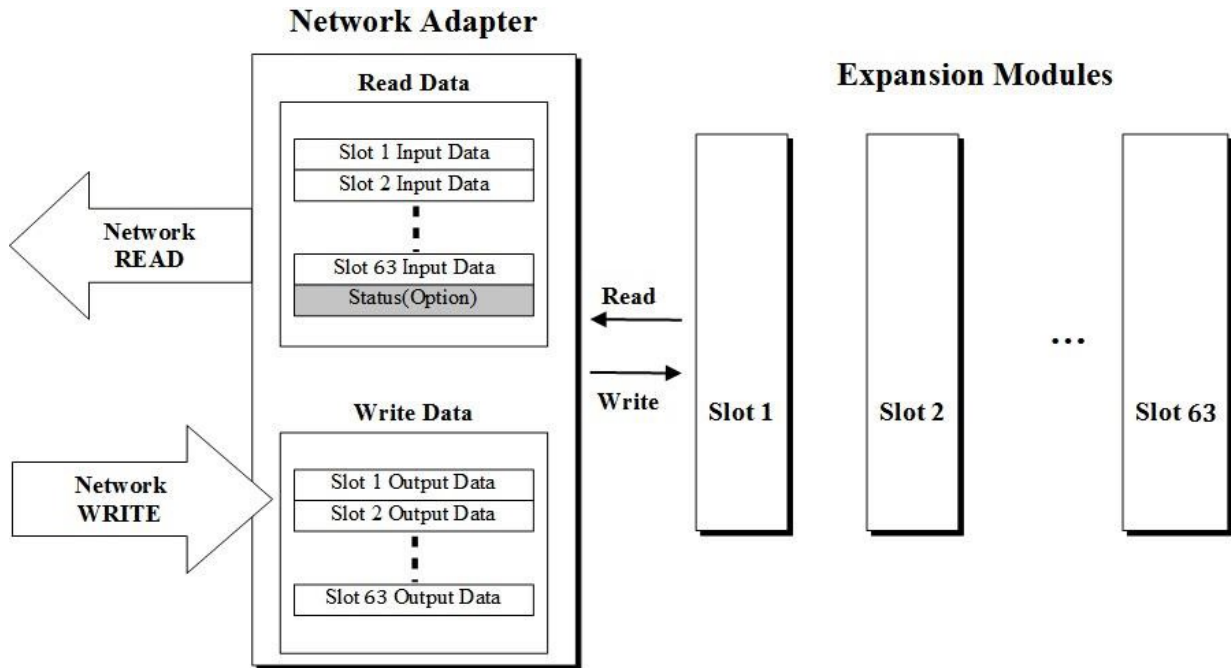


#### System overview

- **Network Adapter Module** - The network adapter module forms the link between the field bus and the expansion modules. The connection to different field bus systems can be established by each of the corresponding network adapter module, e.g., for MODBUS TCP, Ethernet IP, EtherCAT, PROFINET, CC-Link IE Field, PROFIBUS, CANopen, DeviceNet, CC-Link, MODBUS/Serial etc.
- **Expansion Module** - Expansion module types: Digital IO, Analog IO, and Special modules.
- **Messaging** - The system uses two types of messaging: Service messaging and IO messaging.

### 3.1. IO Process Data Mapping

An expansion module has three types of data: IO data, configuration parameter, and memory register. The data exchange between the network adapter and the expansion modules is made via IO process image data by internal protocol.



*Data flow between network adapter (63 slots) and expansion modules*

The input and output image data depend on the slot position and the data type of the expansion slot. The ordering of input and output process image data is based on the expansion slot position. Calculations for this arrangement are included in the manuals for network adapter and programmable IO modules.

Valid parameter data depends on the modules in use. For example, analog modules have settings of either 0-20 mA or 4-20 mA, and temperature modules have settings such as PT100, PT200, and PT500. The documentation for each module provides a description of the parameter data.

## 4. Specifications

### 4.1. Environmental Specifications

Operating temperature	-20°C - 50°C
Relative humidity	5% - 90% non-condensing
Mounting	DIN rail
Shock operating	IEC 60068-2-27 (15G)
Vibration resistance	IEC 60068-2-6 (4 g)
Industrial emissions	EN 61000-6-4: 2019
Industrial immunity	EN 61000-6-2: 2019
Installation position	Vertical and horizontal
Product certifications	CE, UKCA



#### CAUTION

Always read the [Hardware Setup](#) chapter before installing the module.

### 4.2. General Specifications

UL system power	Supply voltage: 24 VDC nominal, Class 2
System power*	Supply voltage: 24 VDC nominal Supply voltage range: 15 - 28.8 VDC Reverse polarity protection
Power dissipation	225 mA typical @ 24 VDC
Current for IO module**	0.5 A @ 5 VDC
Isolation	System power to internal logic: Non-isolation System power IO driver: Isolation
UL field power	Supply voltage: 24 VDC nominal, Class2
Field power	Supply voltage: 24 VDC typical (Max. 28.8 VDC) Field power range is different depending on IO module series. See the module user manual.
Max. current field power contact	DC 8 A
Wiring	0.05 mm <sup>2</sup> - 1.31 mm <sup>2</sup> (30-16 AWG)
Weight	87 g
Module size	22.5 mm x 109 mm x 70 mm
<p>* Since the product does not include UPS, removing the power before soft shutdown can cause data damage. Make sure it has been shut down normally and then disconnect the power.</p> <p>** If the load used exceeds the specifications, a throttling mode may occur due to heat, resulting in degraded performance or abnormal operation. The internal temperature must be less than 85°C.</p>	

### 4.3. Interface Specifications

<b>PIO Type</b>	Embedded type programmable IO
<b>Module versions</b>	GL-9981-L: Linux version GL-9981-C-1: CODESYS version
<b>CPU</b>	ARM Cortex-A72 @ 1 GHz
<b>OS</b>	Linux Debian
<b>RAM</b>	LPDDR4 2 GB
<b>eMMC flash memory</b>	16 GB
<b>Interface connectors</b>	RJ-45 socket x 1 USB 2.0 port x 2 Monitor port x 1
<b>Ethernet baud rate</b>	100 Mbps, Auto-negotiation, Full duplex
<b>LED Indicators</b>	Module status (MOD) - Green Network status (NET) - Green Custom (RUN) - Green/Red Expansion IO module status (IOS) - Green/Red
<b>RTC</b>	Retain time: < 15 day/accuracy, < 2 min/month. Status: Fully recharged battery at room temperature. For more information, refer to the chapter <b>Battery Charging</b> below.

#### 4.3.1. Battery Charging

Recommend charging when the battery is discharged is **16 hours**.

Battery charging time	Retain time (room temperature)
4 hours	> 2 days
12 hours	> 12 days
16 hours	> 15 days

#### Operating problems when the battery is discharged

- Retain data is not saved.
- RTC data is not stored and is the initial value.
- Reset button does not work (PLC Reset and Factory Reset cannot be used).

### 4.4. GL-9981-L Specifications

<b>Programming environment</b>	C/C++ (on Linux)
--------------------------------	------------------

### 4.5. GL-9981-C-1 Specifications

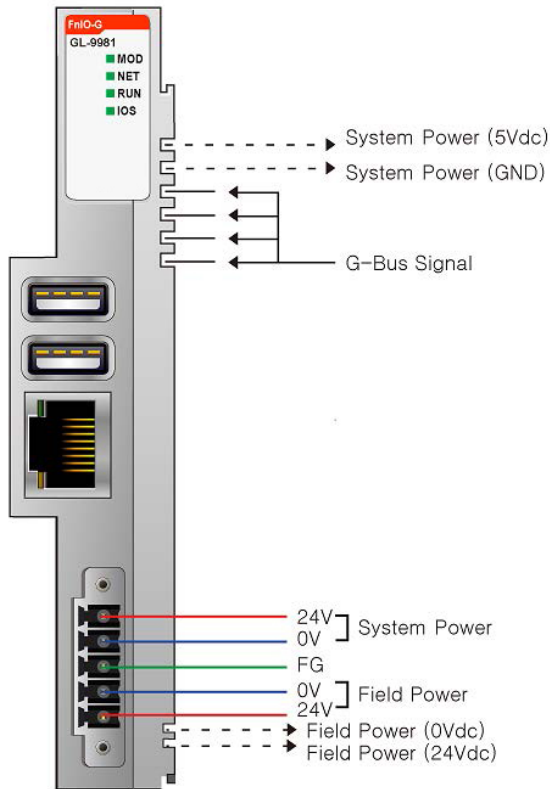
All GL-9981-C modules includes specific CODESYS capabilities based on the installed **CODESYS Application-Based Runtime License**. For instance, **GL-9981-C-1** has the **Control Basic L** license. For further details about the CODESYS Application-Based Runtime License, refer to the [CODESYS website](#).

<b>Programming environment</b>	CODESYS 3.5 SP19 Patch 4
<b>Code size</b>	3 MB
<b>Available I/O channels</b>	256
<b>Fieldbuses</b>	Modbus TCP, 2 instances
<b>Communication</b>	OPC UA server, max. 512 tags
<b>Visualization</b>	Web Visualization, max. 128 tags

## 5. Connectors and Ports

This section describes the modules connectors and ports.

### 5.1. Wiring Diagram



Pin no.	Signal description
1	System power, 24 V
2	System power, ground
3	Frame ground
4	Field power, ground
5	Field power, 24 V

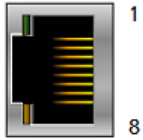


#### WARNING

Never connect system power to field power! Use separate power supplies.

### 5.2. RJ-45 Socket

For information on network installation, refer to [Network Installation](#).



RJ-45	Signal name	Description
1	TD+	Transmit +
2	TD-	Transmit -
3	RD+	Receive +
4	-	-
5	-	-
6	RD-	Receive -
7	-	-
8	-	-
Case	Shield	Shield

### 5.3. USB 2.0 Port



USB 2.0 (Type-A)	Signal name	Description
1	VCC	+5 VDC
2	D-	Data-
3	D+	Data+
4	GND	Ground



#### NOTE

250 mA continuous current per channel.

### 5.4. Monitor Port



Monitor	Signal Name	Description
1	TMDS Data2+	TMDS Lanes
2	TMDS Data2 Shield	

Monitor	Signal Name	Description
3	TMDS Data2-	
4	TMDS Data1+	
5	TMDS Data1 Shield	
6	TMDS Data1-	
7	TMDS Data0+	
8	TMDS Data0 Shield	
9	TMDS Data0-	
10	TMDS Clock+	
11	TMDS Clock Shield	
12	TMDS Clock-	
13	CEC	CEC Remote Control
14	Reserved	
15	SCL	DDC Clock
16	SDA	DDC Data
17	GND	CEC/DDC/HEAC Ground
18	+5V	Power EDID/DDC
19	HPD	Hot Plug Detect

## 6. Buttons and Switches

This section describes the buttons and switches on the module.

### 6.1. Button



Push button	Signal name	Description
Push and detach	User-defined	Operates according to the user's program.

### 6.2. Toggle Switch



Switch position	Module status	Description
Down	Stop	PLC in stop mode
Up	Run	PLC in run mode

### 6.3. DIP Switch



Dip switch	Signal name	Description
Pin 1 ON	Force USB boot mode	After connecting to PC and booting in USB Device mode, OS installation is possible.
Pin 2 ON	EEPROM write protect	Protects the onboard EEPROM that stores the boot code.



#### WARNING

Turn off the DIP switch when the module is not in use.

## 7. LED Indicator



LED	Function	Color
MOD	Module status	Green
NET	Network status	Green
RUN	Run/Stop status or Custom LED	Green/Red
IOS	Expansion module status	Green/Red

### 7.1. MOD (Module Status)

Status	LED indication	Description
Not powered	Off	Power is not supplied to the module.
Idle	Off	Module is operating normally, not accessing eMMC.
Operational	Flashing green (irregular, at bootup)	Module is operating normally, accessing eMMC.
Boot error detected	Flashing green (regular pattern, at bootup)	Error during booting. See <a href="#">Error patterns</a> below.

#### Error patterns

Long flashes	Short flashes	Status
0	3	Generic failure to boot
0	4	start*.elf not found
0	7	Kernel image not found
0	8	SDRAM failure
0	9	Insufficient SDRAM
0	10	HALT state
2	1	Partition not FAT
2	2	Failed to read from partition
2	3	Extended partition not FAT
2	4	File signature/hash mismatch
3	1	SPI EEPROM error
3	2	SPI EEPROM is write-protected

Long flashes	Short flashes	Status
3	3	I2C error
4	4	Unsupported board type
4	5	Fatal firmware error
4	6	Power failure type A
4	7	Power failure type B

## 7.2. NET (Network Status)

Status	LED indication	Description
Not powered	Off	No power supplied to the unit.
Ethernet off	Off	LAN cable is not connected, or Ethernet is inactive.
Ethernet activity	Green	LAN cable is connected and Ethernet is active.

## 7.3. RUN

The RUN LED has different functions depending on the model.

### 7.3.1. Run/Stop Status (GL-9981-C)

For model GL-9981-C, RUN shows the operating state of the PLC program (CODESYS).

Status	LED indication	Description
Run	Green	The PLC program is running.
Stop	Off	The PLC program is stopped.



#### NOTE

If the PLC program is deleted while running, the LED remains in its last state.

### 7.3.2. Custom LED (GL-9981-L)

For model GL-9981-L, RUN is controlled using the provided library.

## 7.4. IOS (Expansion Module Status)

Status	LED indication	Description
No expansion I/O	Off	No expansion modules are connected or the device is not powered.
Expansion I/O active	Green	Expansion modules are connected.

Status	LED indication	Description
Configuration fault	Red	One of the following errors occurred: <ul style="list-style-type: none"><li data-bbox="730 322 1209 353">• Invalid expansion module ID detected</li><li data-bbox="730 367 1023 398">• Initial protocol failure</li><li data-bbox="730 412 1294 479">• Vendor code mismatch between adapter and expansion module</li><li data-bbox="730 492 1246 524">• Expansion module configuration changed</li><li data-bbox="730 537 1102 568">• Too many expansion modules</li><li data-bbox="730 582 1023 613">• Communication failure</li><li data-bbox="730 627 959 658">• I/O size overflow</li></ul>

## 8. Hardware Setup



### CAUTION

Always read this section before installing the module!



### CAUTION

**Hot surface!** The surface of the housing can become hot during operation. If the module is used in high ambient temperatures, always let it cool down before touching it.



### CAUTION

Working on energized devices can damage the equipment! Always turn off the power supply before working on the module.

### 8.1. Environmental Requirements

#### Installation

- Read the [Environmental Specifications](#) carefully!
- Choose a well-ventilated area.
- Avoid enclosed spaces.
- Use a circulation fan to improve airflow, if possible.

#### Temperature Guidelines


- If the ambient temperature reaches 50°C or higher, the CPU clock frequency will be 600 MHz.
- To operate at 55°C or higher, install a fan outside the module.

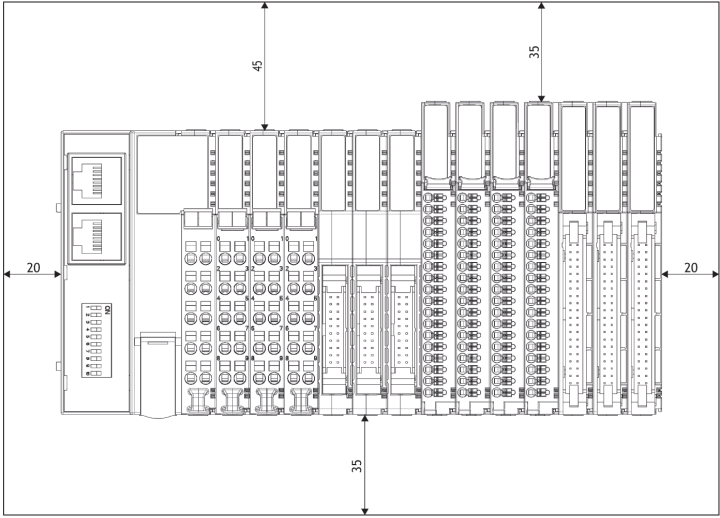
#### Monitoring and Actions

- Operating temperature may vary based on environmental conditions.
- After installation, monitor the CPU temperature based on the application.
- Take appropriate actions if the CPU temperature exceeds 85°C.

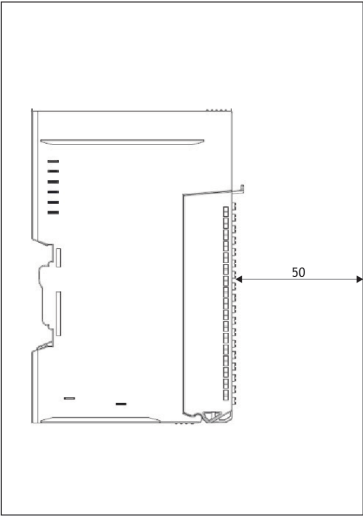
## 8.2. Space Requirements

The following drawings show the space requirements when installing the G-series modules. The spacing creates space for ventilation, and prevents conducted electromagnetic interference from influencing the operation. Installation position is valid vertical and horizontal. The drawings are illustrative and may be out of proportion.

 **CAUTION**  
Not following the space requirements may result in damaging the product.



Vertical and horizontal space requirements



Required distance to door

## 8.3. Mount Module to DIN Rail

The following chapters describe how to mount the module to the DIN rail.



### CAUTION

The module must be fixed to the DIN rail with the locking levers.

### 8.3.1. Mount GL-9XXX or GT-XXXX Module

The following instructions apply to these module types:

- GL-9XXX
- GT-1XXX
- GT-2XXX
- GT-3XXX
- GT-4XXX
- GT-5XXX
- GT-7XXX

GN-9XXX modules have three locking levers, one at the bottom and two on the side. For mounting instructions, refer to [Mount GN-9XXX Module](#).



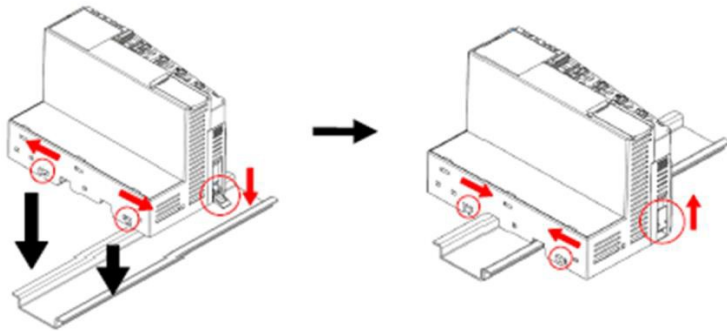
*Mount to DIN rail*



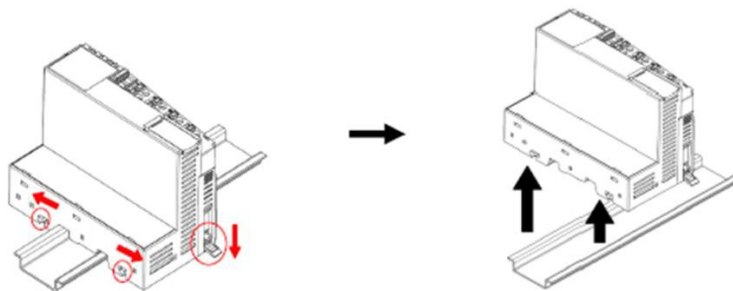
*Dismount from DIN rail*

### 8.3.2. Mount GN-9XXX Module

To mount or dismount a **network adapter** or **programmable IO** module with the product name **GN-9XXX**, for example GN-9251 or GN-9371, see the following instructions:



*Mount to DIN rail*



*Dismount from DIN rail*

## 8.4. Mount Removable Terminal Block

To mount or dismount a removable terminal block (RTB), see the instructions below.

① **Insert**



② **Lock**



*Mount a removable terminal block*

① **Unlock**



② **Pull out**



*Dismount a removable terminal block*

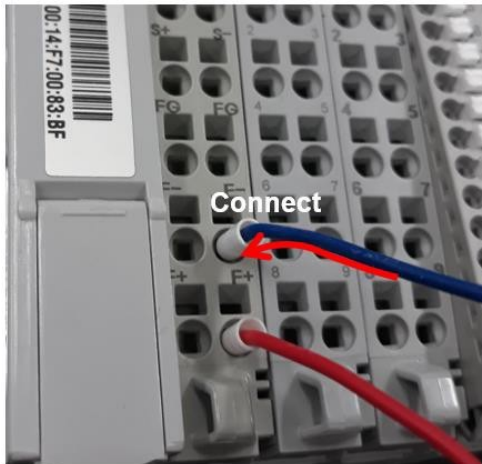
## 8.5. Connect Cables to Removable Terminal Block

To connect/disconnect cables to/from the removable terminal block (RTB), see the instructions below.

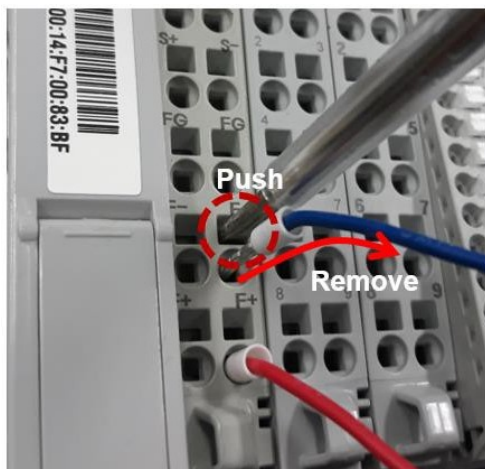


### WARNING

Always use the recommended supply voltage and frequency to prevent damage to the equipment and ensure optimal performance.



*Connect cable*



*Disconnect cable*

## 9. How to Start the Module

1. Turn on the power to boot up the device.
2. After booting is complete, the **BUS Scan process** program is automatically executed. When the scan is completed, the **IOS LED** turns green.

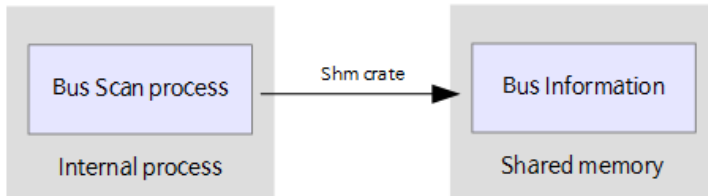
## 10. IP Settings

IP address	192.168.100.72
Subnet mask	255.255.255.0
Router	192.168.100.1

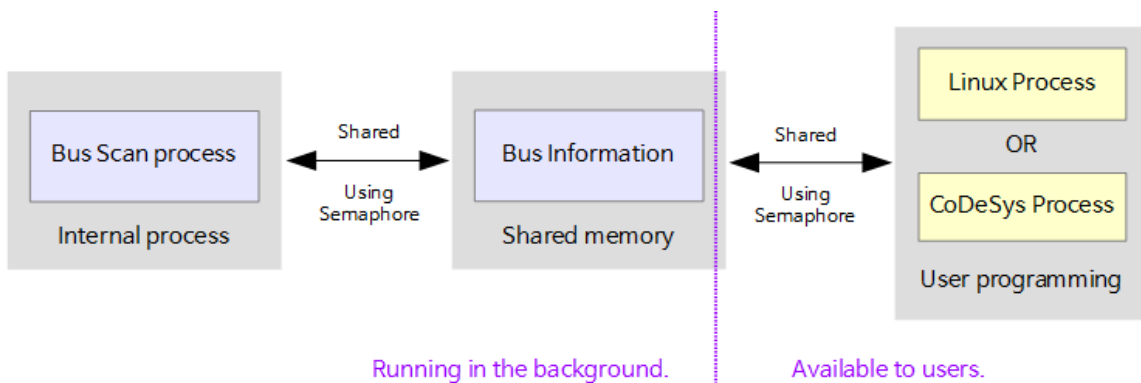
# 11. Configuration Diagrams

## 11.1. Programming Configuration Diagram

Internal process creates shared memory.



Internal processes and user programs share shared memory using semaphores.

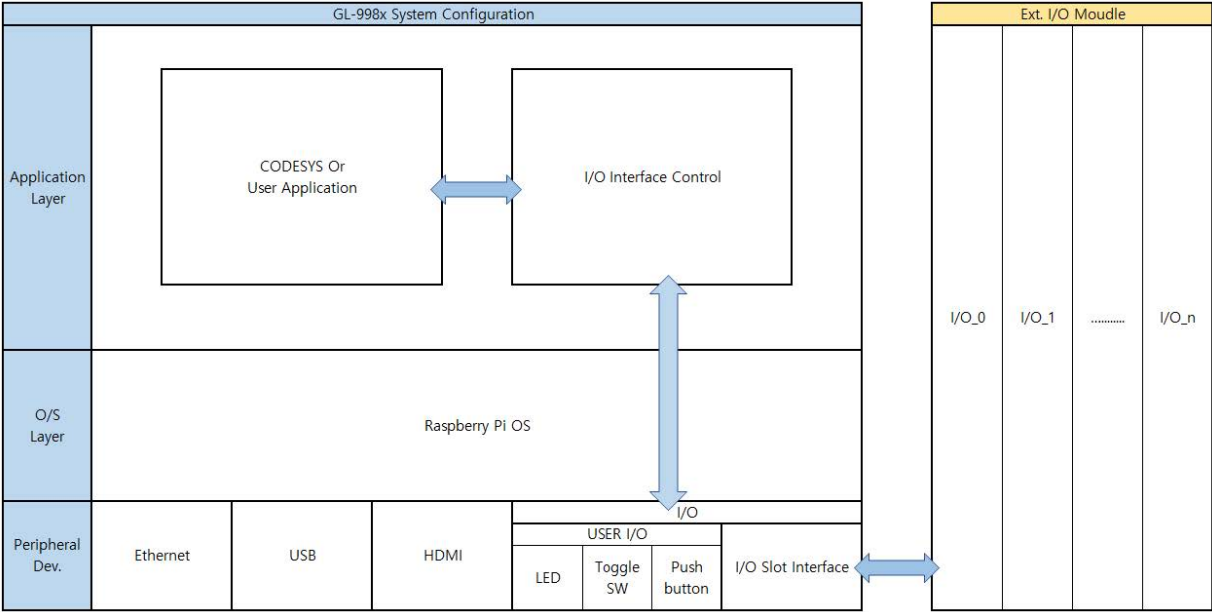


### CAUTION

If any user program is executed earlier than Bus Scan Process, it will not operate normally.

- Bus Scan Process (pibus.service) is automatically executed using SYSTEMD.
- File location of pibus.service: /lib/systemd/system/pibus.service.
- Location of actual executable file: /home/crevis/pibus/M\_PibusScan.
- Only one of the Linux Process or CODESYS process should be running (Cannot run concurrently).

# 11.2. System Configuration Diagram



## 12. CODESYS

This section describes how to use the CODESYS integrated development environment.



### TIP

Download the correct CODESYS IDE version for your target device from [SmartStore](#). See [Specifications](#) for details.

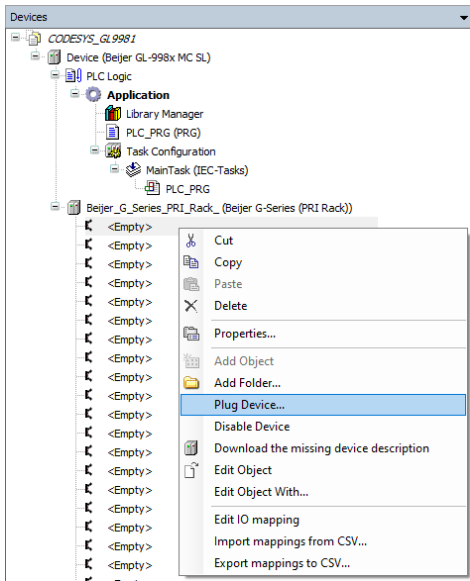
### 12.1. Download CODESYS Template Project

A CODESYS template project file is available on the Beijer Electronics website. This file contains a complete project with all necessary files for an easy startup. Follow these steps to download and open the template:

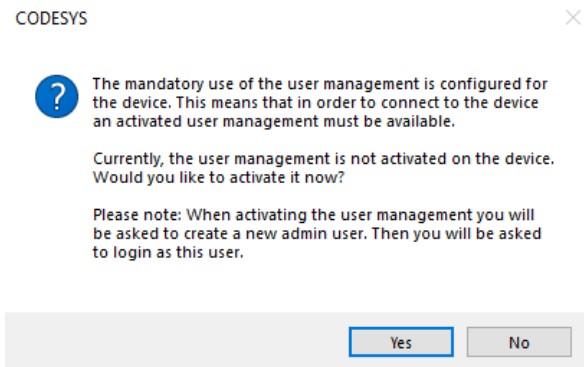
1. Download the template file from the [Beijer Electronics Website](#).
2. In CODESYS, go to **File > Project archive > Extract archive**.
3. Select the template file and extract it.

### 12.2. Create a New CODESYS Project

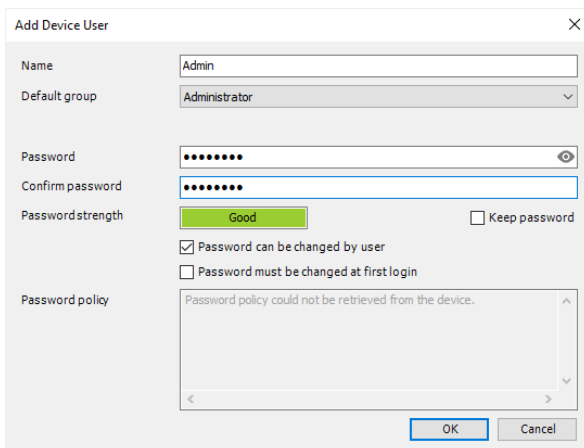
1. Open CODESYS.
2. *Optional: Install device description*  
Instead of using the **Standard project** template (see [Download CODESYS Template Project](#)) in the instructions below, you can use the **device description (XML)**. To install this, go to **Tools > Device Repository > Install**, and install the required XML files for your application.
3. Click **File > New Project**.
4. Select **Standard project**, and click **OK**.
5. Configure the following settings:
  - **Device:** Beijer GL-9981x MC SL (Beijer Electronics)
  - **PLC\_PRG in:** Structured Text (ST)
6. Click **OK**.
7. Right-click on an empty space in the rack and select **Plug Device**.



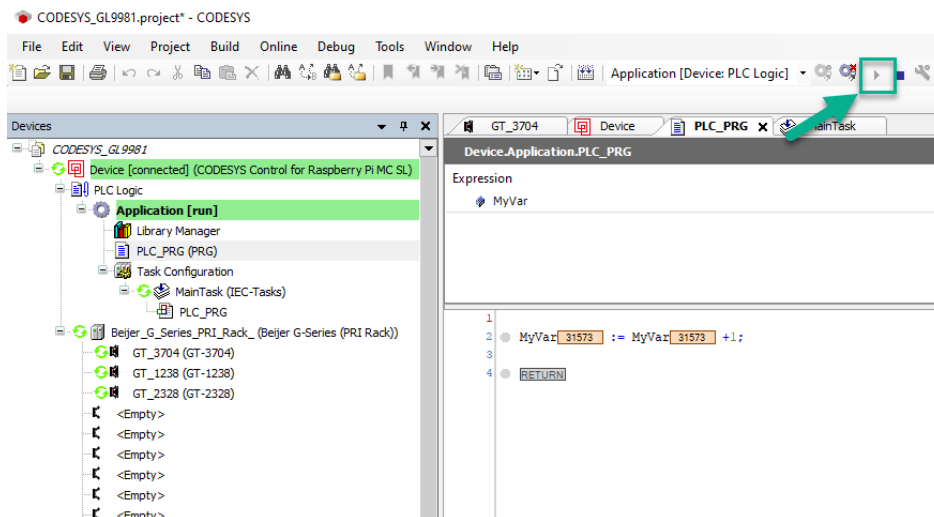
8. Choose a module from the list and click **Plug Device**.
9. Double-click **Device**, then select **Scan network**.
10. Select the desired device from the scanned devices and click **OK**.
11. Click **Yes** to activate user management on the device.



12. Enter the login credentials and click **OK**.



13. Write a test program and click **Login** in the top bar.
14. Click **Start** in the top bar to run the program.



### 12.3. Example: Activate CODESYS License

Use the **License Manager** in CODESYS to manage licenses for add-on products on a CODESYS device. These steps apply only when modifying or adding to an existing CODESYS license. In the following example a license is activated on a soft container device (*Raspberry Pi*).

1. Click **Tools > License Manager**.
2. Select **Device**, and click **Next**.
3. Select **Soft container**, and click **Next**.
4. Select the scanned *Raspberry Pi*, and click **OK**.
5. Enter login credentials and click **OK**.
6. Click **Install Licenses**.
7. Select **Activate license**, then click **Next**.
8. Enter the **Ticket ID**, and click **Next** to complete the installation.

### 12.4. Add a Module XML to CODESYS

1. Click **Tools > Device Repository**.
2. Select the module XML, and click **Open**.
3. After installation, the device will appear in the overview.

## 12.5. I/O Control Functions in CODESYS Project Example

### Folders

<b>DataTypeDef</b>	Data type definitions.
<b>Function</b>	Functions used for I/O control.
<b>global</b>	Global variable.
<b>MainProgram</b>	Example program for function testing.
<b>Struct</b>	Definitions for structures used in the internal bus.

### 12.5.1. I/O Control Functions

The following functions are available in the **Function** folder.

#### Pibus Control (pibus\_Control)

<b>pibusInit</b>	Initialize semaphores and shared memory. Confirm that the internal bus is running before proceeding. <b>VAR_INPUT</b> - None <b>VAR_OUTPUT</b> - Execution result
<b>Read_InputImage</b>	Read values from the input image. <b>VAR_INPUT</b> - Addr (starting address), pBuffer (buffer), Len (length) <b>VAR_OUTPUT</b> - Execution result
<b>Read_OutputImage</b>	Read values from the output image. <b>VAR_INPUT</b> - Addr (starting address), pBuffer (buffer), Len (length) <b>VAR_OUTPUT</b> - Execution result
<b>Write_OutputImage</b>	Write values to the output image. <b>VAR_INPUT</b> - Addr (starting address), pBuffer (buffer), Len (length) <b>VAR_OUTPUT</b> - Execution result
<b>Read_Slot (Input module only)</b>	Read the value of the specified slot. <b>VAR_INPUT</b> - Slot (slot number), pBuffer (buffer) <b>VAR_OUTPUT</b> - Execution result
<b>Write_Slot (Output module only)</b>	Write the value to the specified slot. <b>VAR_INPUT</b> - Slot (slot number), pBuffer (buffer) <b>VAR_OUTPUT</b> - Execution result
<b>Get_Prm</b>	Read the parameters of the specified slot. <b>VAR_INPUT</b> - Slot (slot number), pBuffer (buffer) <b>VAR_OUTPUT</b> - Execution result
<b>Set_Prm</b>	Write the parameters to the specified slot. <b>VAR_INPUT</b> - Slot (slot number), pBuffer (buffer) <b>VAR_OUTPUT</b> - Execution result

## Semaphore Control (sem\_Control)

<b>semInit</b>	Initialize the semaphore. <b>VAR_OUTPUT</b> - Execution result
<b>semEnter</b>	Use the semaphore.
<b>semLeave</b>	Release the semaphore.

## Shared Memory Control (shm\_Control)

<b>shmInit</b>	Initialize shared memory. <b>VAR_OUTPUT</b> - Execution result
<b>shmRead_X</b>	Read shared memory to a buffer. <b>VAR_OUTPUT</b> - Execution result
<b>shmWrite_X</b>	Write buffer data to shared memory. <b>VAR_OUTPUT</b> - Execution result

### 12.5.2. Function Test Example

This example demonstrates manually selecting and executing a test after initialization.

#### Example

Device.Application.pibus_function_test		
Expression	Type	Value
init	INT	255
test_selet	INT	0
test_time	INT	0
slot	BYTE	0
io_rw_byte	ARRAY [0..255] OF u8	
addr	WORD	0
len	BYTE	0

- **init**: If initialization is successful, the **init** variable is set to **255**. Testing is only possible when initialization is completed successfully.
- **test\_selet**: Select the desired test with the **test\_selet** variable.
- **test\_time**: Once the test is executed, the **test\_time** variable is set to **1** to prevent repetition of the test.
- **slot**: Set variable for the slot number.
- **io\_rw\_byte**: The value to be written is stored in advance in the **io\_rw\_byte** buffer.
- **addr**: Set variable for the starting address.
- **len**: Set variable for the length.



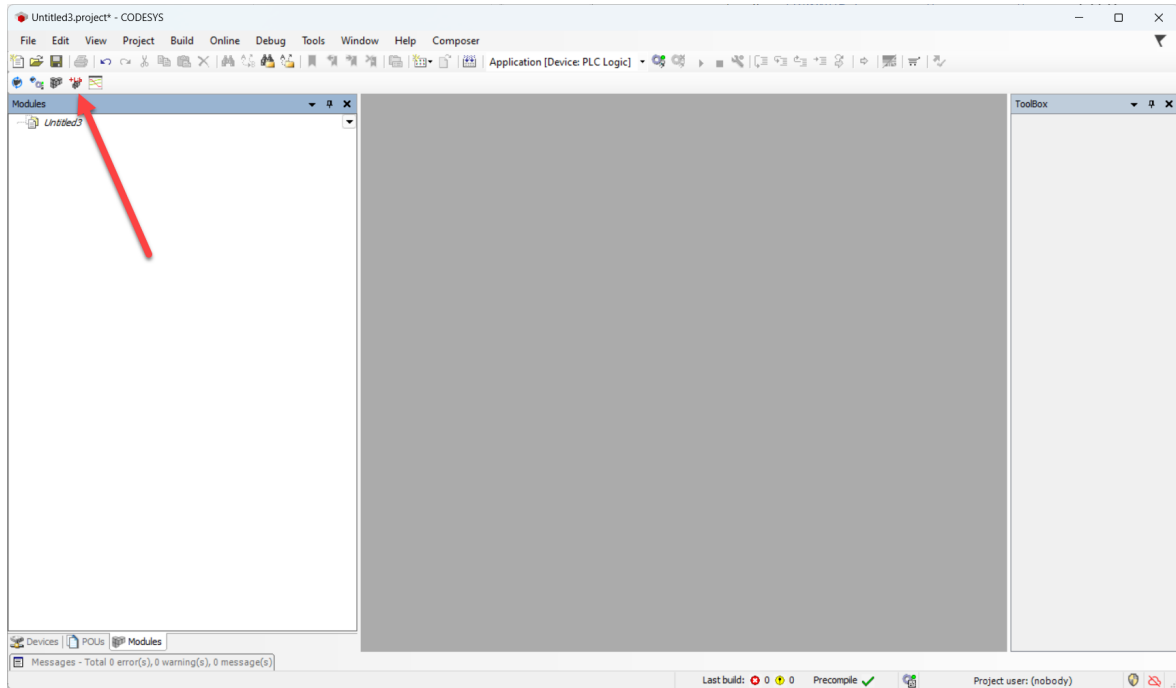
#### TIP

Refer to the **Pibus\_GL9981\_Sample** project for the complete program example.

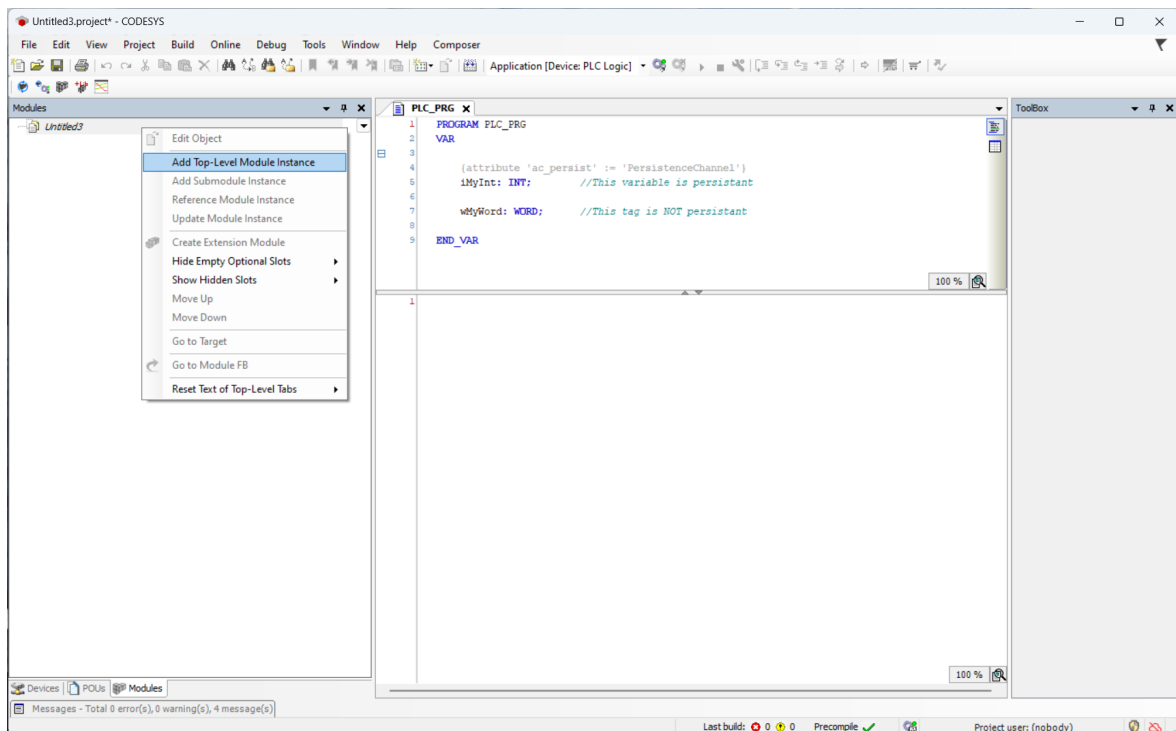
## 12.6. Add the Persistence Manager

The **Persistence Manager** is a standard component of the **Application Composer**. It manages persistent remanent data by recognizing specific attributes in variable declarations. For more details, visit the [CODESYS website](#).

1. In CODESYS, click **View > Modules**.
2. Click **Add module library**.

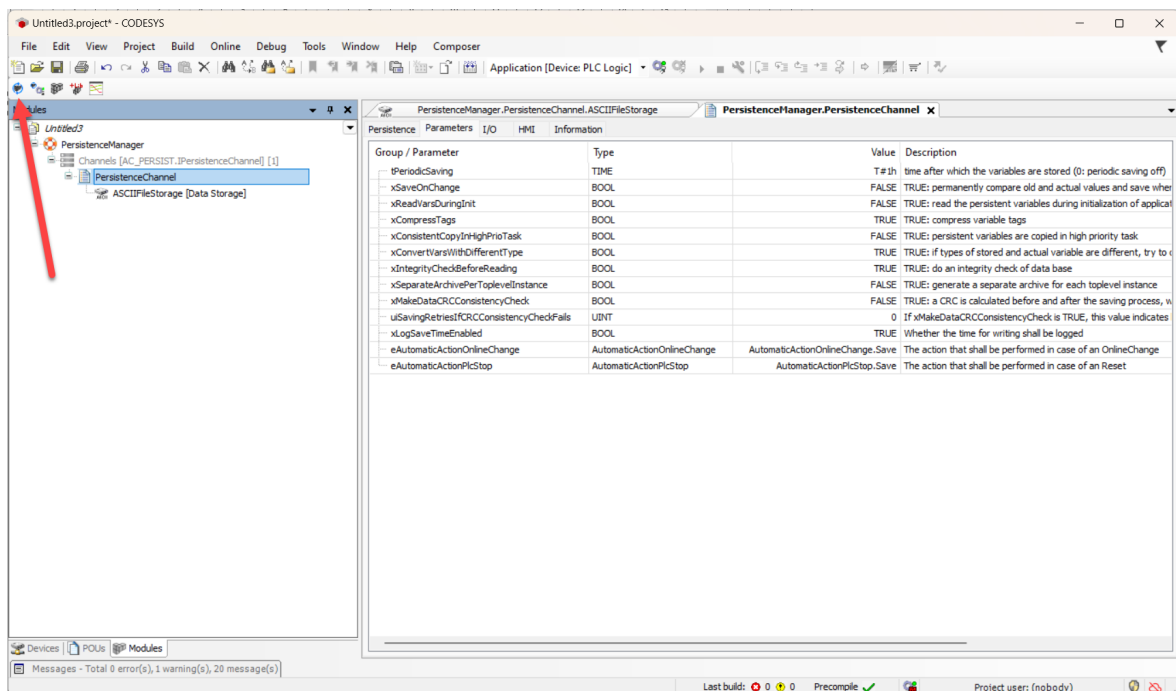


3. Select **AC\_Persistence** and click **OK**.
4. Right click on project name and select **Add Top-Level Module Instance**.



5. Select **PersistenceManager** and click **Add Module**.

6. Right click on **Channels** and select **Add SubModule Instance**.
7. Choose **PersistentChannel** and click **Add Module**.
8. Right click **Data Storage** and select **Add SubModule Instance**.
9. Choose one of the following storage formats, then click **Add Module**:
  - **ASCIIFileStorage** - Saves the values in ASCII format in a file. The **ASCIIFileStorage** format allows to open the archive file in a text editor and to modify the values. Comments can be added to ASCII file archive. These will be ignored when the file is read.
  - **BinaryFileStorage** - Saves the values in binary format in a file.
  - **BinaryMemoryStorage** - Saves the values in binary format to the *RETAIN* area of the compiler, or the device concerned, for example, to the NVRAM of the device.
10. Click on **PersistenceChannel** and **select Parameters**.
11. Ensure that the `xReadVarsDuringInit` parameter is set to FALSE.
  - If TRUE, persistent variables are loaded during initialization.
  - If FALSE, they are loaded during the first application cycle.
12. Click **Generate the project from the module tree**.



13. CODESYS generates new function blocks and tasks for persistence handling.
14. In the **Global variable list** or **POU**, add an attribute before the tags that need to be persistent. All attributes have a reference to the file for the tags, marked in **bold** in the example below.

### Example

```
PROGRAM PLC_PRG
VAR
    {attribute 'ac_persist' := 'PersistenceChannel1'}
    iMyInt: INT;           //This variable is persistent
    wMyWord: WORD;       //This tag is NOT persistent
END_VAR
```

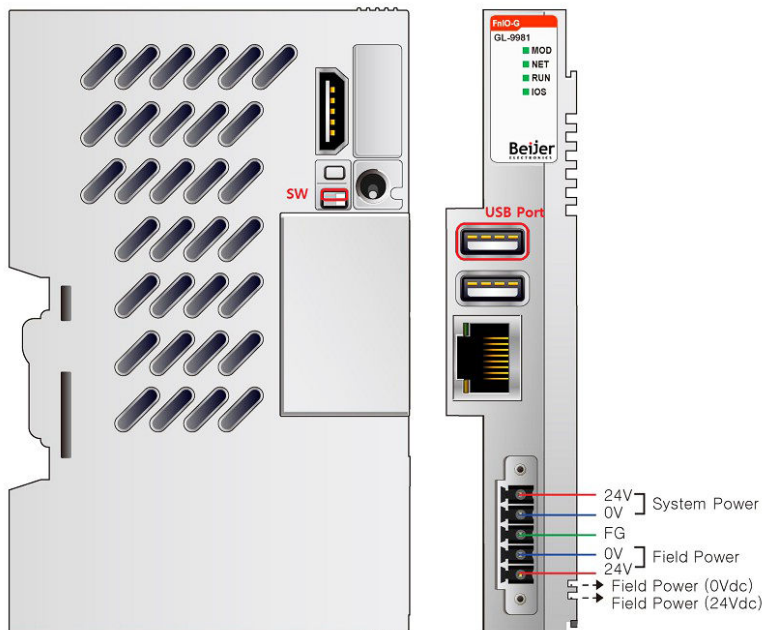
## 13. System Setup (GL-9981-L)

This chapter describes how to set up the GL-9981-L (Linux version). It covers operating system installation, Docker and WebIQ setup, CODESYS Runtime installation, and I/O control using C language.

### 13.1. Install the OS

Follow the steps below to install the OS.

1. On the **dip switch (SW)**, set **Pin 1** to **ON**.



2. Connect the module to the PC with USB.  
**Cables**
  - GL-9981-X-X: USB A(M) to A(M) cable
  - GL-9982-X-X: USB C(M) to A(M) cable



#### NOTE

Ensure that the cables are authentic and that all data lines are properly connected.

3. Connect the module to the **24 VDC system power**.
4. Use **rpi-boot** program to recognize Raspberry Pi as a USB device.



#### TIP

Download **rpi-boot** from the [Raspberry Pi website](#).

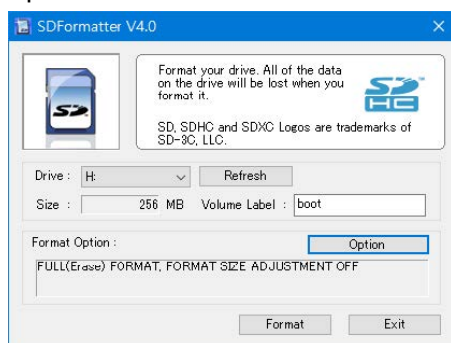
```

rpiboot
RPiBOOT: build-date Jul 18 2022 version 20220718-085937 5a25e04b
Waiting for BCM2835/B/7/2711...
Loading embedded: bootcode4.bin
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/B/7/2711...
Loading embedded: bootcode4.bin
Loading embedded: bootcode4.bin
Second stage boot server
Loading embedded: start4.elf
File read: start4.elf
Second stage boot server done

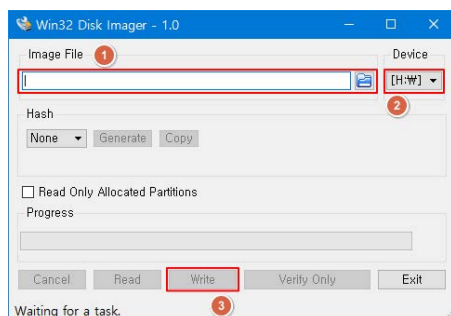
```

5. If the OS is intended to be saved as an image file, proceed to **step 6**. If not, follow these instructions:

- a. Download and install **SD Formatter** from the [SD Association website](#).
- b. Open **SD Formatter**.



- c. Choose **Drive**.
  - d. Click **Option**, select **FORMAT TYPE - FULL**, and click **Format** to format the drive.
6. Download and install **Win32 Disk Imager** from the [Win32 Disk Imager website](#).
7. Open **Win32 Disk Imager**.
1. Select **OS Image file**.
  2. Select **Device**.
  3. Click **Write**. To save the OS in the module as an image file, click **Read**.



## 13.2. Install Docker

### Prerequisites

- The hardware must be connected to the internet during the whole process. All dependencies and installation files are downloaded during this session from the Docker hub.

### Instructions

- Open a console (e.g. Command Prompt or Windows PowerShell).
- Connect to the module with SSH:

```
ssh beijer@192.168.100.72
```

Default IP	192.168.100.72
Username	beijer
Password	Jzy!q7

- Change the IP address:
  - Open the configuration file in nano:
 

```
nano /etc/dhcpd.conf
```
  - In the interface eth0 section, enter your own IP address and router (gateway) address.  
*Example:* 192.168.1.210 (IP) and 192.168.1.1 (router).
  - Press **CTRL+X** to exit.

```

GNU nano 5.4 /etc/dhcpd.conf
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

interface eth0
static netmask=255.255.255.0
static routers=192.168.1.1
static ip_address=192.168.1.210
  
```

- Press **Y** to confirm.

```

beijer@raspberrypi: ~
GNU nano 5.4 /etc/dhcpd.conf *
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

interface eth0
static netmask=255.255.255.0
static routers=192.168.1.1
static ip_address=192.168.1.210

Save modified buffer?
Y Yes
N No  ^C Cancel

```

e. Press **Enter** to save.

f. Reboot the hardware:

```
sudo reboot
```

4. Run the following command to uninstall old or conflicting packages:

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

5. Set up Docker's repository:

a. Add Docker's official GPG key:

```

sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/raspbian/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

```

b. Set up Docker's APT repository:

```

echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/raspbian \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

```

6. Install the latest Docker version:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

```

beijer@raspberrypi: ~
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Get:3 https://download.docker.com/linux/raspbian bullseye InRelease [26.6 kB]
Get:4 https://download.docker.com/linux/raspbian bullseye/stable armhf Packages [30.4 kB]
Fetched 57.1 kB in 3s (19.8 kB/s)
Reading package lists... Done
beijer@raspberrypi:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-pl
ugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  bluez-firmware
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  apparmor dbus dbus-user-session docker-ce-rootless-extras git git-man iptables libdbus-1-3 liberror-perl libip6tc2
  libltdl7 libnetfilter-contrack3 libnftnl0 libslirp0 slirp4netns
Suggested packages:
  apparmor-profiles-extra apparmor-utils cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc
  git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn firewalld
The following NEW packages will be installed:
  apparmor containerd.io dbus-user-session docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin git git-man iptables liberror-perl libip6tc2 libltdl7 libnetfilter-contrack3 libnftnl0
  libslirp0 slirp4netns
The following packages will be upgraded:
  dbus libdbus-1-3
2 upgraded, 18 newly installed, 0 to remove and 166 not upgraded.
Need to get 101 MB of archives.
After this operation, 386 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

7. Press Y to confirm.
8. To test the Docker engine, run the Docker test container:

```
sudo docker run hello-world
```

```

beijer@raspberrypi: ~
See 'docker --help'
beijer@raspberrypi:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2d3d56bba6ab: Pull complete
Digest: sha256:4bd7811b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (arm32v7)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
beijer@raspberrypi:~$

```

If successful, Docker is installed and running.

### 13.2.1. Install the WebIQ Docker Container

#### Prerequisites

- Install and verify the Docker engine before you install the WebIQ Docker container. See [Install Docker](#).
- The hardware must be connected to the internet during the whole process. All dependencies and installation files are downloaded during this session from the Docker hub.

#### Instructions

1. Open a console (e.g. Command Prompt or Windows PowerShell).

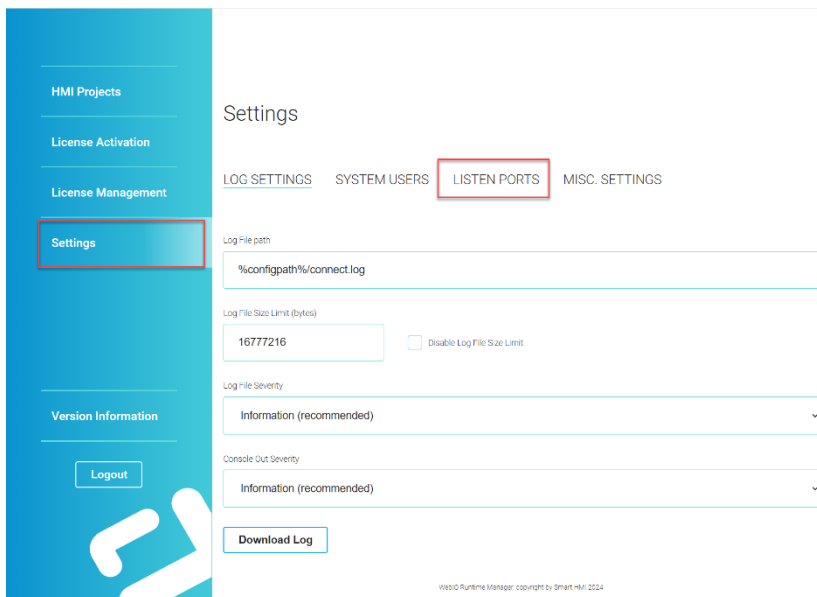
## 2. Connect to the module with SSH:

```
ssh beijer@192.168.100.72
```

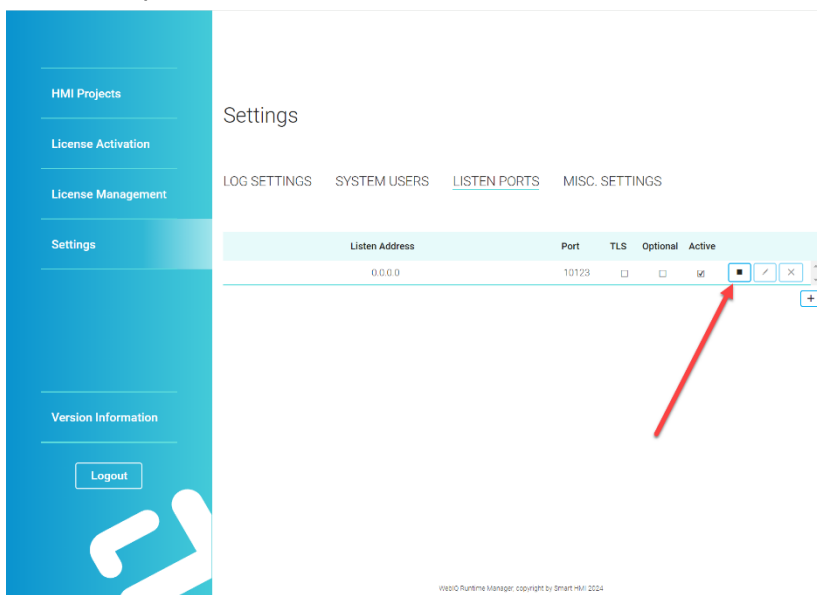
<b>Default IP</b>	192.168.100.72
<b>Username</b>	beijer
<b>Password</b>	Jzy!q7

3. If you use a prototype, disable port **10123** to avoid conflicts:

- a. Log in to the WebIQ portal with <IP:10123>.
- b. Create a user and a password.
- c. Go to **Settings** > **LISTEN PORT**.



## c. Deactivate port 10123 and confirm.



4. Download **WebIQ Server App** from [Beijer Electronics Smart Store](#).
5. Change the file extension from .bapp to .zip.

*Example:* webiq-2.16.3.zip.

6. Extract the ZIP file.
7. Open the app folder.
8. Copy compose.yaml to a USB drive.
9. Insert the USB drive in **USB 1** (top port) on the GL-998X.
10. To verify the operating system, run:

Uname -a



#### NOTE

This step is not required from WebIQ version 2.16.4, since containers are available in all supported formats.

```
beijer@raspberrypi: ~  
Microsoft Windows [Version 10.0.19045.3930]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\MBE>ssh beijer@192.168.1.210  
beijer@192.168.1.210's password:  
Linux raspberrypi 5.10.63-v7l+ #1459 SMP Wed Oct 6 16:41:57 BST 2021 armv7l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sun Feb 18 17:47:30 2024 from 192.168.1.35  
beijer@raspberrypi:~$ uname -a  
Linux raspberrypi 5.10.63-v7l+ #1459 SMP Wed Oct 6 16:41:57 BST 2021 armv7l GNU/Linux  
beijer@raspberrypi:~$
```

11. Read the status and names of connected drives:

sudo fdisk -l

```

beijer@raspberrypi: ~
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes

Disk /dev/mmcblk0: 14.56 GiB, 15634268160 bytes, 30535680 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xcb9d8e30

Device      Boot Start      End  Sectors  Size Id Type
/dev/mmcblk0p1    8192   532479   524288  256M c W95 FAT32 (LBA)
/dev/mmcblk0p2   532480 30535679 30003200 14.3G 83 Linux

Disk /dev/sdb: 15.02 GiB, 16131293184 bytes, 31506432 sectors
Disk model: USB Flash Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x412fd2d7

Device      Boot Start      End  Sectors  Size Id Type
/dev/sdb1   *    2048 31506431 31504384  15G c W95 FAT32 (LBA)
beijer@raspberrypi: ~ $

```

12. Create a mount folder (example: USB1):

```
sudo mkdir /media/USB1
```

13. Mount the USB drive:

```
sudo mount /dev/sdb1 /media/USB1/
```

14. Install the WebIQ server from the USB drive:

```
sudo docker compose -f /media/USB1/compose.yaml up -d
```

15. Verify the Docker installation:

```
sudo docker ps
```

16. Check Docker status for all installed containers:

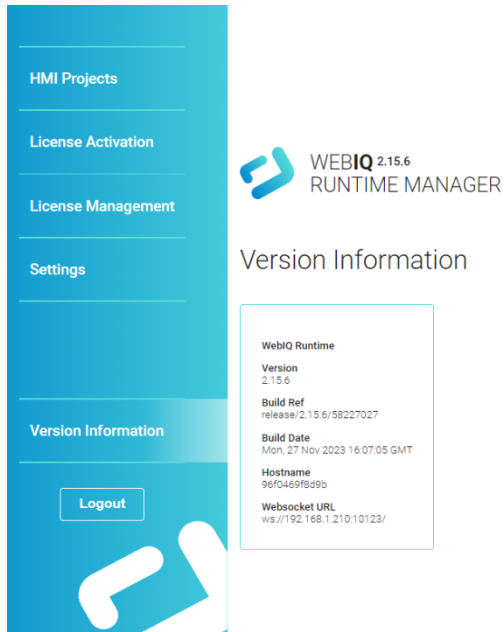
```
sudo docker stats
```

```

beijer@raspberrypi: ~
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O   PIDS
cbe7dd5ca4b2  usb1-nginx-1       5.75%    0B / 0B              0.00%    13.5MB / 4.37MB  69.6kB / 0B   3
96f0469f8d9b  usb1-connect-1     0.59%    0B / 0B              0.00%    2.61MB / 12.2MB  0B / 0B       17
6999f53de158  usb1-licenseAgent-1 0.00%    0B / 0B              0.00%    134kB / 48.5kB   65.5kB / 0B   12

```

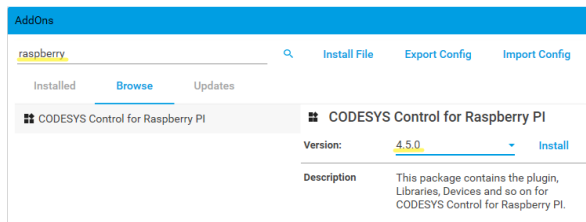
17. Install the runtime licenses.



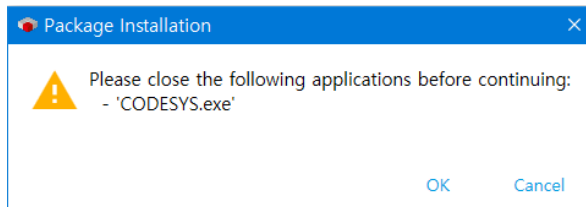
### 13.3. Install CODESYS Runtime

Follow these steps to install CODESYS Runtime on a Raspberry Pi.

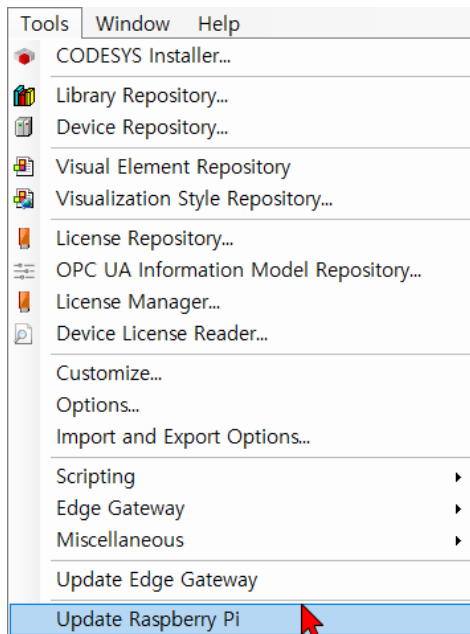
1. Open CODESYS on the PC that will connect to the Raspberry Pi.
2. Click **Tools > CODESYS Installer**.
3. In the **AddOns** section, click **Browse**.
4. In the search field, enter **raspberry**.
5. Select **CODESYS Control for Raspberry Pi**, choose version, and click **Install**.



6. Accept the license agreement and click **Continue**.
7. Close CODESYS and click **OK**.



8. Restart CODESYS and click **Tools > Update Raspberry Pi**.

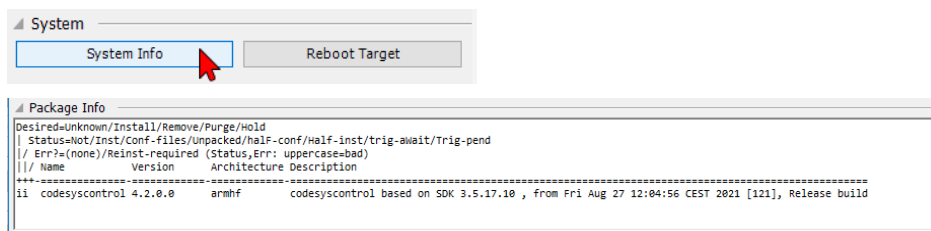


9. Enter the **login credentials** and the **IP address** of the target device. The Raspberry Pi and the PC must be on the same subnet.

### Default credentials and IP

<b>Username</b>	beijer
<b>Password</b>	Jzy!q7
<b>IP Address</b>	192.168.100.72

10. Select the installed **CODESYS Runtime Package** version, then click **Install**.
11. Select **Multicore**, then click **OK**.
12. Click **System Info** to verify the installation in the **Package Info** section.



## 13.4. Delete CODESYS Runtime

Follow these steps to remove **CODESYS Runtime** from the Raspberry Pi:

1. Run **CODESYS** and click **Tools > Update Raspberry Pi**.
2. Enter the **login credentials** and the **IP address** of the target device. The Raspberry Pi and the PC must be on the same subnet.

### Default credentials and IP

<b>Username</b>	beijer
<b>Password</b>	Jzy!q7
<b>IP Address</b>	192.168.100.72

3. Select the installed **CODESYS Runtime Package** version.
4. Click **Remove** to uninstall the CODESYS Runtime.

## 13.5. Login Credentials

Username	beijer
Password	Jzy!q7
Root Password	43dZM#hQDM

## 13.6. Controll I/O with C Language

The following files are used to control I/O in **C Language** for the module:

### Location of the files

- /home/crevis/GL-9981-example\_rev100

### Files and their functions

- **PibusloTest.c**: Example program to test the module, including operation examples for RTC, LED, SW, and I/O.
- **PibusloTest.h**: Contains definitions of pointers and global variables used for I/O control.
- **PibusStruct.h**: Defines structures used in the internal bus.



### TROUBLESHOOTING

If the program does not close properly, **restart the device**.

## 13.7. Functions Used in PibusloTest.c

Function	Description
<b>int pibusInit(void)</b>	Initialize the internal bus <ul style="list-style-type: none"> <li>• Initialize the semaphore to use</li> <li>• Initialize shared memory</li> </ul> Return(-1) - Initialization failed
<b>int Read_InputImage(u16 Addr, u8* pBuffer, u16 Len)</b>	Read values from input image <ul style="list-style-type: none"> <li>• addr - starting address / pBuffer - use buffer / Len - length</li> <li>• Return(-1) - Abnormal Addr and Len values</li> </ul>
<b>int Read_OutputImage(u16 Addr, u8* pBuffer, u16 Len)</b>	Read values from output image <ul style="list-style-type: none"> <li>• addr - starting address / pBuffer - use buffer / Len - length</li> <li>• Return(-1) - Abnormal Addr and Len values</li> </ul>
<b>int Write_OutputImage(u16 Addr, u8* pBuffer, u16 Len)</b>	Write values from output image <ul style="list-style-type: none"> <li>• addr - starting address / pBuffer - use buffer / Len - length</li> <li>• Return(-1) - Abnormal Addr and Len values</li> </ul>

Function	Description
<code>void Read_Slot(u8 slot, u8* pBuffer)</code> (Input module only)	Read the value of the slot • slot - slot number / pBuffer - use buffer
<code>void Write_Slot(u8 slot, u8* pBuffer)</code> (Output module only)	Writes the value of the slot • slot - slot number / pBuffer - use buffer
<code>void Get_Prm(u8 slot, u8* pBuffer)</code>	Read the parameters of the slot • slot - slot number / pBuffer - use buffer
<code>void Set_Prm(u8 slot, u8* pBuffer)</code>	Write the parameters of the slot • slot - slot number / pBuffer - use buffer
<code>void Get_Rtc(u8* pBuffer)</code>	Get the time stored in the RTC • pBuffer - use buffer
<code>void Set_Rtc(void)</code>	Store system time in RTC.
<code>void Read_Sw_push(u8* pBuffer)</code>	Read the state value of SW(Push) • pBuffer - use buffer
<code>void Read_Sw_toggle(u8* pBuffer)</code>	Read the state value of SW(Toggle) • pBuffer - use buffer
<code>void Read_Led(u8* pBuffer)</code>	Read the state value of the LED • pBuffer - use buffer
<code>void Write_Led(u8* pBuffer)</code>	Change the state of the LED • pBuffer - use buffer • 0: off, 1: Green, 2: Red

## 13.8. Compile PibusloTest.c in the Terminal

You can compile the PibusloTest.c example directly in the Raspberry Pi terminal using the following command:

```
crevis@raspberrypi:~/GL-9981-example_rev100 $ gcc -o PibusloTest PibusloTest.c -lpthread -lrt
```

### Example: Reading and Writing Connected I/O

When you run the compiled program, it displays the connected I/O modules, their input and output values, and parameter data.

#### Sample output:

```
***** IO TEST *****
```

```
[1] GT-1238
In: ff
Out:
Get Prm: 55 aa
```

```
=====
```

```
[2] GT-2628
In:
Out: ff
Get Prm: 00 00
```

=====

[3] GT-2764  
In:  
Out: ff  
Get Prm: 05 05

=====

[4] GT-3118  
In: ff Of ff Of fd Of ff Of ff Of ff Of ff Of ff Of  
Out:  
Get Prm: ff aa 00 00 00 00 00 00 9a

=====

[5] GT-4118  
In:  
Out: ff Of ff Of ff Of ff Of ff Of ff Of ff Of ff Of  
Get Prm: 00 00 00 00

=====

[6] GT-2764  
In:  
Out: ff  
Get Prm: 00 02

=====

[7] GT-22ba  
In:  
Out: ff ff ff ff  
Get Prm: 00 00 00 00 00 00 00 00

=====

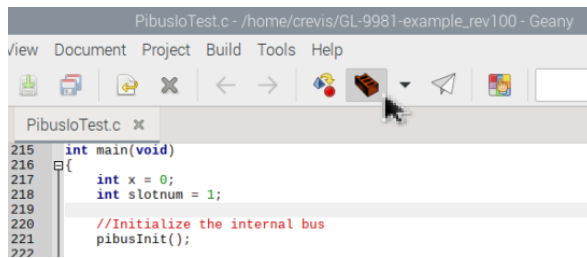
### 13.9. Use the Geany Editor on Raspberry Pi

Follow these steps to use the Geany editor on a Raspberry Pi.

1. Connect a **monitor, mouse, and keyboard** to the Raspberry Pi.
2. Turn on the power and wait for the system to boot.
3. Click **Programming > Geany Programmer`s Editor**.
4. Click **Build > Set Build Commands**.
5. Add `-lpthread -lrt` at the end of the build command.



6. Open the example program.
7. Click the **Build** icon to compile the program.



The screenshot shows the Geany IDE interface. The title bar reads "PibusloTest.c - /home/crevis/GL-9981-example\_rev100 - Geany". The menu bar includes "View", "Document", "Project", "Build", "Tools", and "Help". The toolbar contains icons for file operations and a prominent "Build" icon (a brick with a lightning bolt) which is being clicked by the mouse. The editor window shows the following C code:

```
215 int main(void)
216 {
217     int x = 0;
218     int slotnum = 1;
219
220     //Initialize the internal bus
221     pibusInit();
222 }
```

8. Once the build is complete, an **executable file** is created, and a confirmation message appears.

