



Master in Electrical and Computer Engineering

Introduction to CODESYS
and
Modbus TCP configuration

1. Introduction

CODESYS is a commercial soft PLC solution with wide acceptance in the marketplace. It provides a development environment for IEC 61131-3 programs, as well as runtime execution environments for many execution devices (from embedded platforms like the Raspberry Pi, or Wago IPCs, to personal computers) that execute the compiled IEC 61131-3 programs.

2. Installation

Download and install the “CODESYS Development System V3” on a Windows compatible operating system. You can freely download the installation package directly from either of the following locations:

- <ftp://labs-automacao.fe.up.pt/I005/Demos%20Software/CODESYS/>
- CODESYS.com
(search for “CODESYS Development System V3” in the “CODESYS store”)
(requires web site registration)

NOTE: To access labs-automacao.fe.up.pt you must be on FEUPNET. If necessary, connect to this network using VPN

NOTE: Current windows OS does not support FTP directly in the browser. To access labs-automacao.fe.up.pt open one Window Explorer window, copy the whole address line (including “ftp://”) and paste it in the address line in the window header.

NOTE: We added a new annex at the end of this document with more detailed installation instructions.

NOTE: In order to ease porting the PLC programs you will be writing between your computer and those of the lab I004, it is advisable to install the exact same version that is installed on the lab I004 bench computers. To do this it is advisable to download the software package from the lab's ftp server.

This assignment will require the use of a Raspberry PI as a Modbus master. For RaspberryPi support download and install the following packages (also available from the lab's ftp server):

- “CODESYS Edge Gateway for Linux”
- “CODESYS Control for RaspberryPi”

Start off by installing CODESYS. To install the above packages (in the above order) first run the “CODESYS Development System” program you have just installed. Go to the menu “Tools → Package Manager” and install the “CODESYS Edge Gateway for Linux” and “CODESYS Control for RaspberryPi” packages (in this order). To complete installation close the CODESYS program.

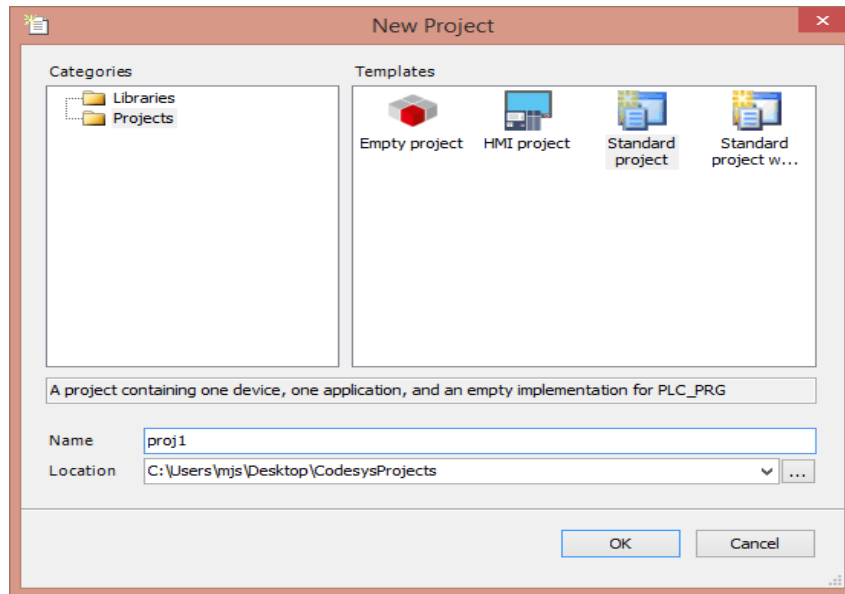
3. Configuring the First Project

3.1 Create a New Project

Run the CODESYS v3 development environment (i.e. the “CODESYS Development System V3”

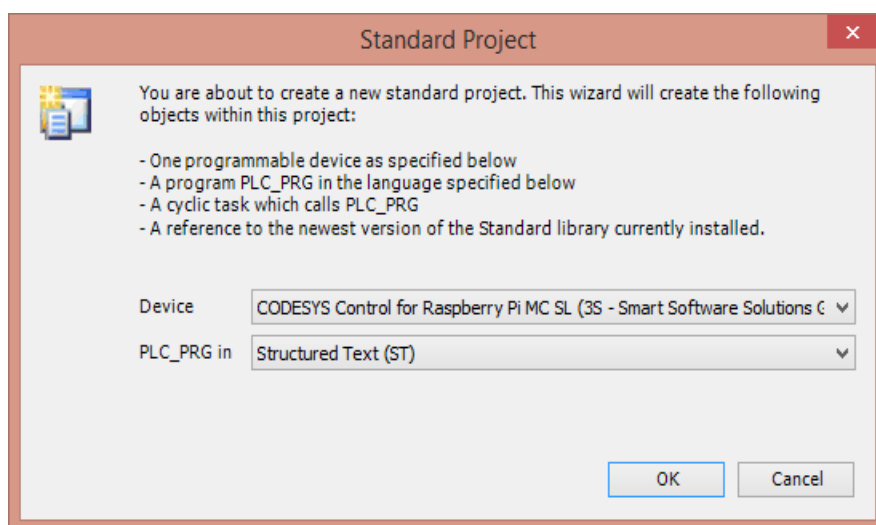
program) and create a new project using the “**Standard project**” template. This will create a project with one default IEC 61131-3 Program (a POU – Program Organization Unit), as well as a configuration with a single task that runs this program. Don’t worry, you can change all this later if required.

You should also preferably create a directory for storing this project as CODESYS will eventually create several files. Using a specific directory will help keep your files organized.



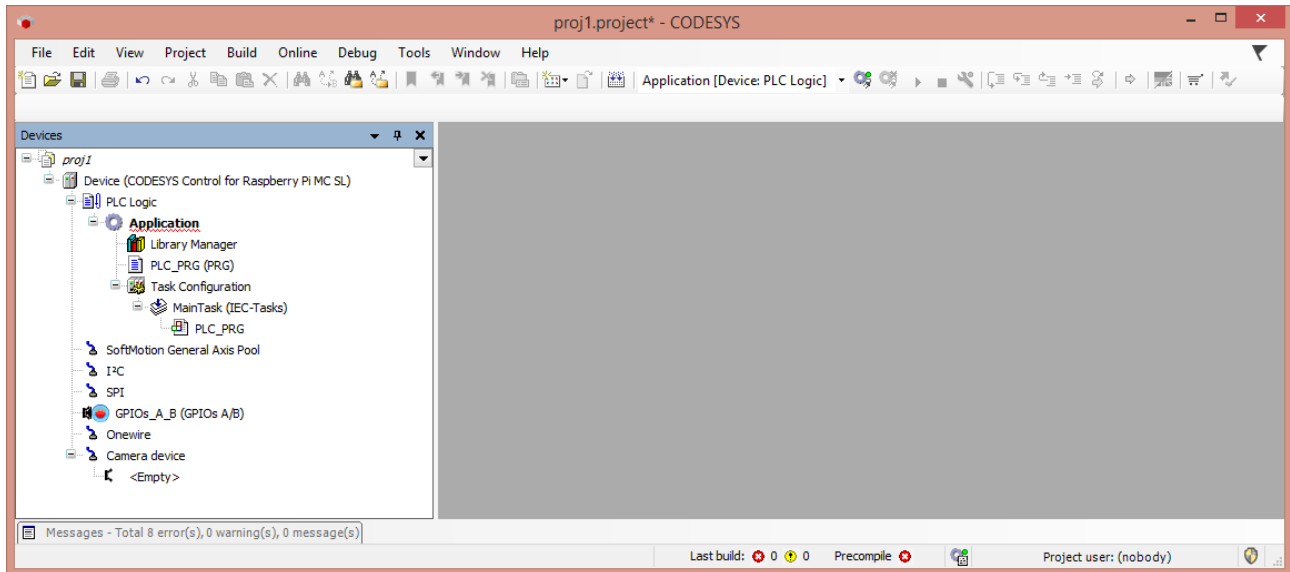
During project creation you will be asked to choose the computing device (computer) that will be running the PLC programs, as well as the programming language for the Program (an IEC 61131-3 POU) that is going to be automatically created.

- Device → “Control for Raspberry Pi MC SL
- PLC_PRG in → “Structured Text (ST)”



You will now see the newly created project in the CODESYS IDE (Integrated Development Environment). The project browser on the left (with the title “**Devices**”, and sometimes also referred to as the “device tree” or “device browser”) gives you an overview of the whole project. By double clicking on (some of) these entries you will be shown an editor on the right-hand side, in which you

will be able to program or configure the respective entry.



You can now compile your program and execute this yet empty program on a runtime.

To compile, use the menu option “Build→Build” or the shortcut key F11.

3.2 Download and Execute the Project

To execute the compiled program, you will first need to connect the Raspberry Pi (with raspbian and CODESYS RaspberryPi runtime virtual machine already installed) to the same IP network as the computer running the CODESYS IDE.

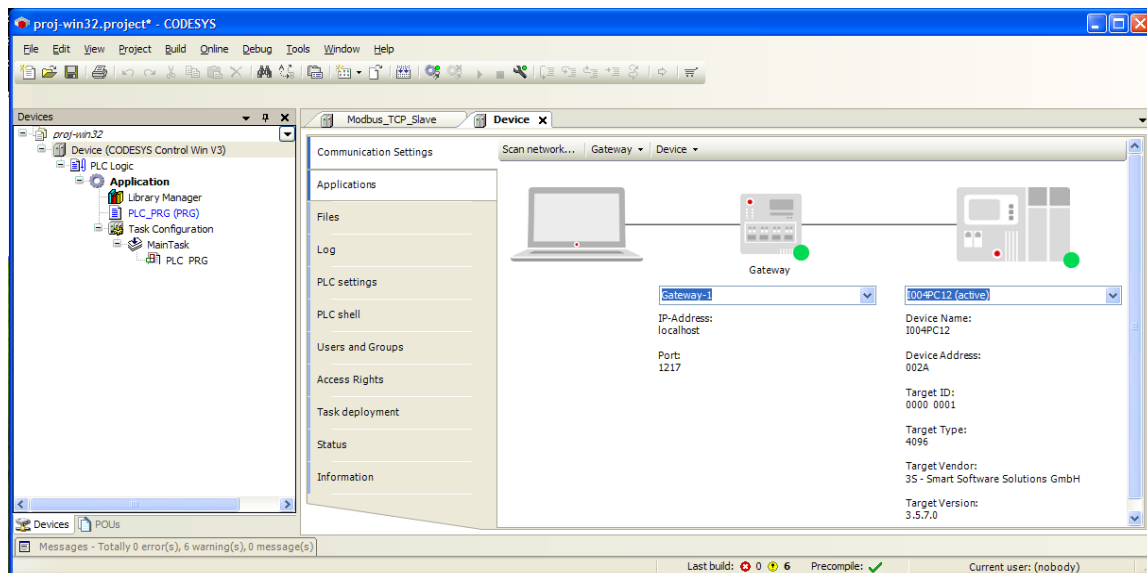
You can now configure how the CODESYS IDE will communicate with the CODESYS runtime virtual machine on the RaspberryPi. Double-click the ‘**Device (CODESYS xxxxxxxxx)**’ on the project explorer window, and configure the “Communication settings”.

Notice that the CODESYS IDE accesses physical devices (e.g. RaspberryPi) through an intermediary device called Gateway. By default your computer (with CODESYS installed) will already be running a gateway (configured by default as “Gateway1”). Using this default gateway, you will be able to access any physical devices (e.g. RaspberryPi) located on the same sub-network as your computer.

Over ‘device name’ place the IP address of the **RaspberryPi PLC** device (see below for a list of the available devices and their IP addresses). Don’t forget to activate the new IP address by **pressing «enter»**. The CODESYS IDE should now confirm that it found the device in question.

You can now connect to that device, download your program, and place the PLC in run mode:

- Connect to the device using the menu option “Online → Login”.
- Download the program (if not yet downloaded) using “Online → Multiple download”.
- Start the program execution using “Debug → Start”.



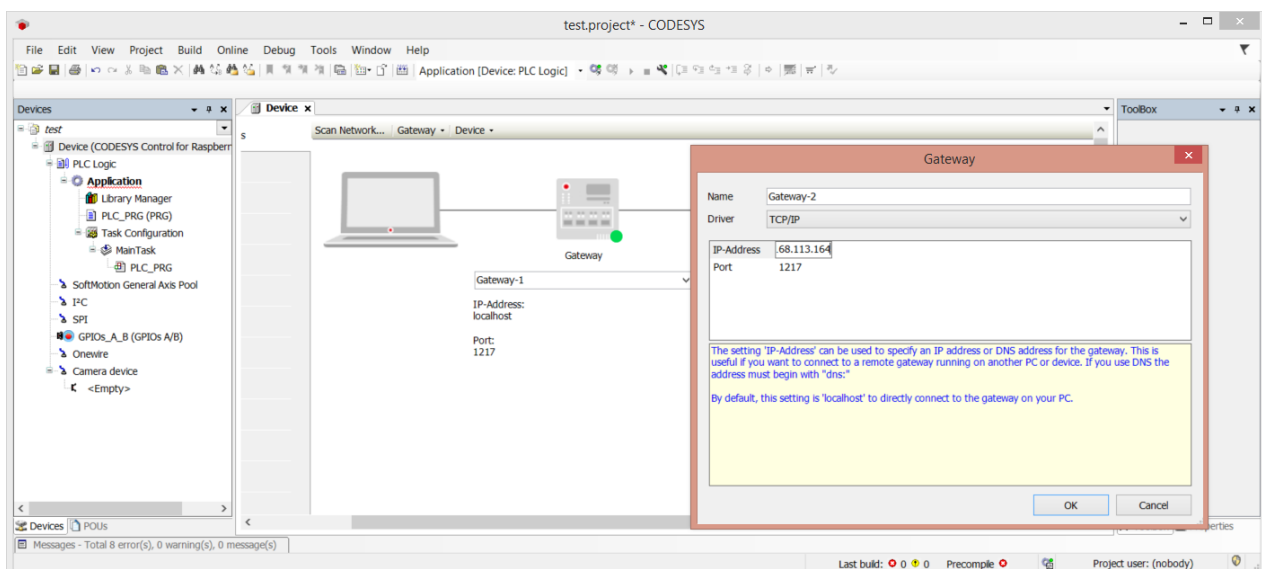
It is also possible to access a Raspberry Pi located on a non-local network - for example, from home you may access a RaspberryPi in the lab I004 if you are connected to FEUPNET by VPN. Accessing the remote RaspberryPi will be done through a gateway that is running on the server in the lab (10.227.113.1).

First create a new Gateway by choosing the menu “**Gateway** → **Add new gateway**” and configure choosing the “TCP/IP” driver and with the lab server's IP address (10.227.113.1) – Note: the “Gateway” menu is inside the device “communication settings” sub-window, it is not on the top menu list.

Activate the newly created gateway by choosing it from the gateway drop-down menu, and **press «enter»** to start the network scan. Now choose from the drop-down menu above “Device Name” the RaspberryPi you wish to access.

If no device was found, directly insert the IP address of the **RaspberryPi PLC** device (see below for a list of the available devices and their IP addresses). Don't forget to activate the new IP address by **pressing «enter»**.

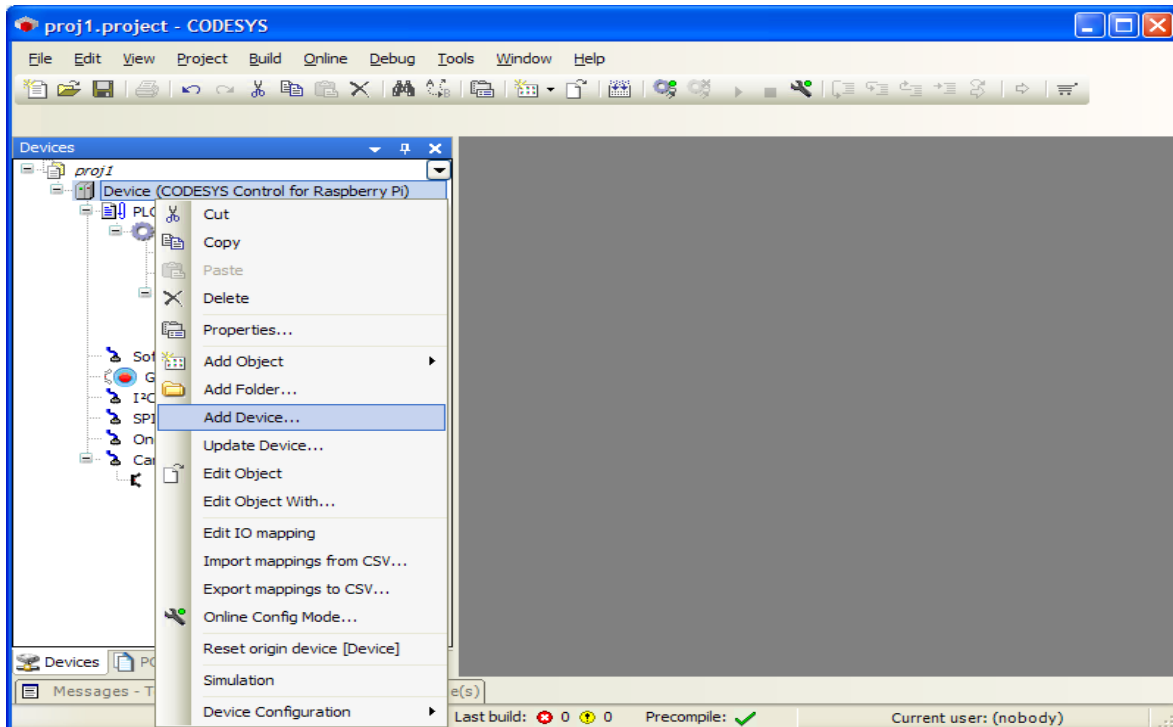
NOTE: You may need to insert a login and password to access the “Device” (i.e. the PLC running on the RaspberryPi). Use login “**admin**”, and password “**admin**”. Do not change this configuration.



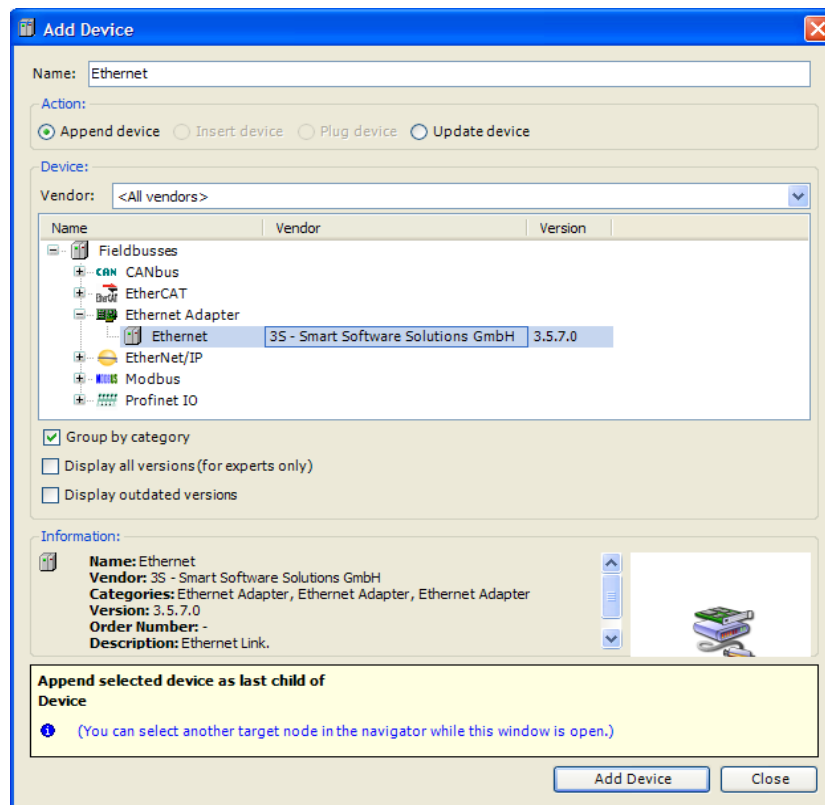
WorkBench	RaspberryPi PLC – Modbus Master		Modbus Slave	
	ID	login:pass@IP	ID	IP
Bancada 1	RaspCan01	pi:rasp@10.227.113.12	RaspMod01	10.227.113.13
Bancada 2	RaspCan02	pi:rasp@10.227.113.22	RaspMod02	10.227.113.23
Bancada 3	RaspCan03	pi:rasp@10.227.113.32	RaspMod03	10.227.113.33
Bancada 4	RaspCan04	pi:rasp@10.227.113.42	RaspMod04	10.227.113.43
Bancada 5	RaspCan05	pi:rasp@10.227.113.52	RaspMod05	10.227.113.53
Bancada 6	RaspCan06	pi:rasp@10.227.113.62	RaspMod06	10.227.113.63
Bancada 7	RaspCan07	pi:rasp@10.227.113.72	RaspMod07	10.227.113.73
Bancada 8	RaspCan08	pi:rasp@10.227.113.82	RaspMod08	10.227.113.83
Bancada 9	RaspCan09	pi:rasp@10.227.113.92	RaspMod09	10.227.113.93
Bancada 10	RaspCan10	pi:rasp@10.227.113.102	RaspMod10	10.227.113.103
Bancada 11	RaspCan11	pi:rasp@10.227.113.112	RaspMod11	10.227.113.113
Bancada 12	RaspCan12	pi:rasp@10.227.113.122	RaspMod11	10.227.113.123

4. Configuring a Modbus TCP Master and Slaves

We will begin by configuring the communication between the Modbus Master and the Modbus Slave (running on another RaspberryPi) using the Modbus TCP communication protocol. Right-click on the **Device** and choose '**Add Device**'.



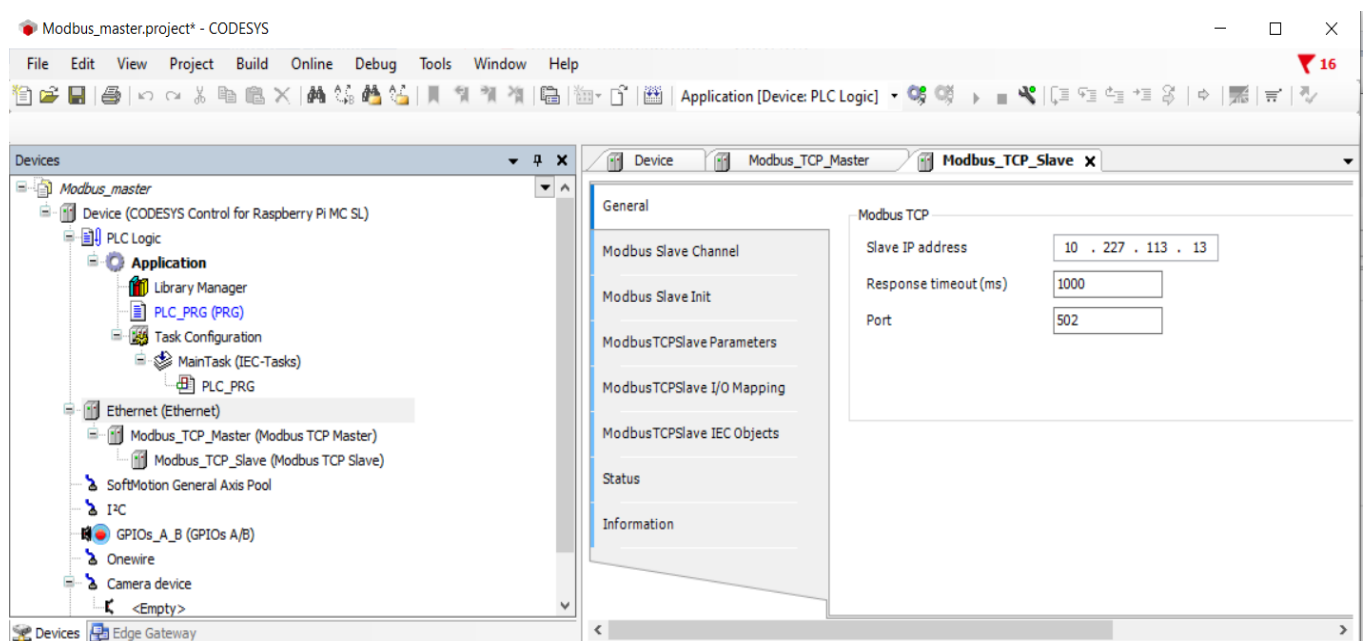
You will be shown a window where you may choose the device to add. Start off by adding an '**Ethernet Adapter**' device.



On the Project browser ('**Devices**') window on the left, select the newly created **Ethernet Adapter**, and add a '**Modbus_TCP_Master**' device. Once again, select the created '**Modbus_TCP_Master**' in the 'Devices' list and add a '**Modbus_TCP_Slave**' device.

Now you should now have an **Ethernet Adapter** containing a **Modbus TCP Master**, which in turn contains a **Modbus TCP Slave**.

Now double-click on **Modbus TCP Slave**, and its configuration window will be shown on the right. Select the **General** tab. Configure here the **IP address** of the Modbus Slave, as well as the **port**.

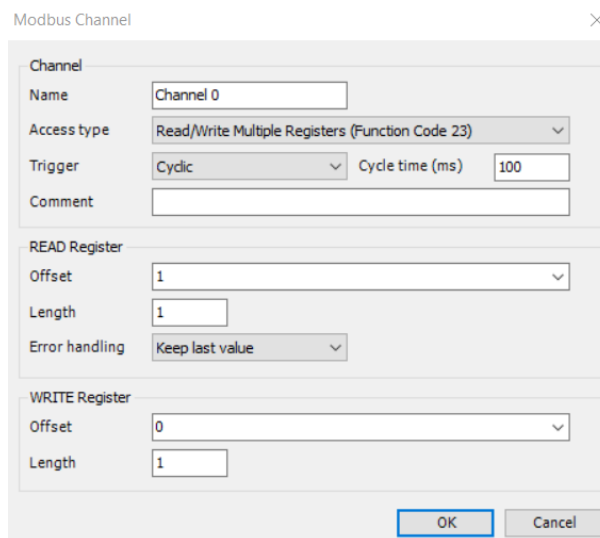


- Use **port 502** (default port for Modbus TCP).
- Make sure to **choose the correct IP address of the Modbus slave**. It should be the IP address of the second RaspberryPi on your lab workbench (see table above).

You can change other Slave parameters on the '**Modbus TCP Slave Parameters**' tab (eg.the UNIT ID). However for this assignment this is not necessary.


You will now need to configure which **Modbus functions** should be used when communicating with this slave. This is done by adding a '**Modbus Slave Channel**' for each required function. On the '**Modbus_TCP_Slave**' configuration select this tab and add the following channel (use the **Add Channel** button):

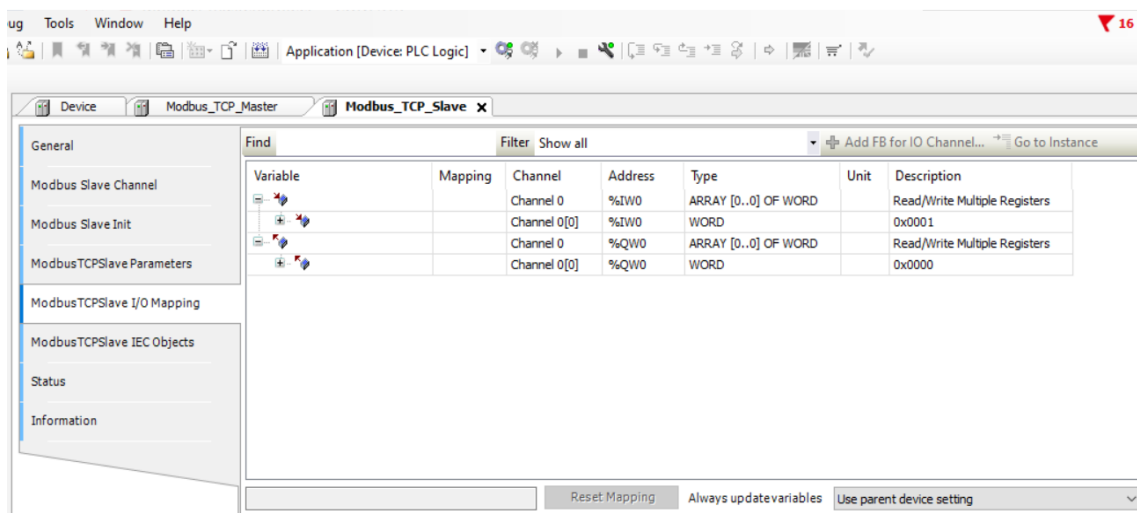
- “Read/Write Multiple Registers (Function Code 23)”, with
 - for READ register: an offset of 1, and length 1
 - for WRITE register: an offset of 0, and length 1



The image shows the 'Modbus Channel' configuration dialog box. It has a 'Channel' section with 'Name' set to 'Channel 0', 'Access type' set to 'Read/Write Multiple Registers (Function Code 23)', 'Trigger' set to 'Cyclic', and 'Cycle time (ms)' set to '100'. Below this is a 'READ Register' section with 'Offset' set to '1', 'Length' set to '1', and 'Error handling' set to 'Keep last value'. At the bottom is a 'WRITE Register' section with 'Offset' set to '0' and 'Length' set to '1'. There are 'OK' and 'Cancel' buttons at the bottom right.

You should now be able to list the status of the Holding Registers on the Modbus slave. On the '**Modbus_TCP_Slave**' configuration open the '**Modbus TCP Slave I/O Mapping**' tab.

Notice that you can determine the syntax to access these inputs and outputs in your program. On this tab click on the  of the device tree. In the example of the following image, you would have %IW1, %IX2.0, ..., %QW1, %QX2.0, %QX2.1, etc.



The image shows the 'Modbus_TCP_Slave' configuration window with the 'Modbus TCP Slave I/O Mapping' tab selected. The left sidebar shows the device tree with a plus icon next to 'Modbus_TCP_Slave'. The main area displays a table of mappings.

Variable	Mapping	Channel	Address	Type	Unit	Description
		Channel 0	%IW0	ARRAY [0..0] OF WORD		Read/Write Multiple Registers
		Channel 0[0]	%IW0	WORD		0x0001
		Channel 0	%QW0	ARRAY [0..0] OF WORD		Read/Write Multiple Registers
		Channel 0[0]	%QW0	WORD		0x0000

At the bottom of the window, there are buttons for 'Reset Mapping', 'Always update variables', and a dropdown menu set to 'Use parent device setting'.

NOTE: although the Modbus functions are configured to read the sensors and write to the actuators, if CODESYS realizes that none of these values are actually being used by any program, it does not in fact run the communication with the Modbus slave.

5. Writing a Program

Go offline from the runtime device.

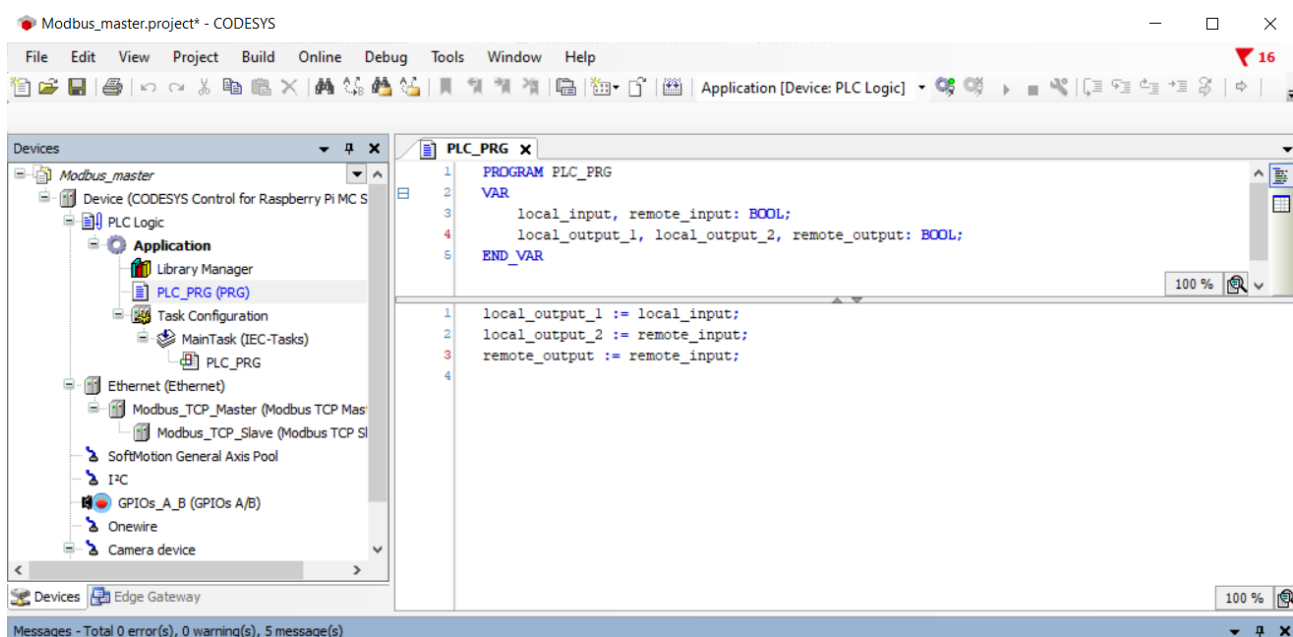
Now we are going to write the program for this assignment and verify if everything works properly.

Start by creating the variables (BOOL type) that are needed for this assignment. You can use any names, but we suggest these:

- **local_input** : input **S** connected to the master controller (RaspeberryPI Master)
- **local_output_1** : output **out1** available on the master controller (RaspeberryPI Master)
- **local_output_2** : output **out2** available on the master controller (RaspeberryPI Master)
- **remote_input**: input **Sn** connected to the slave (RaspeberryPI Slave)
- **remote_output**: outupt **An** available on the slave (RaspeberryPI Slave)

Open the program '**PLC_PRG**' (the **PLC_PRG** program is automatically inserted when the project was created) and enter the following instructions (see example):

- Read the **local_input** and write it to the **local_output_1** (necessary to compute **t1**)
- Read the **remote_input** and write it to the **local_output_2** (necessary to compute **t2**)
- Read the **remote_input** and write it to the **remote_output** (necessary to compute **t3**)



Compile, download, and run this program (note: this program is not supposed to do anything...for now).

5.1 Mapping Modbus registers

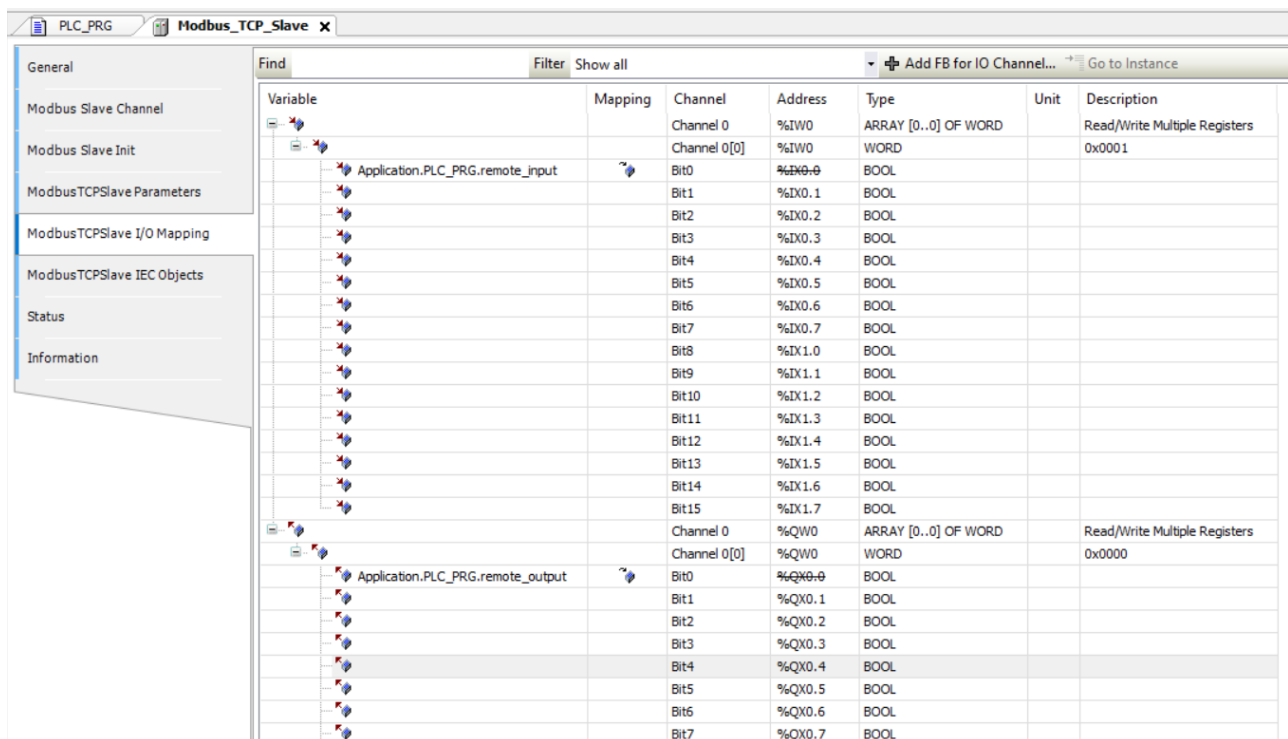
With the variables created, you can now map these variables to specific input /output Modbus registers. This is done in the '**ModbusTCPSlave IO Mapping**' tab of the slave configuration.

To map a variable defined in the program (Application.PLC_PRG) double click in the cell (i.e. the cell under the 'variable' column, not the 'mapping' column) and select the box → **Application** → **name_of_the_variable**

In our case we have 2 Modbus registers:

- An Input Register (READ register), at address 1, whose 1st bit represents **Sn**
- An Holding Register (WRITE register), at address 0, whose 1st bit represents **An**

You should map the variables **remote_input** and **remote_output** on those bits.

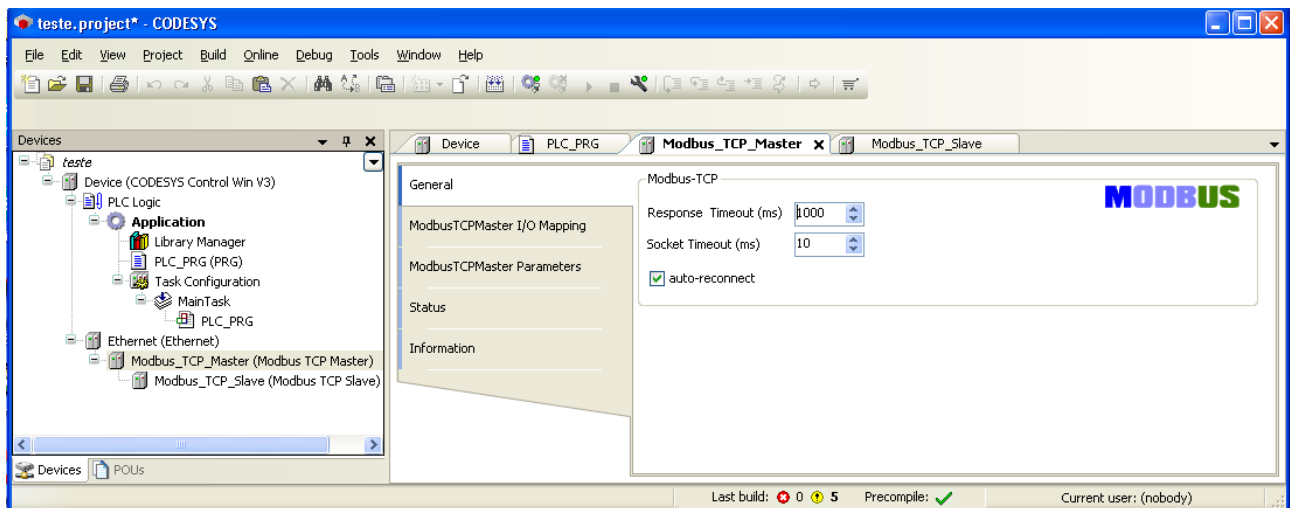


Variable	Mapping	Channel	Address	Type	Unit	Description
		Channel 0	%IW0	ARRAY [0..0] OF WORD		Read/Write Multiple Registers
		Channel 0[0]	%IW0	WORD		0x0001
Application.PLC_PRG.remote_input	Bit0		%IX0.0	BOOL		
	Bit1		%IX0.1	BOOL		
	Bit2		%IX0.2	BOOL		
	Bit3		%IX0.3	BOOL		
	Bit4		%IX0.4	BOOL		
	Bit5		%IX0.5	BOOL		
	Bit6		%IX0.6	BOOL		
	Bit7		%IX0.7	BOOL		
	Bit8		%IX1.0	BOOL		
	Bit9		%IX1.1	BOOL		
	Bit10		%IX1.2	BOOL		
	Bit11		%IX1.3	BOOL		
	Bit12		%IX1.4	BOOL		
	Bit13		%IX1.5	BOOL		
	Bit14		%IX1.6	BOOL		
	Bit15		%IX1.7	BOOL		
		Channel 0	%QW0	ARRAY [0..0] OF WORD		Read/Write Multiple Registers
		Channel 0[0]	%QW0	WORD		0x0000
Application.PLC_PRG.remote_output	Bit0		%QX0.0	BOOL		
	Bit1		%QX0.1	BOOL		
	Bit2		%QX0.2	BOOL		
	Bit3		%QX0.3	BOOL		
	Bit4		%QX0.4	BOOL		
	Bit5		%QX0.5	BOOL		
	Bit6		%QX0.6	BOOL		
	Bit7		%QX0.7	BOOL		

5.2 Final Notes

You will notice that for the PLC (acting as the ModbusTCP Client) to correctly communicate with the Modbus slave you will need to guarantee that the slave is running **before** downloading and launching the PLC program. This is because, by default, the PLC will only try to connect to the Modbus Slave after a cold boot. If, upon program start-up, the PLC is unable to establish a connection to the Modbus slave then the connection will never be established.

To change this behavior and have the PLC program continuously try to reconnect to the Modbus Slave after a communication failure, please tick the option "**Auto-reconnect**" on the Modbus Master **General** tab.



6. Mapping Local I/Os

In this assignment we will need to access to the GPIO (General Purpose Input/Output) pins that are available on the Raspberry running the Modbus Master. It is therefore necessary to configure the access to those pins.

6.1 GPIO Device

First, is necessary to configure a GPIO Device. Do one of the two following procedures:

- If your project already has a '**GPIOs_A_B**' device, you will need to change it to a more recent device named "GPIOs_A_B (GPIOs B+ /Pi2)". Right click on the '**GPIOs_A_B**' and from the context menu choose "**Update device**". Change the device to '**GPIOs_A_B (GPIOs B+ /Pi2)**'.
- If your project does not have a '**GPIOs_A_B**' (or similar) device, you will need to add it. Right click on the "Device (CODESYS Control for Raspberri Pi ...)" and from the context menu choose '**Add device**'. Choose the device '**GPIOs_A_B (GPIOs B+ /Pi2)**'.

6.2 Mapping Local I/Os

The following GPIO pins are used by the setup that will measure the local and networked response times of the PLC:

- GPIO pin 26 → Input **S**
- GPIO pin 16 → Output **Out1**
- GPIO pin 20 → Output **Out2** (for CanOpen)
- GPIO pin 21 → Output **Out2** (for Modbus)

First, we will configure the access to the individual pins. By clicking on the '**GPIOs_A_B (GPIOs B+ Pi2)**' device in the device tree, the device's property window will open on the right:

- On the '**GPIOs Parameters**' tab of the property window you can see the current mapping of the I/Os. Each GPIO pin can be mapped as an Input or an Output. On the '**Value**' column set Bit 26 to '**Input**', and bits 16 and 21 to '**Output**'.

GPIOs_A_B						
GPIOs Parameters						
Parameter	Type	Value	Default Value	Unit	Description	
GPIO4	Enumeration of BYTE	not used	not used		configuration of GPIO4	
GPIO5	Enumeration of BYTE	not used	not used		configuration of GPIO5	
GPIO6	Enumeration of BYTE	not used	not used		configuration of GPIO6	
GPIO12	Enumeration of BYTE	not used	not used		configuration of GPIO12	
GPIO13	Enumeration of BYTE	not used	not used		configuration of GPIO13	
GPIO16	Enumeration of BYTE	Output	not used		configuration of GPIO16	
GPIO17	Enumeration of BYTE	not used	not used		configuration of GPIO17	
GPIO18	Enumeration of BYTE	not used	not used		configuration of GPIO18	
GPIO19	Enumeration of BYTE	not used	not used		configuration of GPIO19	
GPIO20	Enumeration of BYTE	Output	not used		configuration of GPIO20	
GPIO21	Enumeration of BYTE	Output	not used		configuration of GPIO21	
GPIO22	Enumeration of BYTE	not used	not used		configuration of GPIO22	
GPIO23	Enumeration of BYTE	not used	not used		configuration of GPIO23	
GPIO24	Enumeration of BYTE	not used	not used		configuration of GPIO24	
GPIO25	Enumeration of BYTE	not used	not used		configuration of GPIO25	
GPIO26	Enumeration of BYTE	Input	not used		configuration of GPIO26	
GPIO27	Enumeration of BYTE	not used	not used		configuration of GPIO27	

- On the “**GPIOs I/O Mapping**” tab of the property window you can see the current mapping of local I/Os to variables declared in the program. To perform the allocation of the variables to the GPIO pins, just follow a pattern similar to the mapping of Modbus registers: select the bit → select box → Application → **name_of_the_variable**

Repeat the above procedure for the variables **local_input**, **local_output_1** and **local_output_2**.

GPIOs_A_B						
GPIOs Parameters						
GPIOs I/O Mapping						
Variable	Mapping	Channel	Address	Type	Unit	
Application.PLC_PRG.local_input		Bit26	%IX7.2	BOOL		
		Bit27	%IX7.3	BOOL		
		digital outputs (GPIO0..GPIO31)	%QD1	DWORD		
		Bit4	%QX4.4	BOOL		
		Bit5	%QX4.5	BOOL		
Application.PLC_PRG.local_output_1		Bit6	%QX4.6	BOOL		
		Bit12	%QX5.4	BOOL		
		Bit13	%QX5.5	BOOL		
		Bit16	%QX6.0	BOOL		
		Bit17	%QX6.1	BOOL		
Application.PLC_PRG.local_output_2		Bit18	%QX6.2	BOOL		
		Bit19	%QX6.3	BOOL		
		Bit20	%QX6.4	BOOL		
		Bit21	%QX6.5	BOOL		
		Bit22	%QX6.6	BOOL		
		Bit23	%QX6.7	BOOL		

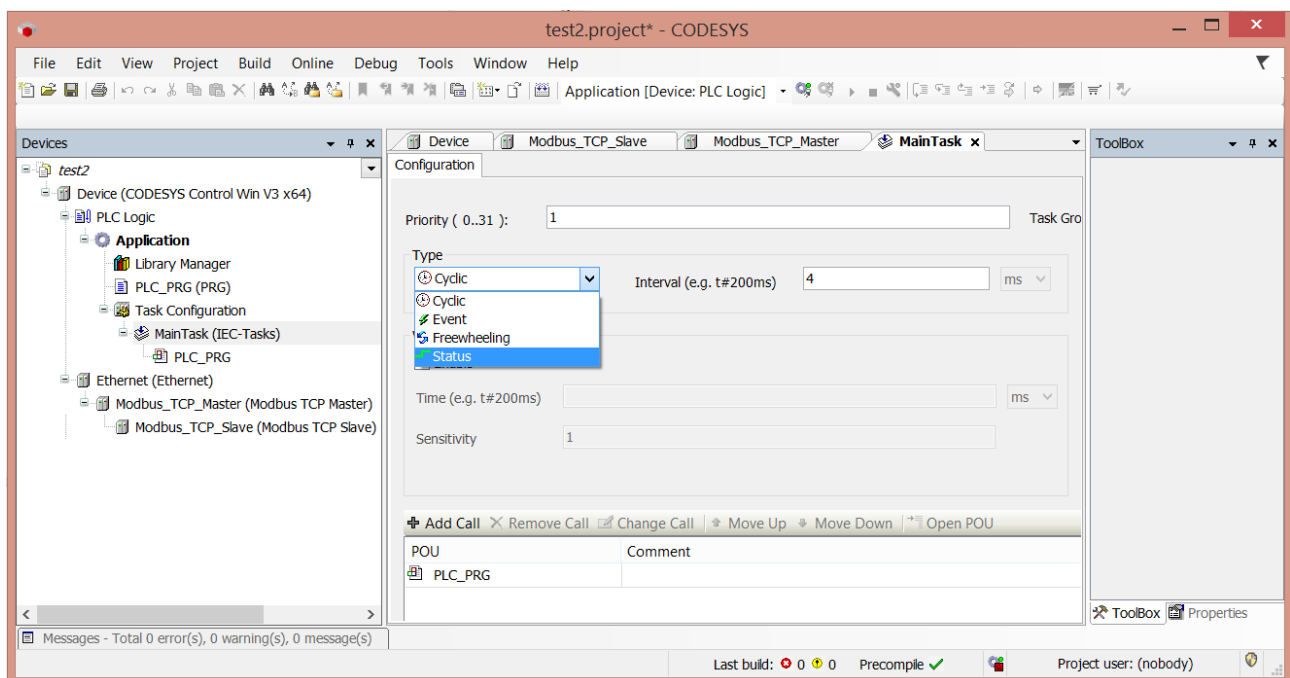
Output Bit 20 will be used to test the CANopen network. Later on, once you have the CANopen network configured, you will then need to map this output bit to the value read from the input on the remote CANopen slave.

7. Changing the configuration parameters

7.1 Changing Program Parameters

By double-clicking on the '**MainTask**' on the project explorer panel (window on the left) you can access the task (and program) activation parameters.

- Cyclic: The task/program (PLC scan cycle) is activated periodically, using the user defined period ('interval'). Appropriate delays are inserted at the end of each scan cycle to guarantee periodic activation.
- Freewheeling: The task/program is executed continuously / cyclically. As soon as it finishes executing one scan cycle, the next is started again.
- Status/Event: The execution of the task/program is activated by an event on a user configured variable.



7.2 Changing Modbus parameters

When creating or double-clicking on a '**Modbus slave Channel**' tab, the channel configuration window will open.

ModbusChannel

Channel

Name: Channel 0

Access type: Read Holding Registers (Function Code 3)

Trigger: Cyclic (dropdown menu open showing: Cyclic, Rising edge, Application)

Cycle time (ms): 100

Comment:

READ Register

Offset: 0x0000

Length: 1

Error handling: Keep last Value

WRITE Register

Offset: 0x0000

Length: 0

OK Cancel

The trigger parameter configures when the Modbus function is sent to the slave.

- Cyclic: The function is sent periodically, using the user defined period ('cycle time')
- Rising edge: The function is sent when a Boolean variable changes from 0 to 1. The Boolean variable that controls the function activation is defined in the I/O mapping window.
- Application: The function execution is to be controlled by the PLC application using a FB of type 'ModbusChannel'

ANNEX – More Information on Installing CODESYS

Remember you must be on FEUPNET to access labs-automacao.fe.up.pt. If necessary, connect to this network using VPN.

The lab repository for CODESYS is available here:

`ftp://labs-automacao.fe.up.pt/I005/Demos%20Software/CODESYS/`

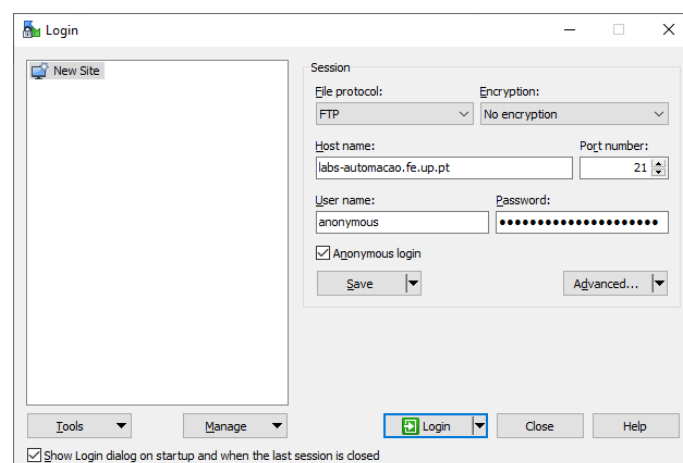
However, FTP is not that much supported anymore, at least by the browsers. Thus, open an **Window Explorer** window, copy the whole address line and paste it in the address line in the window header.

An alternative is to install an FTP client, for example WinSCP

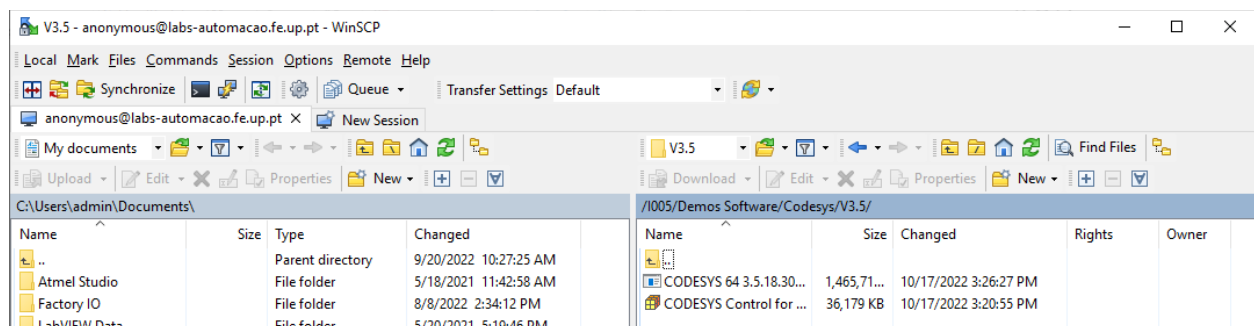
<https://winscp.net/eng/index.php>



Select FTP as protocol and use as Host name “labs.automacao.fe.up.pt” then tick the anonymous login box.



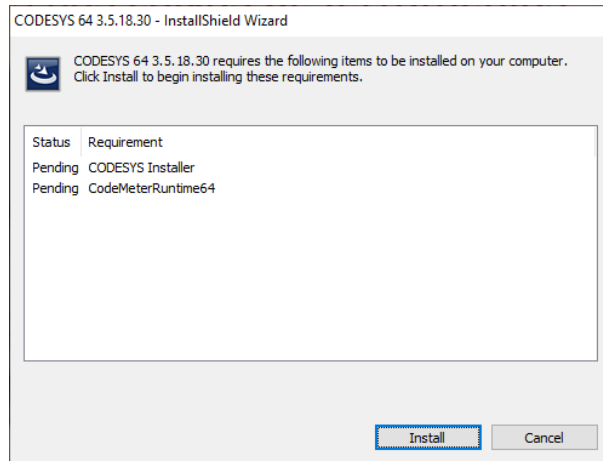
Download the contents of folder “/i005/Demos Software/Codesys/V3.5”



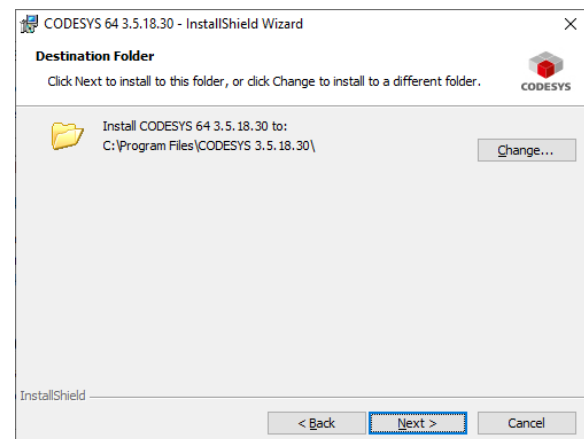
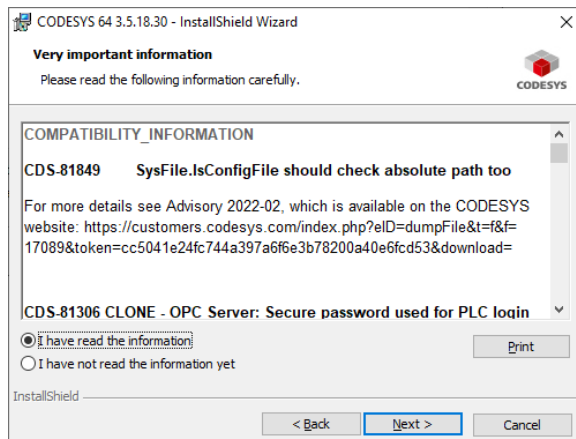
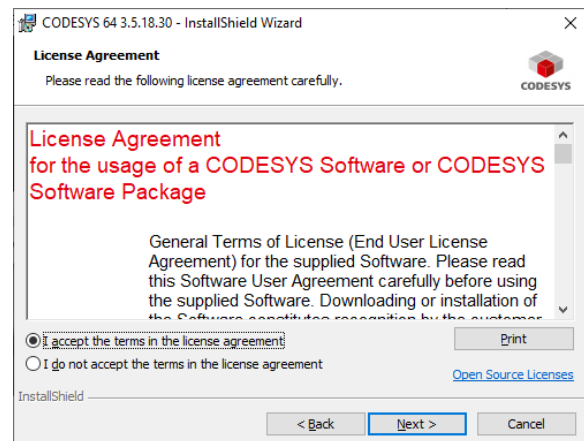
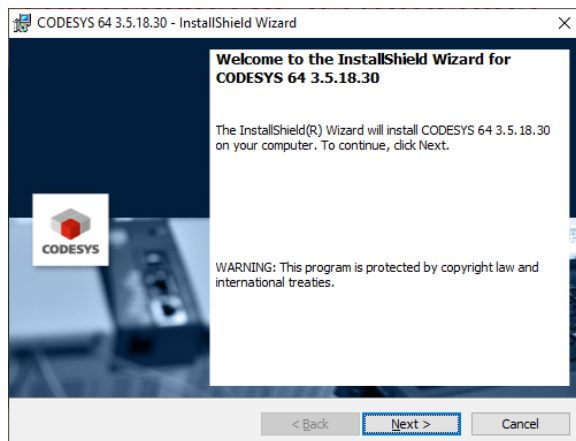
Execute file “CODESYS 64 3.5.18.30.exe”

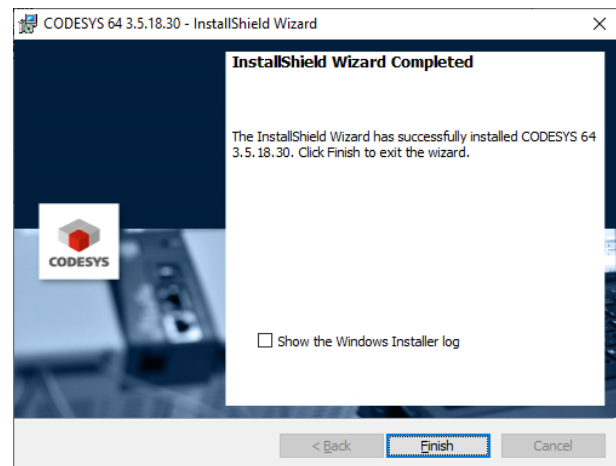
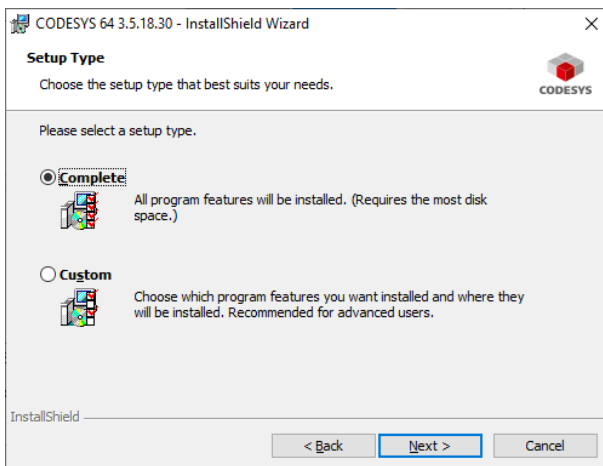


Click on Install. The requirements list may vary and you may need to restart the computer before proceeding with the remainder of the installation process.



Continue the installation accepting the terms and confirming you read the information. It is recommended to do a complete CODESYS installation. The installation process can take between 10 and 45 minutes according to the speed of the computer.

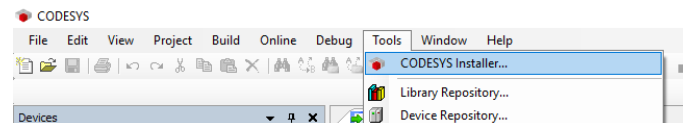




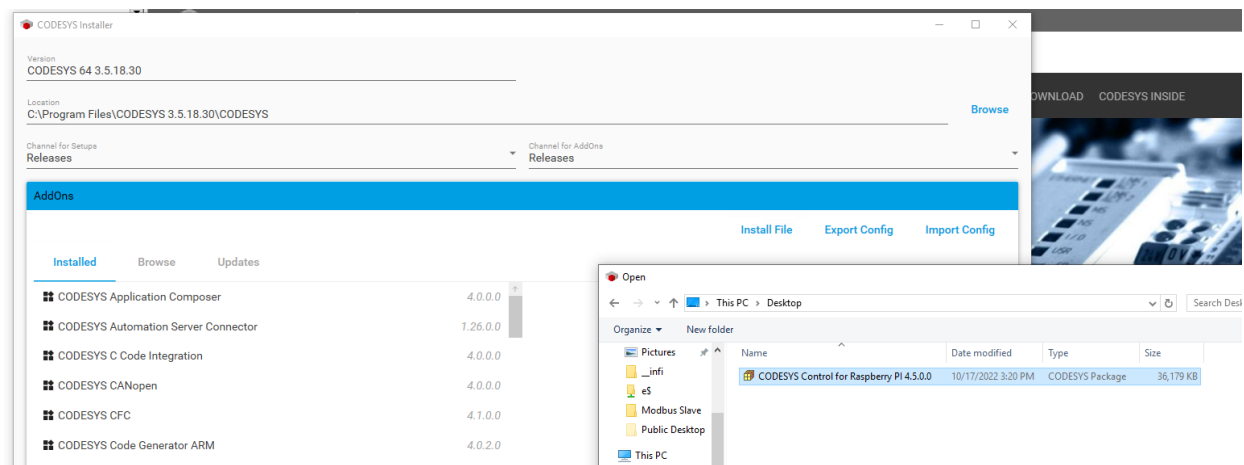
Open CODESYS after the installation ended



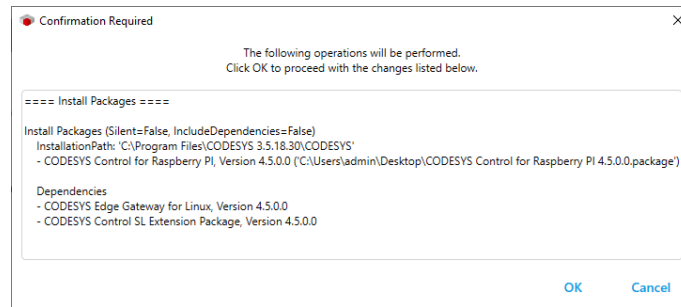
Go to “Tools” and open “CODESYS Installer”



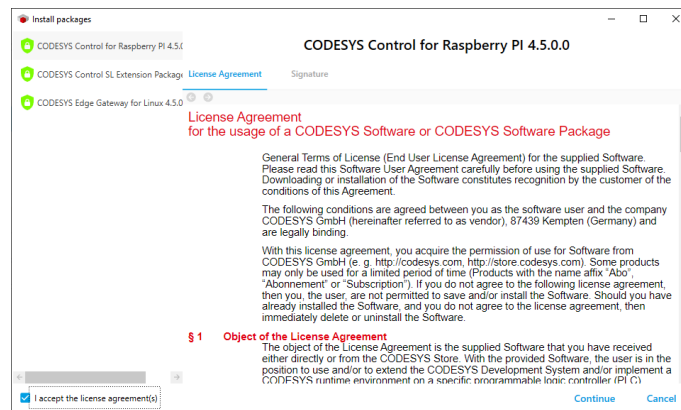
Click on “install File” and localize file “CODESYS Control for Raspberry PI 4.5.0.0.package”



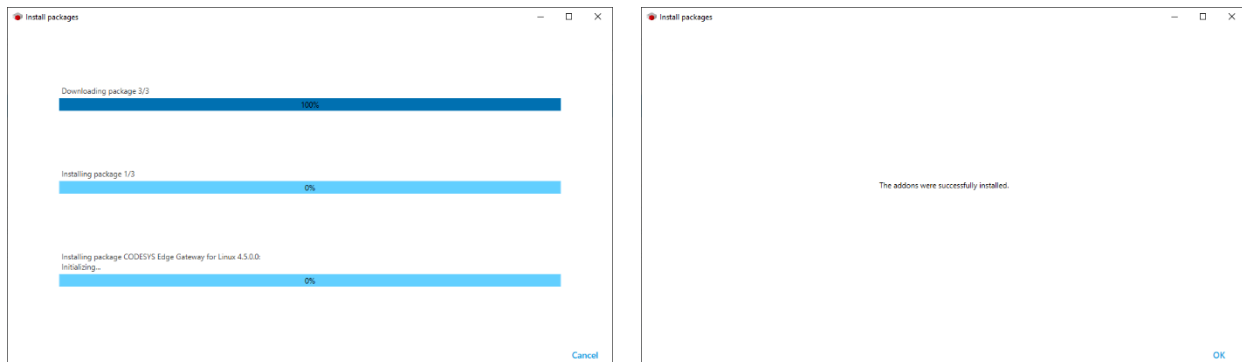
The following configuration should show up with dependencies to be resolved and which will be installed automatically.



Accept the license and click on “Continue”



Wait for the end of the installation.



At this point CODESYS should be installed in your computer with support for the RaspberryPi