Industrial communication labs (Schneider)

1. Introduction

Modern complex Industrial Control System (ICS), which includes SCADA and Distributed Control Systems (DCS), heavily relies on the use of a communication system tailored for the real-time and reliability requests of distributed applications.

In contrast with general communication systems (like Ethernet/TCP/IP networks and other Internet related technologies) industrial communication systems are part of the controller and are able to guarantee that real-time constrains of the application will be met. Despite the fact that some of the protocols are Ethernet embedded (like GOOSE and ProfinetI/O) or even TCP/IP transported (like ModbusTCP and S7) the deployment of the TCP/IP stack requires some modifications in the Internet protocols (like sockets reuse, never-closing TCP connection, Ethernet Vlan frames tagging and priority, etc) that brings the TCP/IP stack close to a deterministic behavior.

Therefore, for a control engineer, the practical knowledge of industrial communication protocols like behavior of the data flows, relationship between PLC programs and network connections, bandwidth use and optimization is an important requirement in order to be able to deploy an industrial control system. The purpose of these labs is to let you acquire a minimal knowledge in the industrial communication field.

Lab objectives

1) Program an application using Grafcet
2) Understand the communication between PLC and HMI/SCADA
2) Implement and observe communication flows
3) Analyze the flows, optimize the communication

2. General organization of the lab, work environment

The lab is organized in two workshops corresponding to the study of one major SCADA protocol each: S7 (Siemens) and Modbus/TCP (Schneider). Several PLCs are available for S7 and for Modbus/TCP, together with development computers, a SCADA software (TIA Portal for S7 and PC Vue for Modbus/TCP) and HMIs. Each student will work alone on one machine.

2.1 Network :

The 10.10.0.0/16 network interconnects different elements :
- the PLCs ,
- the "configuration" computers (10.10.3.x),
- the "process simulator" cards (10.10.100.x, not visible on the general diagram in Appendix A).
- the Human Machine Interfaces (HMI).

These elements are interconnected via two CISCO switches.

Control architecture and supervision architecture

You have to understand and specify the global functioning of the architecture and the control loop. This architecture consists of :

Element of the architecture for control Interfaces	Interfaces
A PLC to receive the command	- Analog and digital inputs/outputs - Network card - Special case: 10.10.4.1: Remote I/O via the left ET200S interface via Profibus (the one without IP address...)
A "physical process" represented by a simulation card (via address 10.10.100.x) embedded SMT32.	- Analog and digital inputs/outputs - Network card
A remote control interface (GICS Tester) implemented on the "configuration" machine (10.10.3.x)	- Network card

QUESTION : Draw the control loop with the controller, the actuators, the sensors, the action and specify the interfaces used and the links/modes of "connections" (ex: network(x), direct link I/O...)

Beyond the control loop, the architecture is also used for the configuration.

2.2 Identification of your working space

The first aspect will be to identify your computer (3.X), your PLC, your HMI, your simulation card, and their relative IP addresses. These addresses should be kept (not changed). Describe your architecture, the interconnections between elements, the IP addresses...

2.3 Getting to know your workspace

On each computer a local account is opened for the students (ex MISTREA01, you must ask the teacher what is your login and password on your machine, and note it down).
The useful software are:
- Software to configure your PLC: TIA Portal for Siemens and Unity Pro XL for Schneider
- A remote control interface GICS Tester

QUESTION: What is the use of the GICS Tester control interface?

2.4 Configuration of the GICS Tester remote control interface

On the GICS Tester control interface (the "configuration" computer), the IP address of the simulation card you are using must be configured as "target IP": 10.10.100.x, on port 2015. We will not use the analog inputs and outputs (the digital values are between 0 and 4095 for voltages between -10 and +10 V).
From now, you can test the outputs by forcing them through the GICS Tester control interface and by observing their variations on the physical interfaces of the I/O cards.

DO NOT TOUCH THE PHYSICAL WIRING.

If necessary, use a table to match the I/O numbers on the physical interface with the numbers on the GICS Tester control interface.

2.5 The simulation card

For practical (available space) reasons, it is not possible to have 12 plants to be controlled by the 12 PLCs. Therefore the plant is simulated by a "simulation card" which is actually an embedded system built around a microprocessor and allowing programming for simulation of plants... However the PLC receives "true signals" on the I/O card interacting with the "simulation card". Figure 2 displays the lab configuration (it is called a Hardware in the Loop simulation).

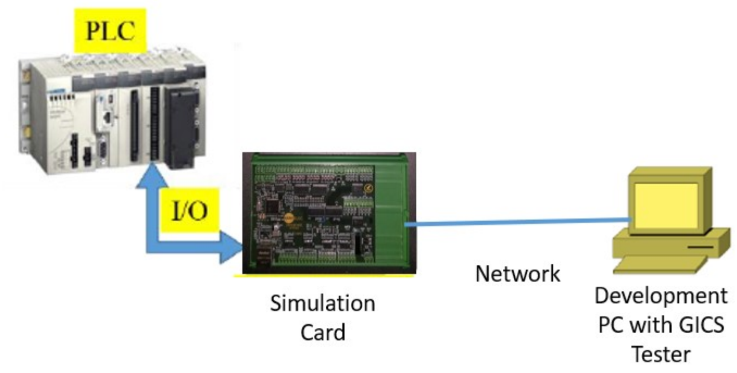


Figure 2. Hardware in the loop bench

3. Work to do.

You'll have to:

A. Write a control program for the PLC (to control the tank according to the example given in the slides **2. PLC**).

B. Set-up the communication between the SCADA and the PLC

C. Check the communication flows using a network sniffer

Control specification.

The control specification concerns a system based on a tank to mix-up two products (one yellow and one blue product) into a green product. All the specifications are detailed in the set of slides 2. PLC.

4. Configuration of the PLC and I/O

In the first part of the lab you have to build a simple SCADA system. The system schematics are presented in Figure 1.

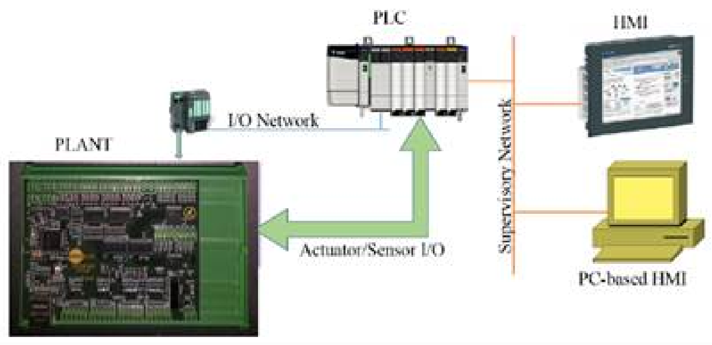


Figure 1. The simple SCADA system

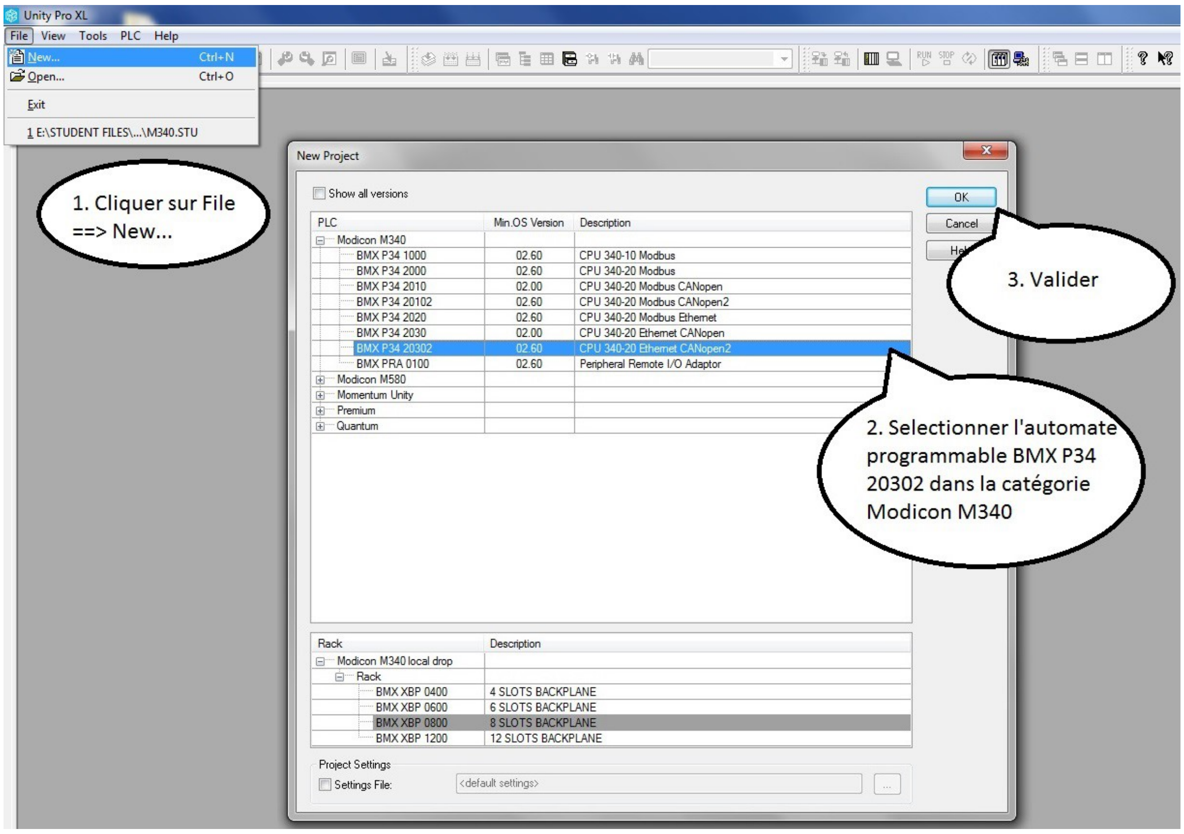
The PLC will interact with the plant either by direct data exchange with the sensor and actuators connected to the I/O cards or through a remote I/O unit accessible via a dedicated network. The process data (including inputs, outputs and device status) is collected by the SCADA system and displayed by a field HMI or a PC-based software.

4.1 Project of configuration of your PLC

On the "configuration" computer, you will configure your PLC (depending on the PLC and the environment, these operations can be more or less automated...), the objective is that you do it to see and understand the different steps. It is an operation where you have to be rigorous and careful, choose the right elements and configure them well, because then the corresponding drivers will be sent to the PLC... and must correspond to the precise elements of the physical architecture of the PLC. You have to configure everything including the units you will not use (for example the analog I/O) because the software and hardware configurations have to match.

The figures below are for the Schneider PLC of the Modicon M340 family. Configuration in Unity Pro XL .

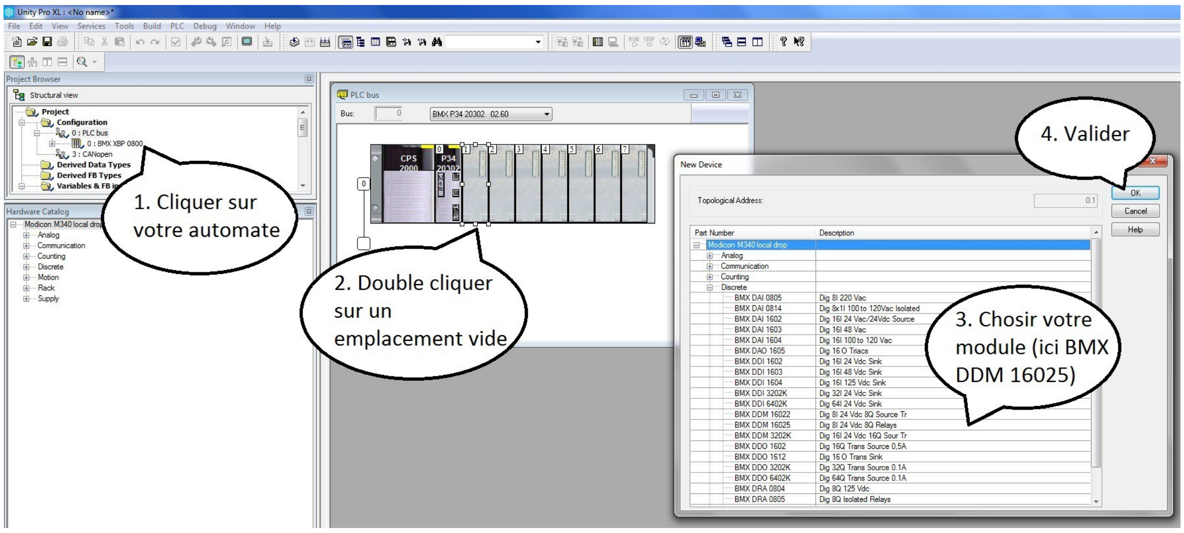
**PLEASE TAKE CARE THESE FIGURES DOES NOT CORRESPOND EXACTLY TO YOUR ARCHITECTURE,
THESE ARE EXAMPLES AND YOU SHOULD ADAPT TO YOUR SPECIFICATIONS**



PLC	Min.OS Version	Description
Modicon M340		
BMX P34 1000	02.60	CPU 340-10 Modbus
BMX P34 2000	02.60	CPU 340-20 Modbus
BMX P34 2010	02.00	CPU 340-20 Modbus CANopen
BMX P34 20102	02.60	CPU 340-20 Modbus CANopen2
BMX P34 2020	02.60	CPU 340-20 Modbus Ethernet
BMX P34 2030	02.00	CPU 340-20 Ethernet CANopen
BMX P34 20302	02.60	CPU 340-20 Ethernet CANopen2
BMX PRA 0100	02.60	Peripheral Remote I/O Adaptor
Modicon M580		
Momentum Unity		
Premium		
Quantum		

Rack	Description
Modicon M340 local drop	
Rack	
BMX XBP 0400	4 SLOTS BACKPLANE
BMX XBP 0600	6 SLOTS BACKPLANE
BMX XBP 0800	8 SLOTS BACKPLANE
BMX XBP 1200	12 SLOTS BACKPLANE

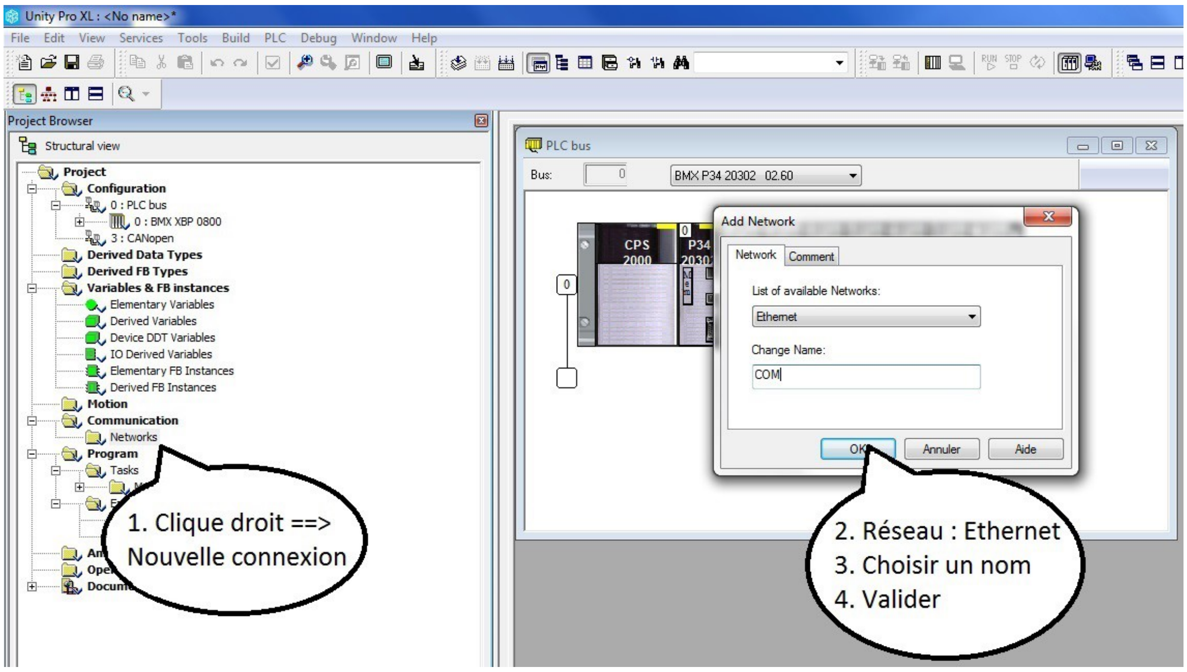
TAKE CARE HERE: OUR CONFIGURATION IS GENERALLY BMX PRA 100…


1. Cliquer sur votre automate
2. Double cliquer sur un emplacement vide
3. Chosir votre module (ici BMX DDM 16025)
4. Valider
New Device
OK
Cancel
Help

Topological Address: 0.1

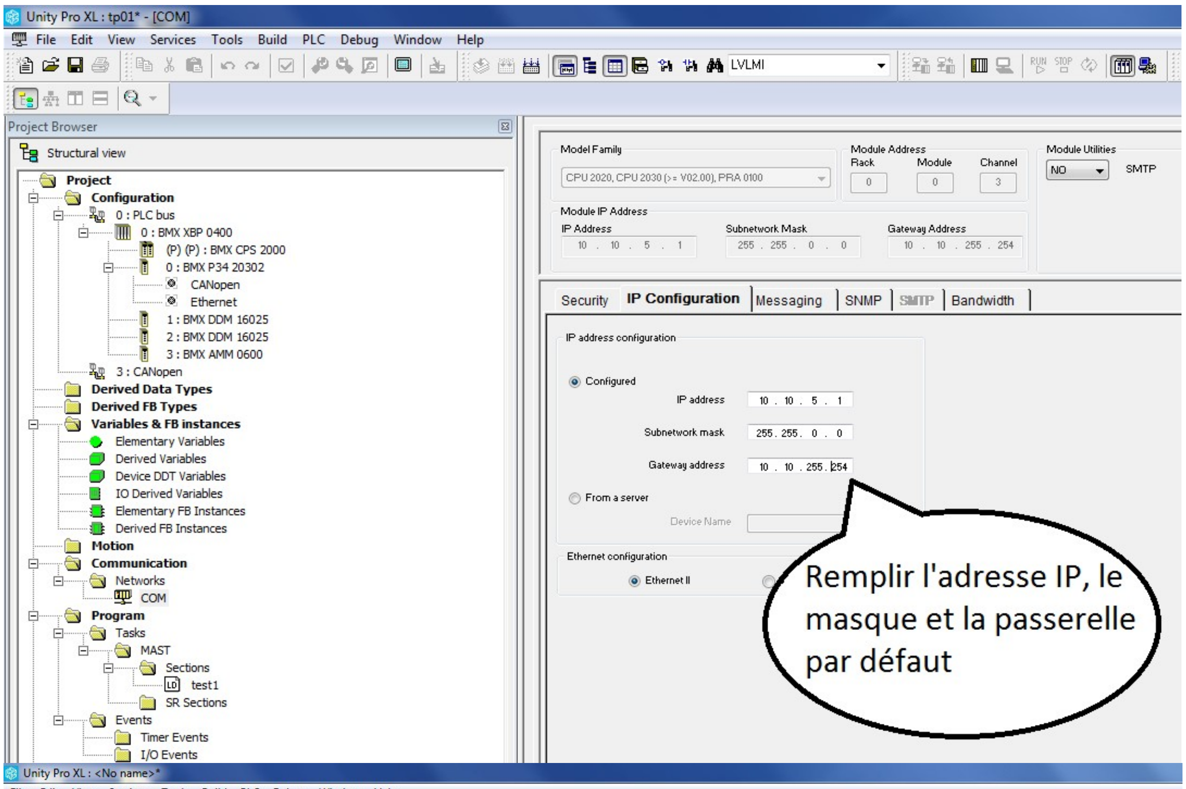
Part Number	Description
Modicon M340 local drop	
Analog	
Communication	
Counting	
Discrete	
BMX DAI 0805	Dig 8I 220 Vac
BMX DAI 0814	Dig 8x1I 100 to 120Vac Isolated
BMX DAI 1602	Dig 16I 24 Vac/24Vdc Source
BMX DAI 1603	Dig 16I 48 Vac
BMX DAI 1604	Dig 16I 100 to 120 Vac
BMX DAO 1605	Dig 16 O Triacs
BMX DDI 1602	Dig 16I 24 Vdc Sink
BMX DDI 1603	Dig 16I 48 Vdc Sink
BMX DDI 1604	Dig 16I 125 Vdc Sink
BMX DDI 3202K	Dig 32I 24 Vdc Sink
BMX DDI 6402K	Dig 64I 24 Vdc Sink
BMX DDM 16022	Dig 8I 24 Vdc 8Q Source Tr
BMX DDM 16025	Dig 8I 24 Vdc 8Q Relays
BMX DDM 3202K	Dig 16I 24 Vdc 16Q Sour Tr
BMX DDO 1602	Dig 16Q Trans Source 0,5A
BMX DDO 1612	Dig 16 O Trans Sink
BMX DDO 3202K	Dig 32Q Trans Source 0.1A
BMX DDO 6402K	Dig 64Q Trans Source 0.1A
BMX DRA 0804	Dig 8Q 125 Vdc
BMX DRA 0805	Dig 8Q Isolated Relays

4.2 Configuration of the PLC network interface, clock synchronization



List of available Networks: Ethernet

Change Name: COM


Model Family: CPU 2020, CPU 2030 (>= V02.00), PRA 0100

Module Address: Rack 0, Module 0, Channel 3

Module Utilities: NO SMTP

Module IP Address: IP Address 10 . 10 . 5 . 1; Subnetwork Mask 255 . 255 . 0 . 0; Gateway Address 10 . 10 . 255 . 254

IP address configuration: (●) Configured

IP address: 10 . 10 . 5 . 1

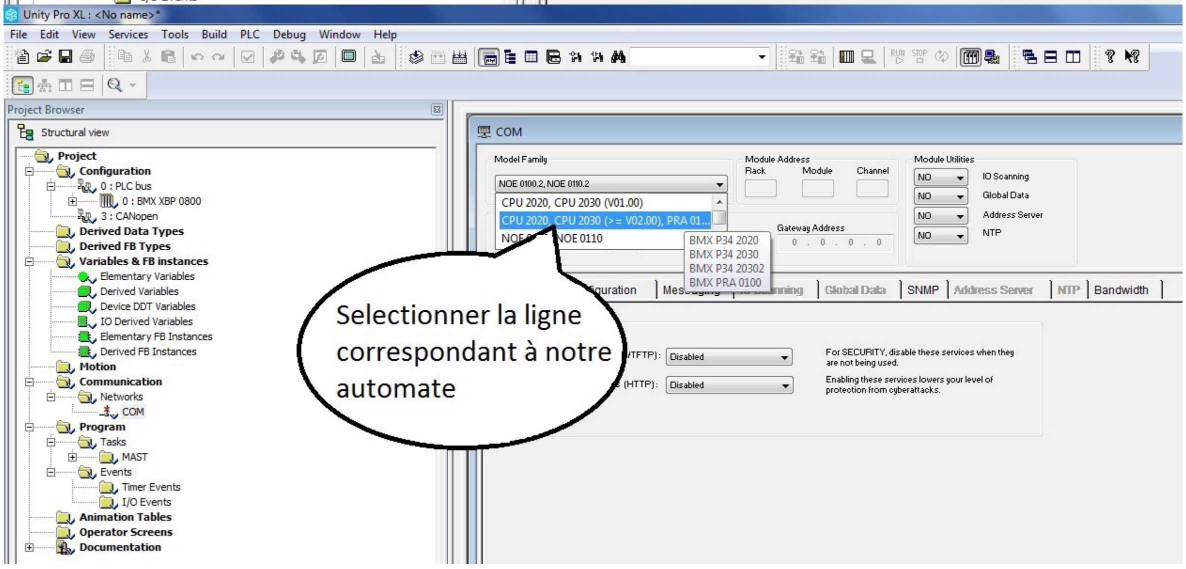
Subnetwork mask: 255 . 255 . 0 . 0

Gateway address: 10 . 10 . 255 . 254

( ) From a server

Ethernet configuration: (●) Ethernet II

Unity Pro XL : tp01* - [COM]
Security | IP Configuration | Messaging | SNMP | SMTP | Bandwidth
Remplir l'adresse IP, le masque et la passerelle par défaut


Model Family: NOE 0100.2, NOE 0110.2

CPU 2020, CPU 2030 (V01.00)

CPU 2020, CPU 2030 (>= V02.00), PRA 01...

NOE 0100, NOE 0110

BMX P34 2020, BMX P34 2030, BMX P34 20302, BMX PRA 0100

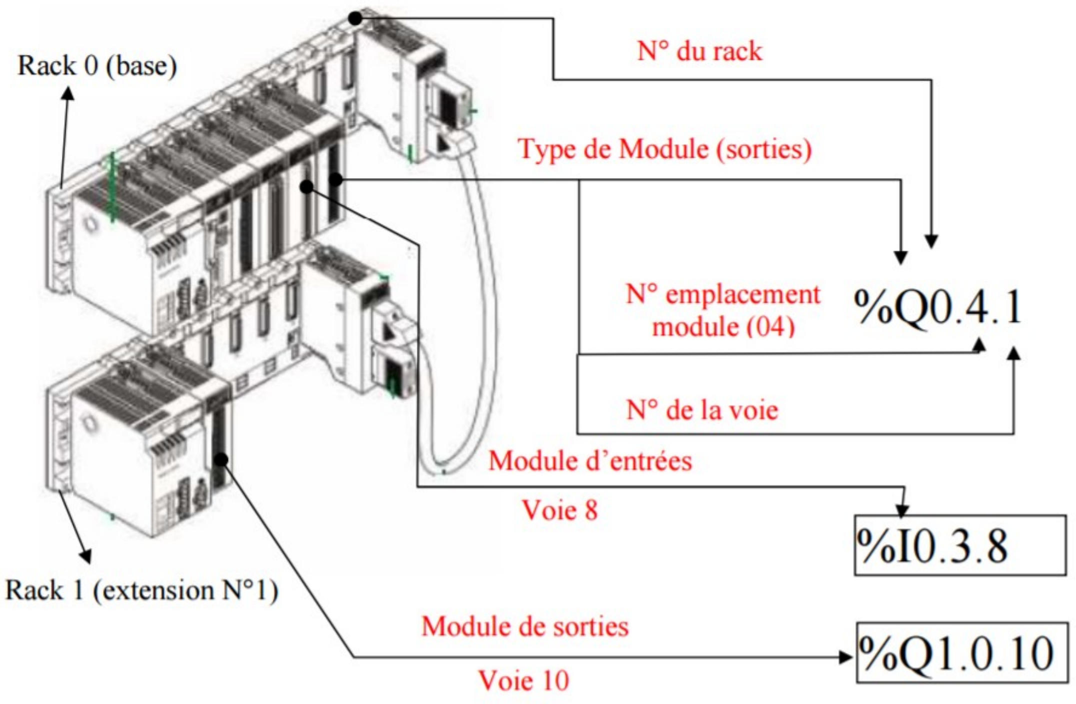
For SECURITY, disable these services when they are not being used.
Enabling these services lowers your level of protection from cyberattacks.

Unity Pro XL : <No name>*
COM
Selectionner la ligne correspondant à notre automate

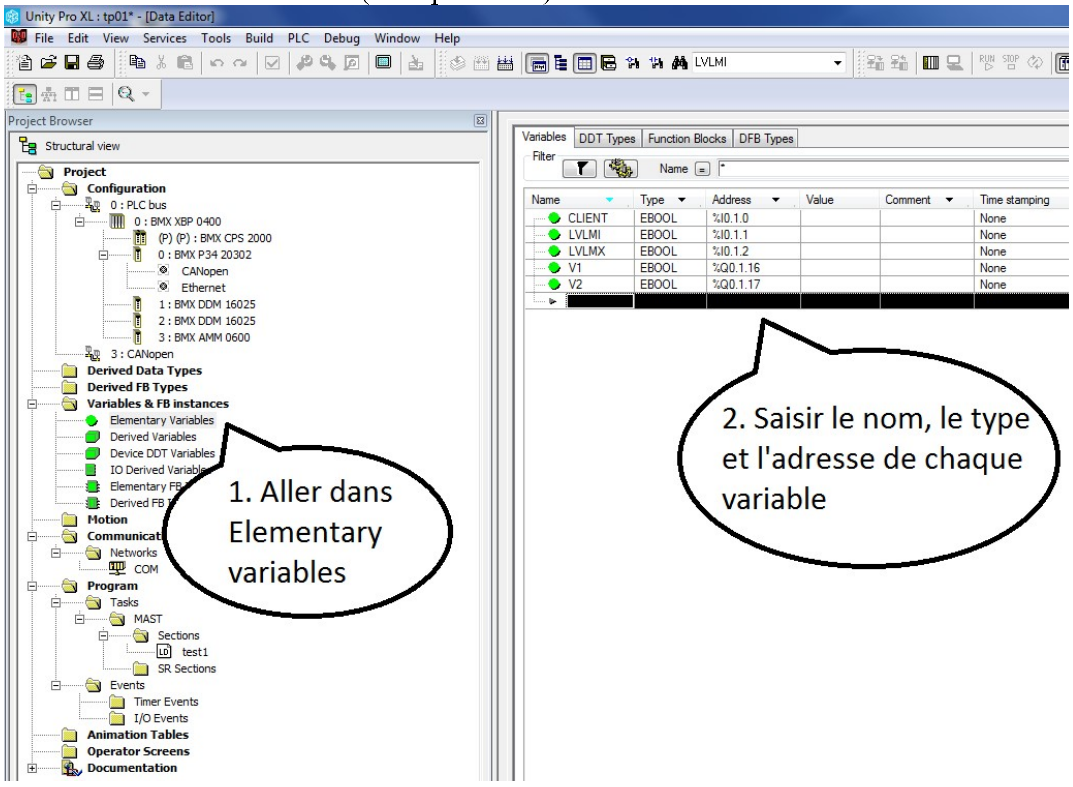
TAKE CARE ONCE AGAIN ABOUT YOUR OWN REAL CONFIGURATION…

4.3 Configuration of the variables

After the configuration of the PLC, it is necessary to define the variables that the PLC needs. To add a variable, you have to give it a name, define its type and know the address that corresponds to it on the I/O module ports of the PLC. To define the latter we can use the following method:



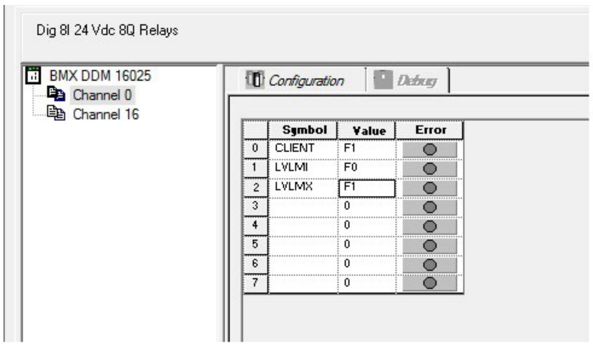
You must then create the variables (example below):



Name	Type	Address	Value	Comment	Time stamping
CLIENT	EBOOL	%I0.1.0			None
LVLMI	EBOOL	%I0.1.1			None
LVLMX	EBOOL	%I0.1.2			None
V1	EBOOL	%Q0.1.16			None
V2	EBOOL	%Q0.1.17			None

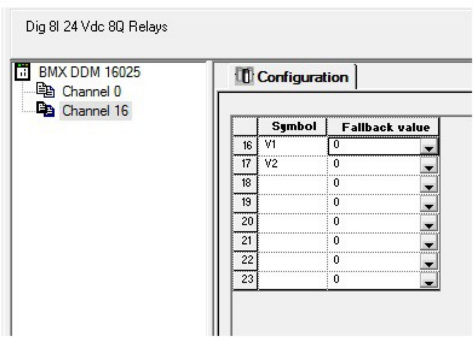
The variables created are of type EDT, which stands for Elementary Data Type, which includes simple rather than derived data types (arrays, structures, function blocks). The available EDT types are: BOOL,

EBOOL, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, DATE, TOD and DT. In our case we will use variables of type EBOOL because this type offers us the possibility to force the desired value (freeze a value) which is not true for type BOOL.
After the creation, we can check that the variables are perfectly associated with the correct input/output interfaces of the PLC:



	Symbol	Value	Error
0	CLIENT	F1	●
1	LVLMI	F0	●
2	LVLMX	F1	●
3		0	●
4		0	●
5		0	●
6		0	●
7		0	●

Les entrées



	Symbol	Fallback value
16	V1	0
17	V2	0
18		0
19		0
20		0
21		0
22		0
23		0

Les sorties

TAKE CARE ONCE AGAIN ABOUT YOUR OWN REAL CONFIGURATION…

5. Compiling and sending the configuration to the PLC CPU

You have to compile your project (right click on the name of the PLC).
When the compilation is correct, you have to send the program to the CPU of the PLC. Be careful to be rigorous in the setting of the IP addresses!

QUESTION: At the compilation we can have a security "warning" because we have given the access to the web server, why this warning ?

5.1 Debugging and execution mode

We have the possibility to work in debugging mode (mode which allows to see what really happens on the PLC itself during the execution of the programs).

5.2 Writing a program, compiling, sending, executing

We will programme using SFC and ladder (see the specfications in the slides 2. PLC)
Do your programmes (Grafcet and ladder) respecting the specifications of the Lab.

Then you have to perform the following actions:
- compilation,
- send to the CPU,
- Start the execution (running).

6 SCADA interfacing

Generally the HMI will be an industrial touch screen (it can be also a computer based HMI) using the industrial supervisory control software PCVue for instance. Your HMI has to display the values of the sensors, the actuators controls.

Details on PLC programming, digital I/O mapping and HMI interfacing depend on the hardware available for your lab.
See with the teacher which HMI you will use (local or remote and so the way to interact with it).

HMI creation.

To create a new view of the process in PCVue you'll click on "File->New". A SCADA synoptic is called a mimic in PCVue. In the creation dialog select the branch you have previously created and let the template empty. After creation save your new mimic in the Local library.

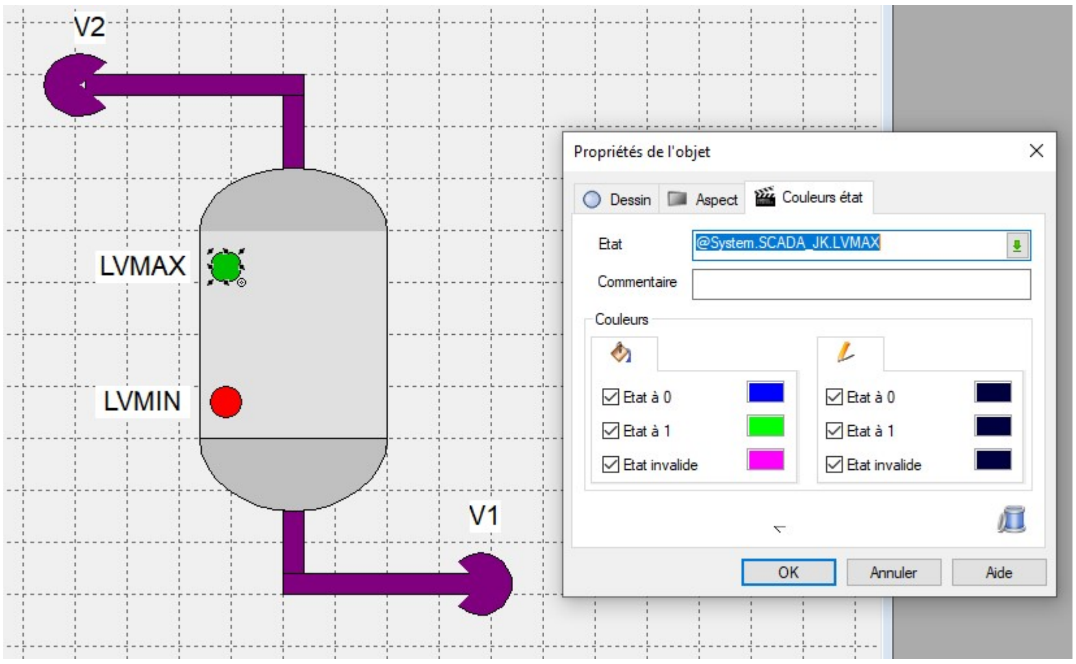
To visualize binary variables add an object to the mimic (a shape from the left tool bar, for example) then right click the object and select "Animate->Color". Then point to the bit variable you want to display. When the mimic is running the object color will change when the binary variable changes. If PCVue shall modify the value go on "Animate->Send->Bit".

To display integer values add a text object to the mimic. Then right click "Animate->Text->Display register".

Supervisory control with PCVue

In order to set-up a supervisory HMI one has to follow three main steps:

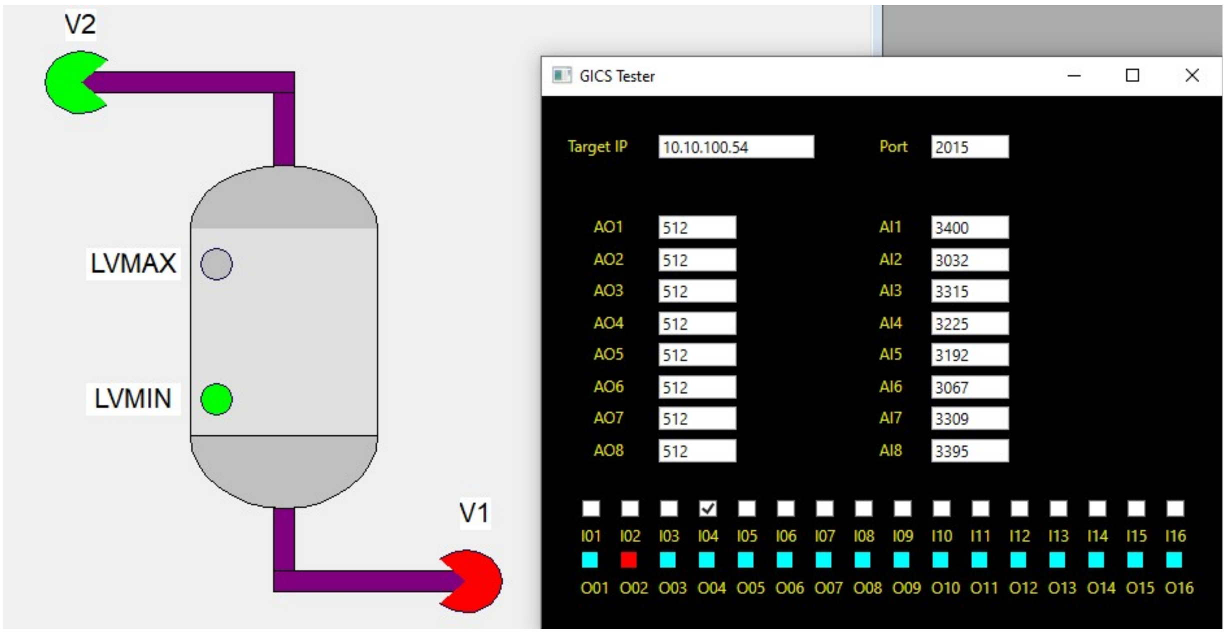
1) Configure the communication with a PLC
2) Declare PCVue variables and link them with the communication
3) Add some controls to the HMI and link variables to controls animation



Etat: @System.SCADA_JK.LVMAX
Commentaire:

Couleurs

☑ Etat à 0	🟦	☑ Etat à 0	⬛
☑ Etat à 1	🟩	☑ Etat à 1	⬛
☑ Etat invalide	🟪	☑ Etat invalide	⬛


V2
LVMAX
LVMIN
V1
GICS Tester

Target IP 10.10.100.54 Port 2015

AO1	512	AI1	3400
AO2	512	AI2	3032
AO3	512	AI3	3315
AO4	512	AI4	3225
AO5	512	AI5	3192
AO6	512	AI6	3067
AO7	512	AI7	3309
AO8	512	AI8	3395

I01 I02 I03 I04 I05 I06 I07 I08 I09 I10 I11 I12 I13 I14 I15 I16
O01 O02 O03 O04 O05 O06 O07 O08 O09 O10 O11 O12 O13 O14 O15 O16

Communication setup (SEE ALSO APPENDIX B)

The communication setup program is started from the menu
“Configure->Communication->Equipment” (see figure B1)

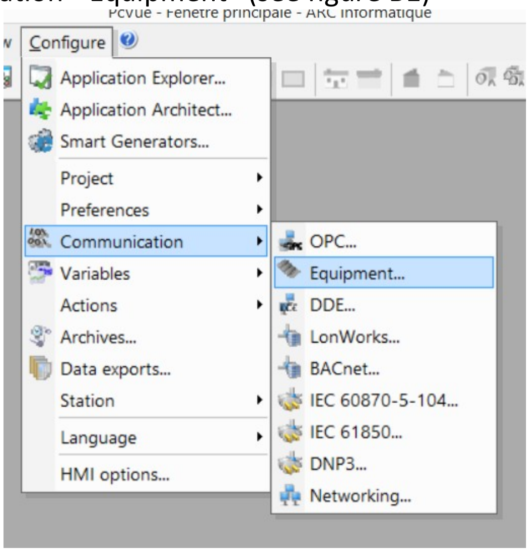


Figure B1: communication menu

Most the parameters you have to configure in the following screens concern the internal variable database structure of PCVue. For the purpose of this lab the names you are choosing are not important (you have only a few equipment and variables) but keep in mind that in general the variables database structure is an important topic.

You’ll have to declare three groups of parameters used to identify the protocol you’ll use, the equipment you'll talk to and the variables you scan (figure B2).

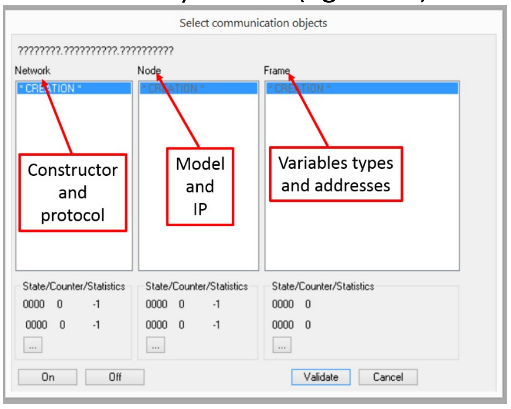


Figure B2: Communication setup

At each step in your configuration you shall click the "Validate" button.

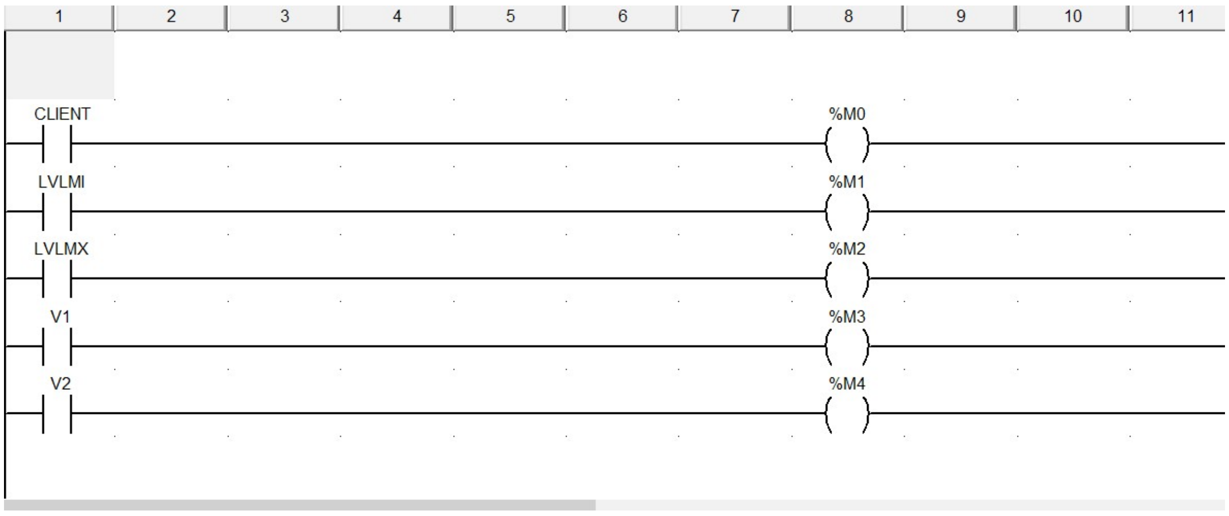
Modbus TCP communication setup

Modbus and Modbus TCP where initially Schneider protocols. Therefore whatever the Modbus device is (Schneider, ABB or Wago) you have to choose "Schneider" and "XBUS-IP-MASTER" for the "Network" creation.

For the "Node" configuration choose "Modbus Hex" (for instance) for "Equipment Type" and add IP number of your equipment into "Network address". Let the other parameters to default.

Into the "Frame" configuration you have to choose the type of variables you want to retrieve (bit/byte/word/etc) and the variables addresses into the PLC memory. Depending of the type (bit, byte word, etc) a different number of addresses types are available. The main categories are:

1) Base objects into the PLC main memory. They correspond to the %M and %MW addresses
2) Extended object: memory objects in modules other than the CPU. This is heavily depending on the PLC type and are mostly compatible with older Schneider devices.
3) Input or Output objects. They correspond to %I (or %IW) respectively %Q (or %QW) objects. **Unfortunately it supposes that the I/O are on the CPU (main module). Which it is not true in most cases.** Because of that, we will have to make correspondences between our IO and new variables in memory. An example is given below.



4) Extended Input or Output objects. Input/output objects in modules other than the CPU. Unfortunately, again, this is heavily depending on the PLC type and are mostly compatible with older Schneider devices.

Eventually, the only reliable transfer mode for Modbus is %M (or %MW) variables scan. Actually, most PLC docs and tutorials strongly advise to avoid direct %I and %Q access from PCVue but copy them to memory object and read memory objects only from PCVue.

An undocumented bug in PCVue Modbus/TCP driver bit transfer function: the driver is unable to handle quantities of data bits other than multiples of 8. The Modbus standard says that the number of transferred bits shall be automatically filled up to the closer multiple of 8, but PCVue driver is unable to do it. If, for instance, you need the value of the first 3 bits starting with %M0, you still have to explicitly ask for 8 bits. Otherwise the driver will not start but no error message is returned (It seems that it is not any more the case with the present version of PCVue but…).

Variables creation.

There are several ways to create variables in PCVue. We'll use the variable selector: menu "Configure -> Variable -> Selector".

First, create a new branch in the variable tree for your application (it will be easier to find your variables in the tree). Then start adding new variables to your system. The main characteristics of a variable are presented in Figure B3.

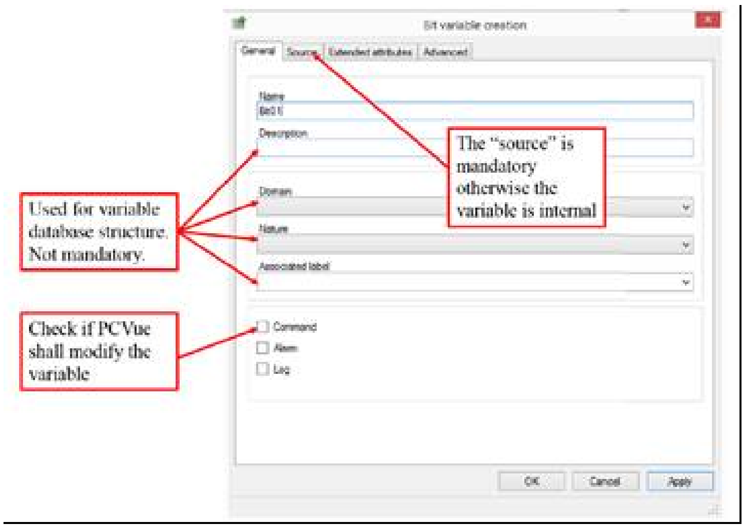

Bit variable creation
General
Source
Extended attributes
Advanced
Name
Description
Domain
Nature
Associated label
Command
Alarm
Log
OK
Cancel
Apply
The "source" is mandatory otherwise the variable is internal
Used for variable database structure. Not mandatory.
Check if PCVue shall modify the variable

Figure B3: Variable creation dialog

Do not forget to click on "source" tab and link the variable to your communication. Otherwise the variable is internal and the communication will not start.

3 Communication analysis.

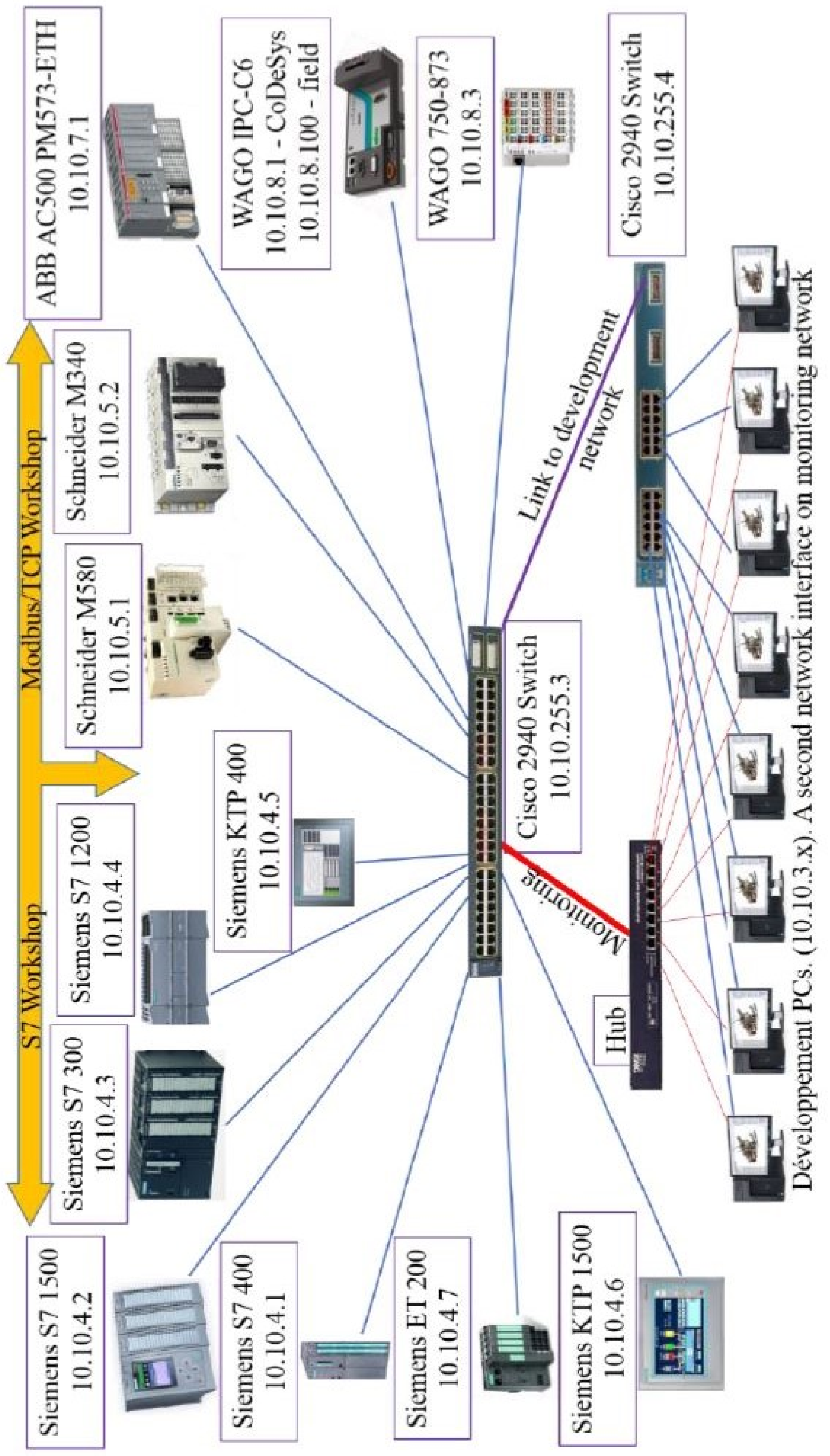
Using Wireshark software on your computer you'll intercept the traffic between the PLC and the SCADA. You're asked to identify the flows (data exchanges) corresponding to every logical flow (variable exchange) in your system. Caution: when you intercept the traffic you'll receive ALL the traffic from the industrial network of the lab, meaning also the traffic from the other groups. You'll need to filter the packets you'll receive.

4 Conclusion

At the end of the lab you shall be able to:
- Understand a practical SCADA communication architecture
- Understand the rationale of some industrial protocols (Modbus/TCP and S7 for instance)
- Identify practically data flows in an industrial network using a network sniffer
- Find the relationship between functional data flows (what your project needs) and physical data flows (what the network really carries)

APPENDIX A. G-ICS network architecture

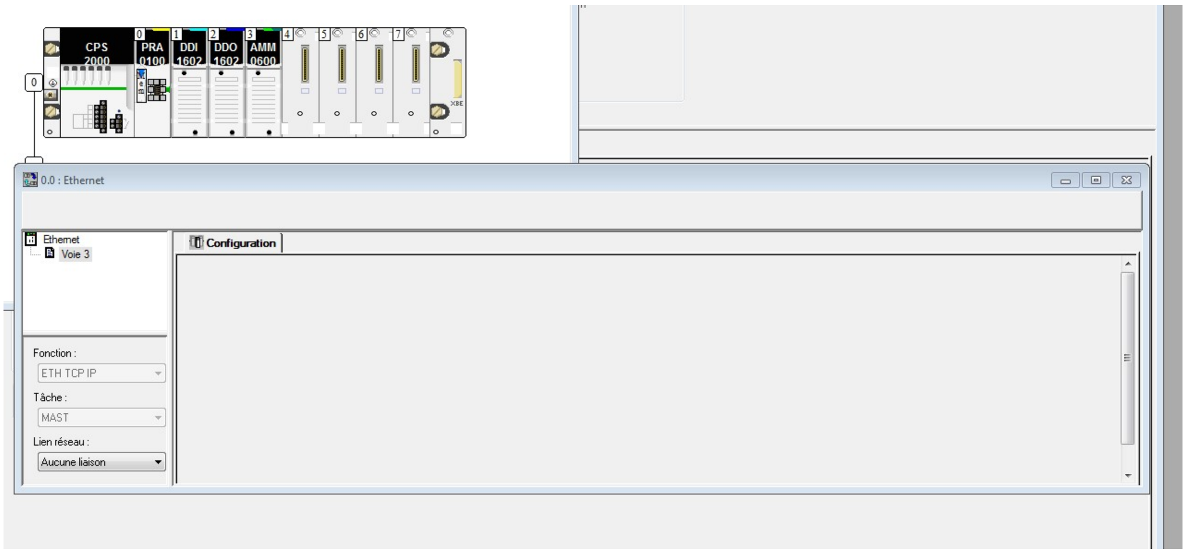
Only the devices used for the lab are included into the diagram



APPENDIX B. Documentation technique Réseau & IHM (SCHNEIDER & Autres automates, voir avec l'enseignant)

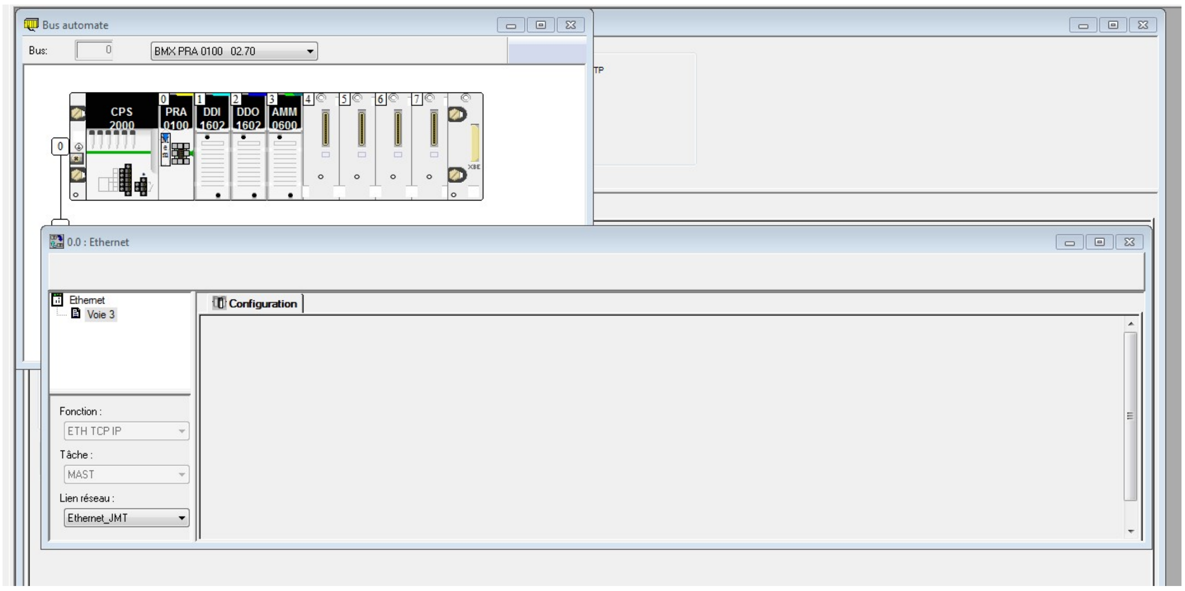
Une fois que vous avez créé votre nœud « réseau », il faudra le lier à l'interface réseau souhaitée de votre automate. Vous pouvez vous inspirer des copies d'écran ci-dessous.

1. Cliquez sur l'interface réseau souhaitée (ici l'interface du PRA 100)



Fonction : ETH TCP IP
Tâche : MAST
Lien réseau : Aucune liaison

2. Etablir un « lien » avec un des nœuds qui a été configuré auparavant.

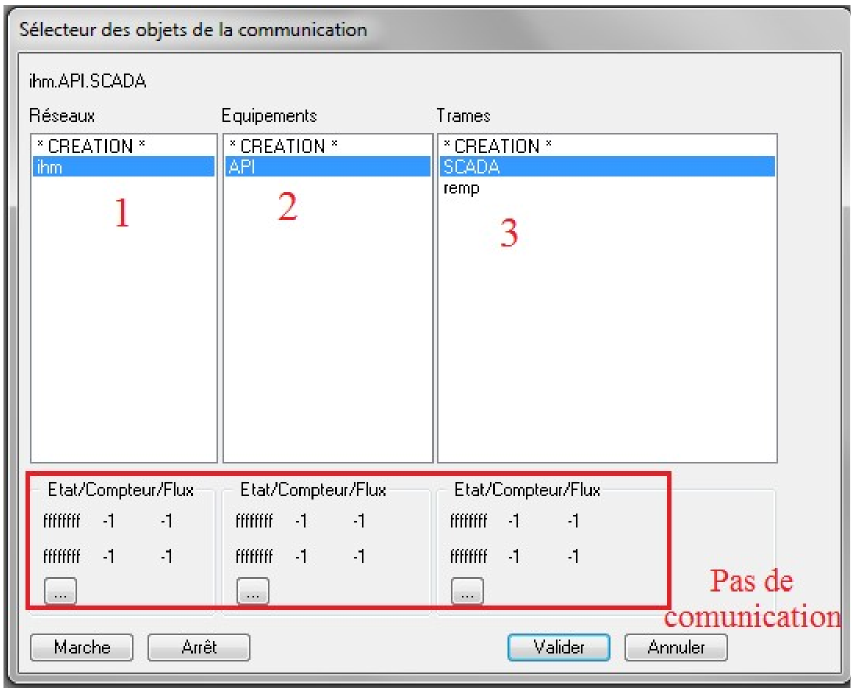


Fonction : ETH TCP IP
Tâche : MAST
Lien réseau : Ethernet_JMT

Documentation technique Réseau & IHM (Schneider)

3. IHM (PcVue)

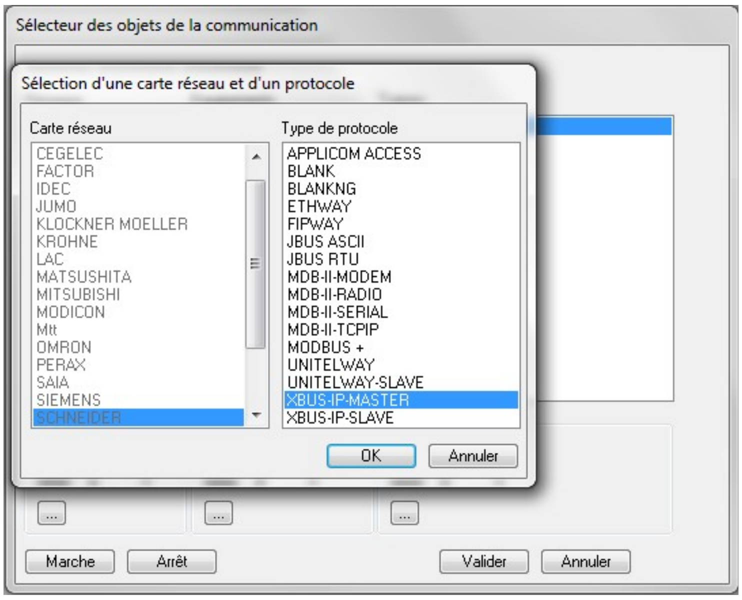
PcVue est utilisé pour les automates Schneider, ABB, WAGO et Siemens 300 et 400.
Dans le menu du logiciel PCVue nous commençons par **configure** → **communication** → **Equipement** pour obtenir le panel de configuration IHM de SCADA



Dans cette figure, les compteurs de flux de la communication n'incrémentent pas parce que la communication n'est pas encore établie.

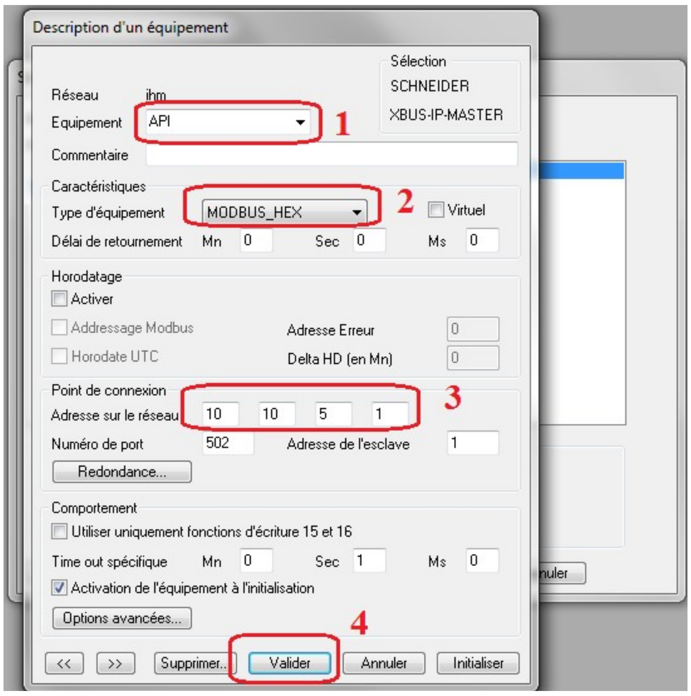
4. Création du réseau Ethernet : SCHNEIDER/ XBUS-IP-MASTER

En cliquant sur **CREATION**, une seconde fenêtre s'ouvre pour sélectionner la carte réseau et le type de protocole.



Il faut ensuite valider puis annuler.

Dans équipements, il faut définir le nom de l'automate ici nommée **PLC**, on l'associe à un port de type MODBUS et on fait la configuration de **l'adresse IP** (attention à bien prendre l'adresse de votre automate).


Description d'un équipement

Sélection: SCHNEIDER, XBUS-IP-MASTER

Réseau: ihm
Equipement: API
Commentaire:

Caractéristiques
Type d'équipement: MODBUS_HEX — Virtuel [ ]
Délai de retournement: Mn 0, Sec 0, Ms 0

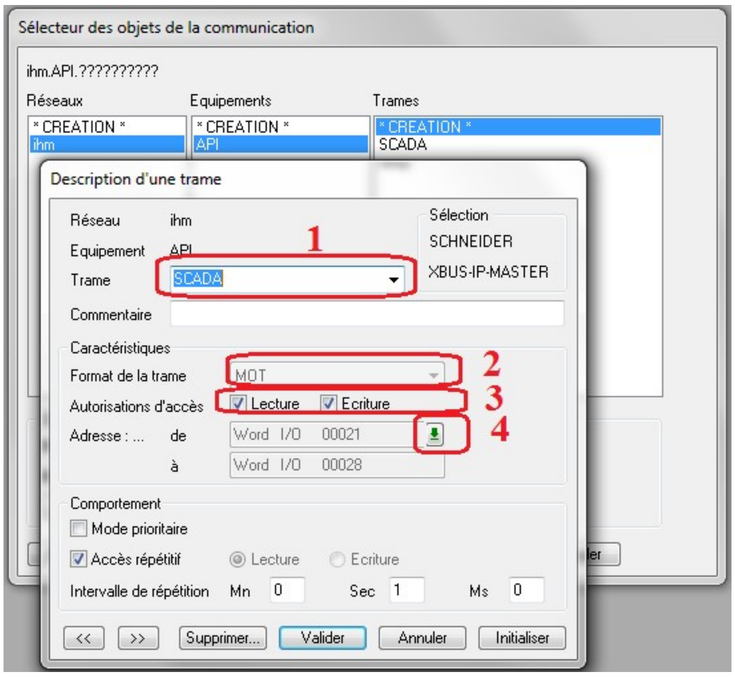
Horodatage
[ ] Activer
[ ] Addressage Modbus — Adresse Erreur: 0
[ ] Horodate UTC — Delta HD (en Mn): 0

Point de connexion
Adresse sur le réseau: 10 10 5 1
Numéro de port: 502 — Adresse de l'esclave: 1

Comportement
[ ] Utiliser uniquement fonctions d'écriture 15 et 16
Time out spécifique: Mn 0, Sec 1, Ms 0
[✓] Activation de l'équipement à l'initialisation

5. Création d'une trame de bits en lecture et écriture

Dans la colonne Trames faire un double-clic sur **CREATION** puis on nomme notre trame ici « lecture/écriture /mot » au format bit avec une autorisation en lecture et écriture. La trame créée sera lue de façon cyclique toutes les secondes.


Sélecteur des objets de la communication

ihm.API.?????????

Réseaux	Equipements	Trames
* CREATION *	* CREATION *	* CREATION *
ihm	API	SCADA

Description d'une trame

Sélection: SCHNEIDER, XBUS-IP-MASTER

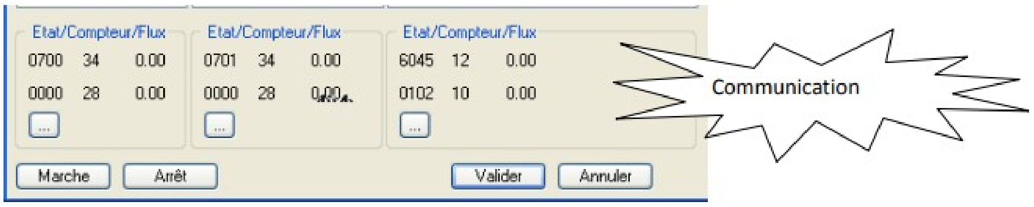
Réseau: ihm
Equipement: API
Trame: SCADA
Commentaire:

Caractéristiques
Format de la trame: MOT
Autorisations d'accès: [✓] Lecture [✓] Ecriture
Adresse : ... de Word I/O 00021
à Word I/O 00028

Comportement
[ ] Mode prioritaire
[✓] Accès répétitif — ○ Lecture ○ Ecriture
Intervalle de répétition: Mn 0, Sec 1, Ms 0

Il faut ensuite définir la taille de la trame, c'est-à-dire le nombre de données ou d'éléments binaire qui seront lu dans l'automate. Il faut aussi préciser dans quelle zone de l'automate (ou adresse) les données seront lues. Pour cela on clique sur la flèche verte et on définit l'adresse de départ, et l'adresse de fin de la trame à lire dans l'automate. Ici c'est une trame de MOT en lecture/écriture

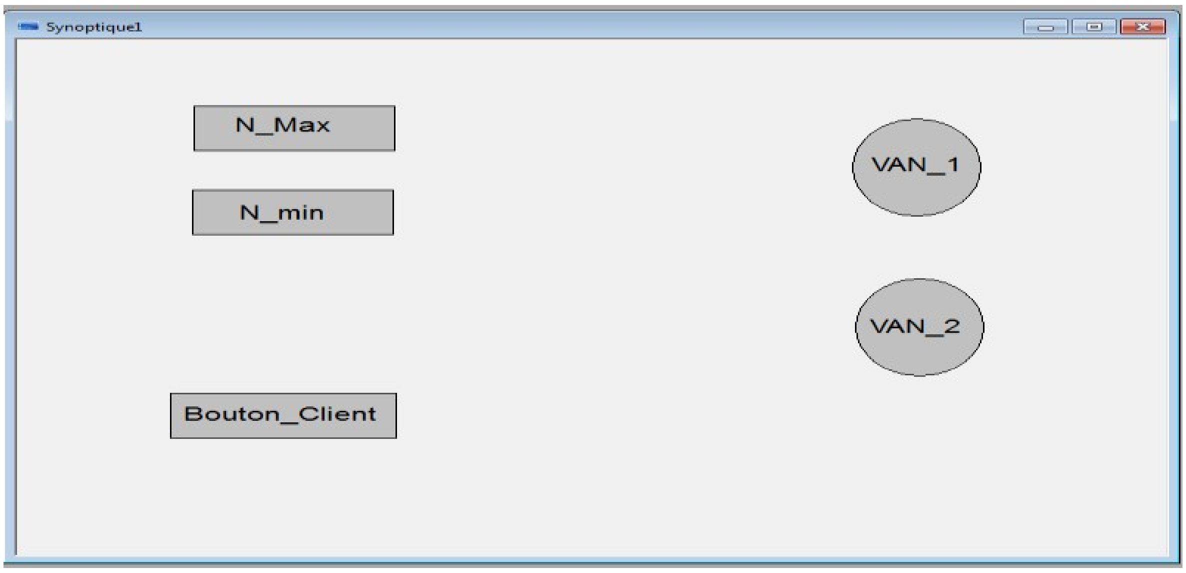
d’une longueur de 16 éléments (octets), qui commence à l’adresse de l’automate 21 et qui finira à l’adresse 28.
Une fois que la communication est paramétrée, il ne reste plus qu’à activer la communication avec le bouton Marche. Si la communication est disponible avec l’automate les compteurs de flux de la communication doivent s’incrémenter.



Etat/Compteur/Flux			Etat/Compteur/Flux			Etat/Compteur/Flux		
0700	34	0.00	0701	34	0.00	6045	12	0.00
0000	28	0.00	0000	28	0.00	0102	10	0.00

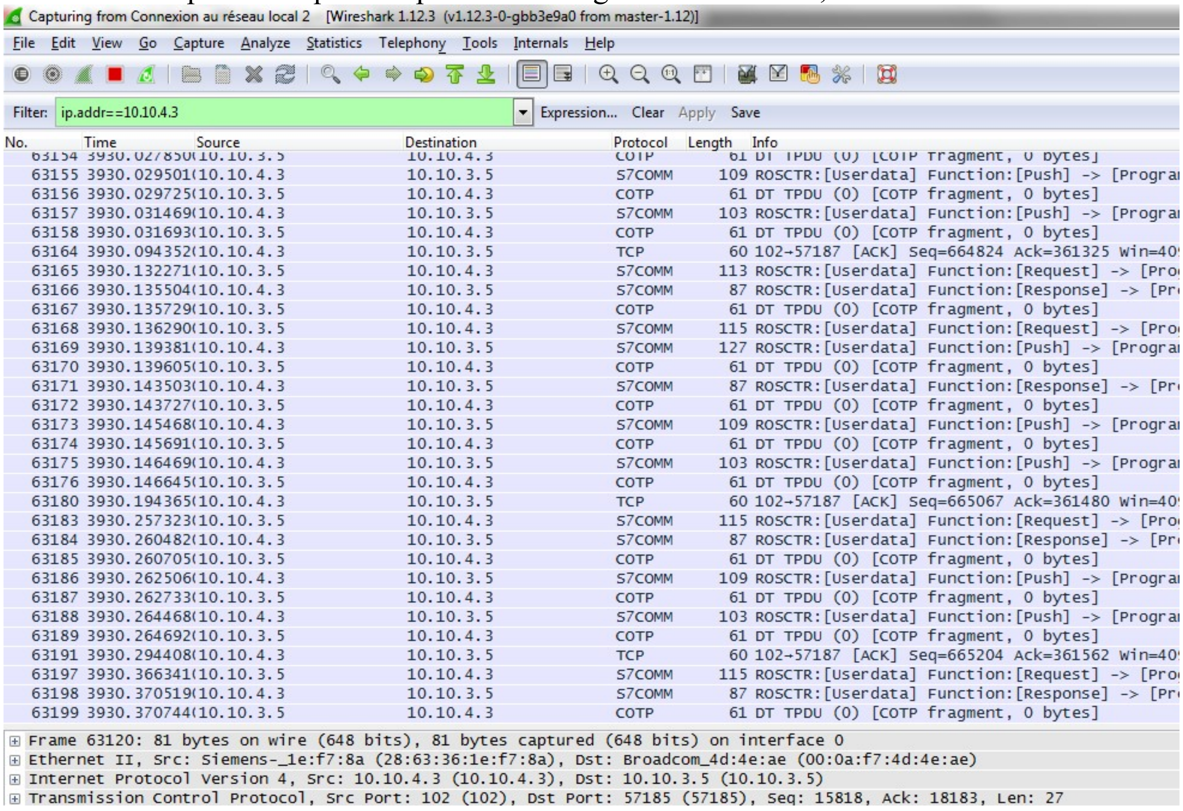
6. Création d’une trame de bits en lecture et écriture

L’interface homme-machine



ASPECT WIRESHARK

Voici un exemple de ce que l'on peut observer grâce à Wireshark, sur l'adresse 10.10.4.3



```
Capturing from Connexion au réseau local 2   [Wireshark 1.12.3  (v1.12.3-0-gbb3e9a0 from master-1.12)]

Filter: ip.addr==10.10.4.3

No.     Time          Source              Destination        Protocol  Length  Info
 63154 3930.027850(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63155 3930.029501(10.10.4.3          10.10.3.5          S7COMM     109 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63156 3930.029725(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63157 3930.031469(10.10.4.3          10.10.3.5          S7COMM     103 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63158 3930.031693(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63164 3930.094352(10.10.4.3          10.10.3.5          TCP         60 102→57187 [ACK] Seq=664824 Ack=361325 Win=40
 63165 3930.132271(10.10.3.5          10.10.4.3          S7COMM     113 ROSCTR:[Userdata] Function:[Request] -> [Pro
 63166 3930.135504(10.10.4.3          10.10.3.5          S7COMM      87 ROSCTR:[Userdata] Function:[Response] -> [Pr
 63167 3930.135729(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63168 3930.136290(10.10.3.5          10.10.4.3          S7COMM     115 ROSCTR:[Userdata] Function:[Request] -> [Pro
 63169 3930.139381(10.10.4.3          10.10.3.5          S7COMM     127 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63170 3930.139605(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63171 3930.143503(10.10.4.3          10.10.3.5          S7COMM      87 ROSCTR:[Userdata] Function:[Response] -> [Pr
 63172 3930.143727(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63173 3930.145468(10.10.4.3          10.10.3.5          S7COMM     109 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63174 3930.145691(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63175 3930.146469(10.10.4.3          10.10.3.5          S7COMM     103 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63176 3930.146645(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63180 3930.194365(10.10.4.3          10.10.3.5          TCP         60 102→57187 [ACK] Seq=665067 Ack=361480 Win=40
 63183 3930.257323(10.10.3.5          10.10.4.3          S7COMM     115 ROSCTR:[Userdata] Function:[Request] -> [Pro
 63184 3930.260482(10.10.4.3          10.10.3.5          S7COMM      87 ROSCTR:[Userdata] Function:[Response] -> [Pr
 63185 3930.260705(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63186 3930.262506(10.10.4.3          10.10.3.5          S7COMM     109 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63187 3930.262733(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63188 3930.264468(10.10.4.3          10.10.3.5          S7COMM     103 ROSCTR:[Userdata] Function:[Push] -> [Prograı
 63189 3930.264692(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]
 63191 3930.294408(10.10.4.3          10.10.3.5          TCP         60 102→57187 [ACK] Seq=665204 Ack=361562 Win=40
 63197 3930.366341(10.10.3.5          10.10.4.3          S7COMM     115 ROSCTR:[Userdata] Function:[Request] -> [Pro
 63198 3930.370519(10.10.4.3          10.10.3.5          S7COMM      87 ROSCTR:[Userdata] Function:[Response] -> [Pr
 63199 3930.370744(10.10.3.5          10.10.4.3          COTP        61 DT TPDU (0) [COTP fragment, 0 bytes]

⊞ Frame 63120: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
⊞ Ethernet II, Src: Siemens-_1e:f7:8a (28:63:36:1e:f7:8a), Dst: Broadcom_4d:4e:ae (00:0a:f7:4d:4e:ae)
⊞ Internet Protocol Version 4, Src: 10.10.4.3 (10.10.4.3), Dst: 10.10.3.5 (10.10.3.5)
⊞ Transmission Control Protocol, Src Port: 102 (102), Dst Port: 57185 (57185), Seq: 15818, Ack: 18183, Len: 27
```