Industrial communication labs (Schneider)

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) Understand the communication between PLC and HMI/SCADA
2) Implement and observe communication flows
3) Analyze the flows, optimize the communication

1. 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.

1.1 Réseau : Le réseau 10.10.0.0/16 interconnecte différents éléments :

- les API ,
- les ordinateurs de « configuration » (10.10.3.x),
- les cartes « simulateurs de processus » (10.10.100.x, non visible sur le schéma général de l'Annexe A).
- les interfaces Homme-Machine (HMI).

Ces éléments sont interconnectés via deux switches CISCO.

Architecture de commande et architecture de supervision

Bien comprendre et préciser le fonctionnement global de l'architecture et de la boucle de commande. Cette architecture est constituée de :

Elément de l'architecture par groupe pour la commande	Interfaces
Un API destiné à recevoir la commande	- Entrées/Sorties analogiques et numériques - Carte réseau - Cas particulier : 10.10.4.1 : E/S déportées par l'interface ET200S de gauche via Profibus (celui sans adresse IP…)
Un « processus physique » représenté par une carte de simulation (via l'adresse 10.10.100.x) embarquée SMT32.	- Entrées/Sorties analogiques et numériques - Carte réseau
Une interface de commande à distance (GICS Tester) implémentée sur la machine de « configuration » (10.10.3.x)	- Carte réseau

QUESTION : Dessiner la boucle de commande avec le contrôleur, les actionneurs, les mesures, l'action de demande de vidange (client_request) et préciser les interfaces utilisées et les liens/modes de « connexions » (ex : réseau(x), lien direct E/S…)

Au-delà de la boucle de commande, l'architecture sert également à la configuration.

1.2 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...

1.3 Prise en main de votre espace de travail

Sur chaque ordinateur un compte local est ouvert pour les étudiants (ex MISTREA01, vous devez demander à l'enseignant quel est votre login et mot de passe sur votre machine, et bien le noter).
Les logiciels utiles sont :

- Les logiciels pour configurer votre API : TIA Portal pour Siemens et Unity Pro XL pour Schneider
- Une interface de commande à distance GICS Tester

QUESTION : Quelle est l'utilité de l'interface de contrôle GICS Tester ?

1.4 Configuration de l'interface de commande à distance GICS Tester

Sur l'interface de contrôle GICS Tester (ordinateur de « configuration »), il faut configurer l'adresse IP de la carte de simulation que vous utilisez « target IP » : 10.10.100.x, sur le port 2015. Nous n'utiliserons pas les entrées et sorties analogiques (les valeurs numériques sont comprises entre 0 et 4095 pour des tensions comprises entre -10 et +10 V).
A partir de là, vous pouvez tester les sorties en les forçant par l'interface de contrôle GICS Tester et en observant leurs variations sur les interfaces physiques des cartes E/S.

NE PAS TOUCHER LES CABLAGES PHYSIQUES.

Si besoin, faites la correspondance via un tableau entre les numéros des E/S sur l'interface physique et les numéros sur l'interface de contrôle GICS Tester (décalage éventuel).

2. Configuration of the PLC

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

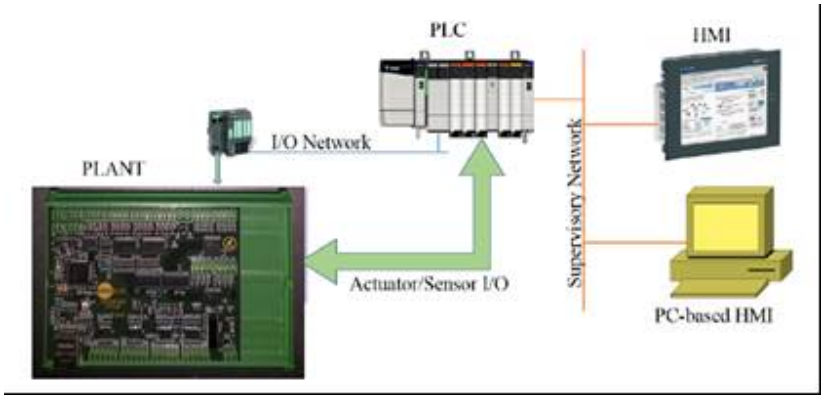


Figure 1. The simple SCADA system

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.

Work to do.

This job does not involve communication between various PLCs, but involves just your own PLC.
You'll have to:

A. Write a very simple control program for the PLC (the objective of the lab is communication, not PLC programming).

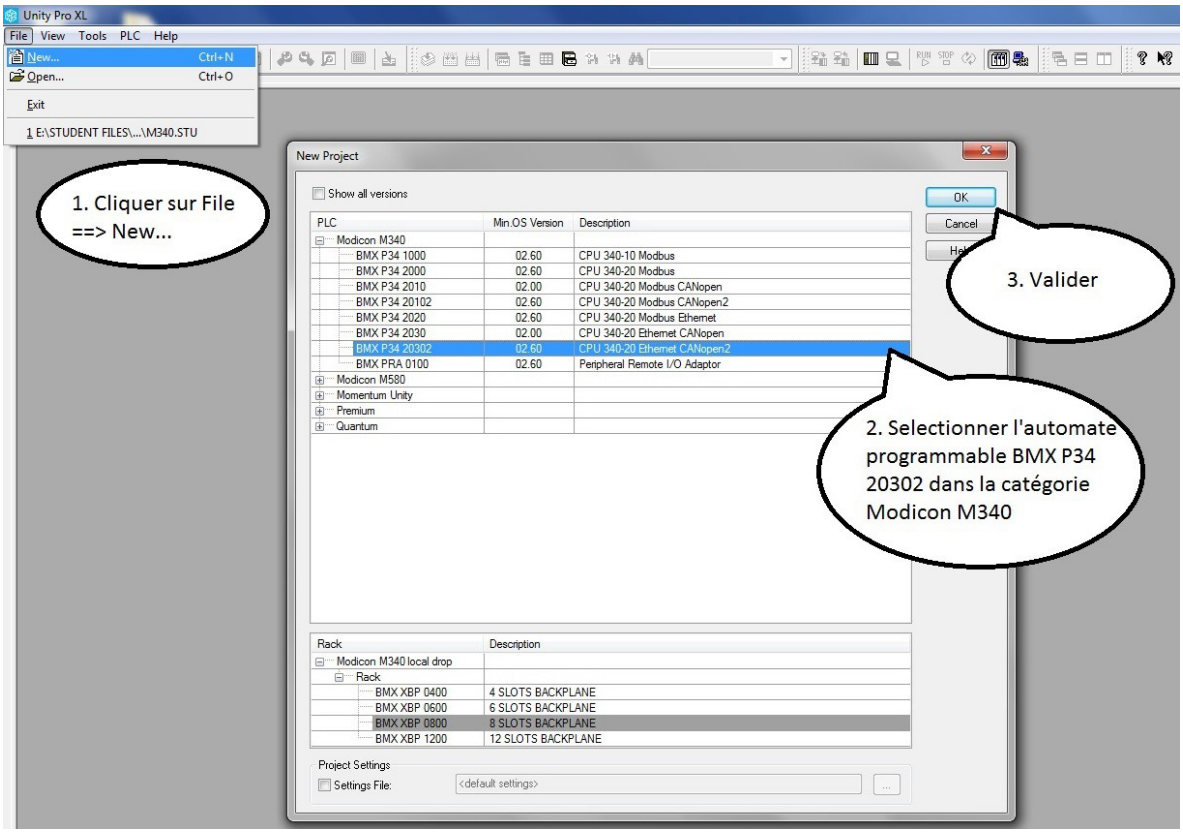
B. Set-up the communication between the SCADA and the PLC

C. Check the communication flows using a network sniffer

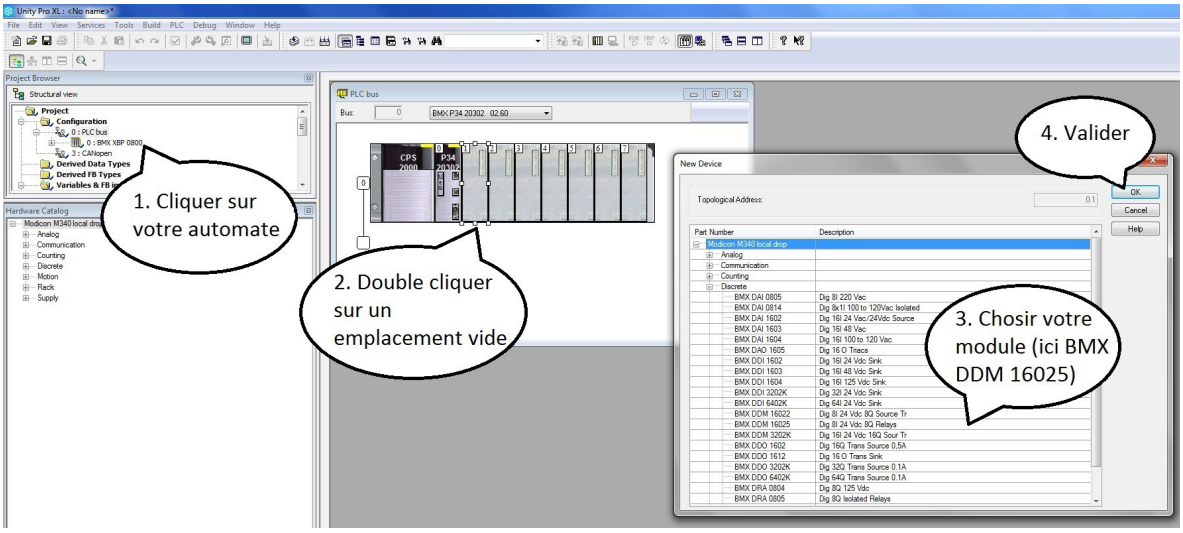
2.1 Projet de configuration de votre API

Sur l'ordinateur de « configuration », vous allez configurer votre API (selon les API et les environnements, ces opérations peuvent être plus ou moins automatisées …), l'objectif est que vous le fassiez pour bien voir et comprendre les différentes étapes. C'est une opération où il faut être rigoureux et précautionneux, choisir les bons éléments et bien les configurer, car ensuite les drivers correspondants seront envoyés à l'automate… et doivent correspondre aux éléments précis de l'architecture physique de l'automate. Il faut tout configurer y compris les unités que vous n'utiliserez pas (par exemple les E/S analogiques) car les configurations logicielle et matérielle doivent être conformes.

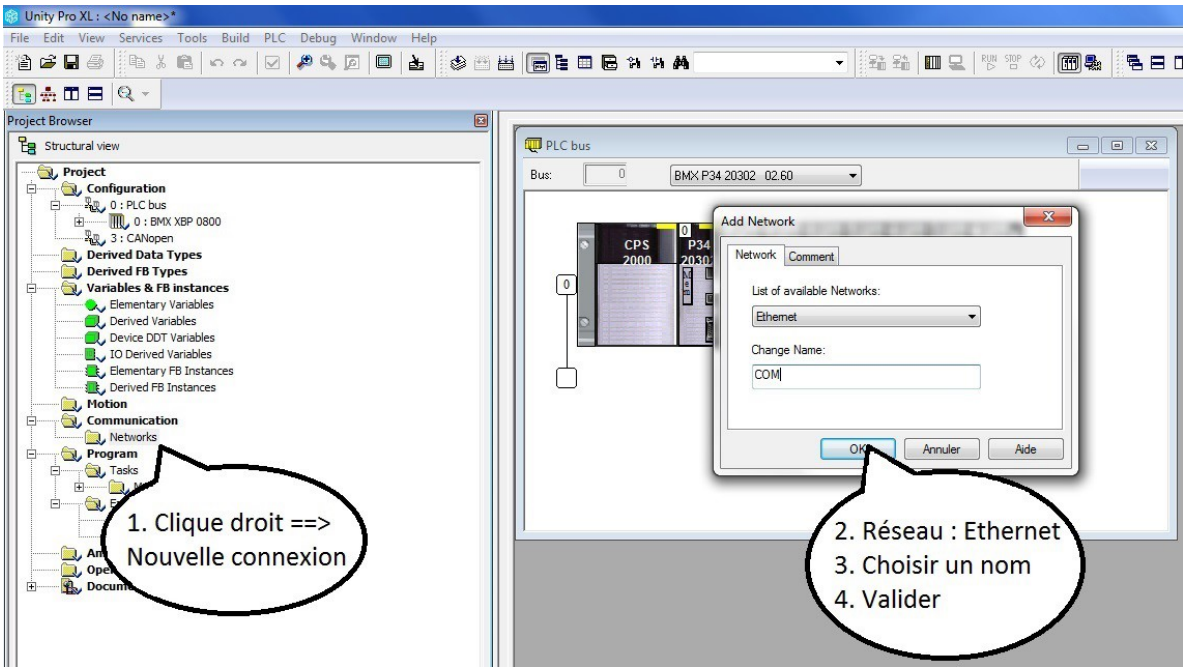
Les figures ci-dessous concernent l'API Schneider de la famille Modicon M340 . Configuration sous Unity Pro XL .

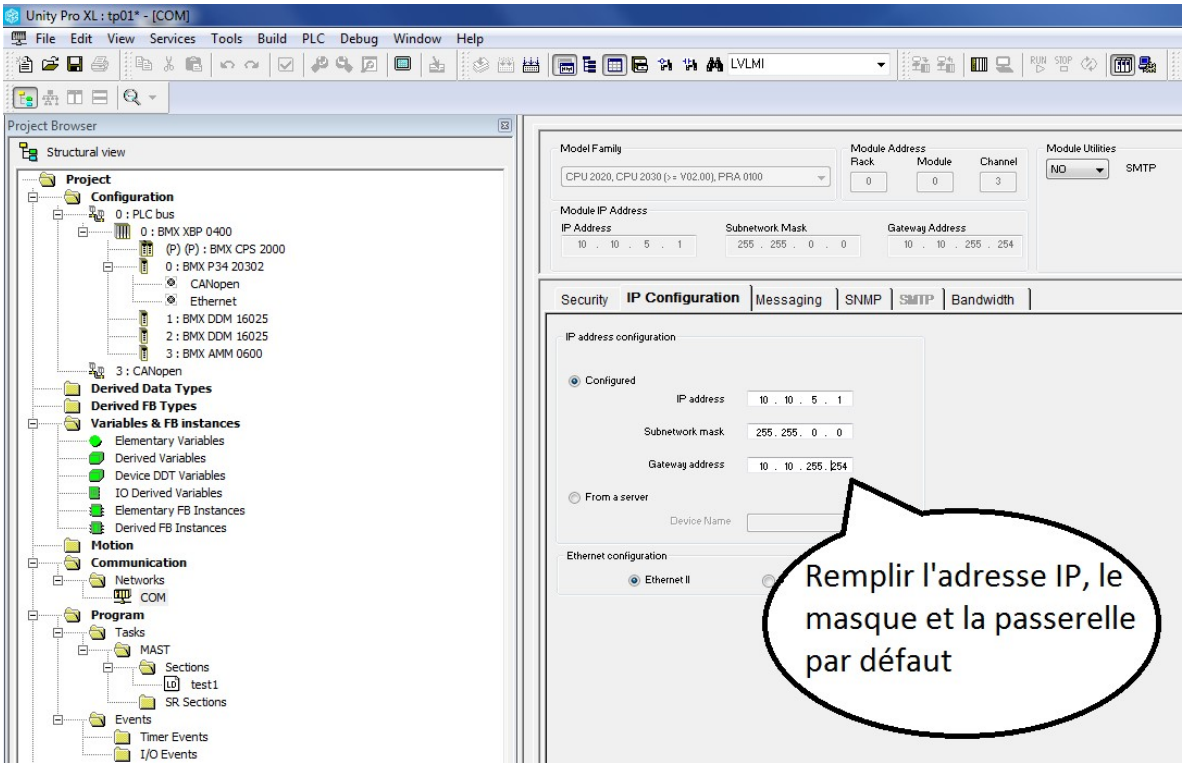


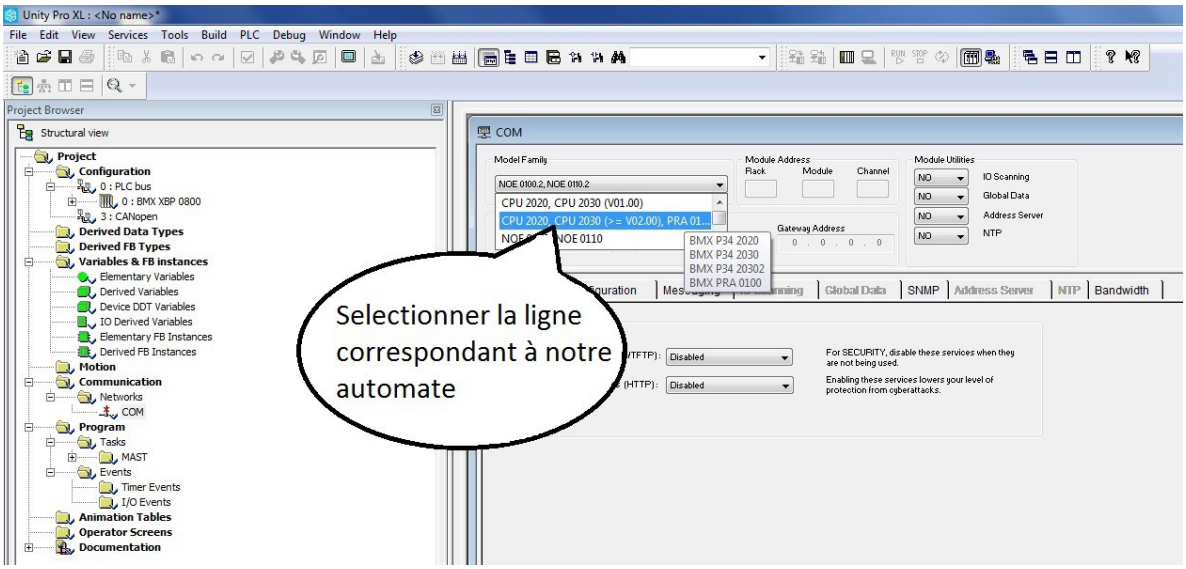
ATTENTION : NOUS POUVONS AUSSI UTILISER UN BMX PRA 100…



2.2 Configuration de l'interface réseau de l'automate, synchronisation d'horloge


Unity Pro XL : <No name>*
Project Browser
Structural view
Project
Configuration
0 : PLC bus
0 : BMX XBP 0800
3 : CANopen
Derived Data Types
Derived FB Types
Variables & FB instances
Elementary Variables
Derived Variables
Device DDT Variables
IO Derived Variables
Elementary FB Instances
Derived FB Instances
Motion
Communication
Networks
Program
Tasks
1. Clique droit ==>
Nouvelle connexion
PLC bus
Bus:
BMX P34 20302 02.60
Add Network
Network
Comment
List of available Networks:
Ethernet
Change Name:
COM
OK
Annuler
Aide
2. Réseau : Ethernet
3. Choisir un nom
4. Valider

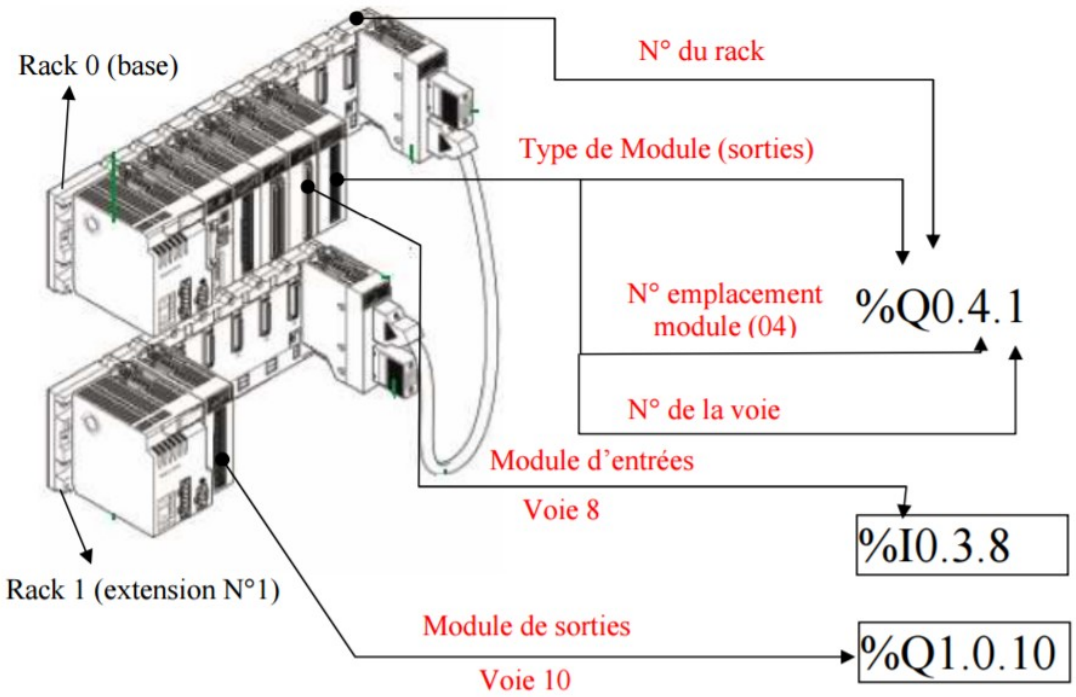

Unity Pro XL : tp01* - [COM]
File Edit View Services Tools Build PLC Debug Window Help
LVLMI
Project Browser
Structural view
Project
Configuration
0 : PLC bus
0 : BMX XBP 0400
(P) (P) : BMX CPS 2000
0 : BMX P34 20302
CANopen
Ethernet
1 : BMX DDM 16025
2 : BMX DDM 16025
3 : BMX AMM 0600
3 : CANopen
Derived Data Types
Derived FB Types
Variables & FB instances
Elementary Variables
Derived Variables
Device DDT Variables
IO Derived Variables
Elementary FB Instances
Derived FB Instances
Motion
Communication
Networks
COM
Program
Tasks
MAST
Sections
test1
SR Sections
Events
Timer Events
I/O Events
Model Family
CPU 2020, CPU 2030 (>= V02.00), PRA 0100
Module Address
Rack
Module
Channel
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
Security
IP Configuration
Messaging
SNMP
SMTP
Bandwidth
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
Device Name
Ethernet configuration
Ethernet II
Remplir l'adresse IP, le masque et la passerelle par défaut



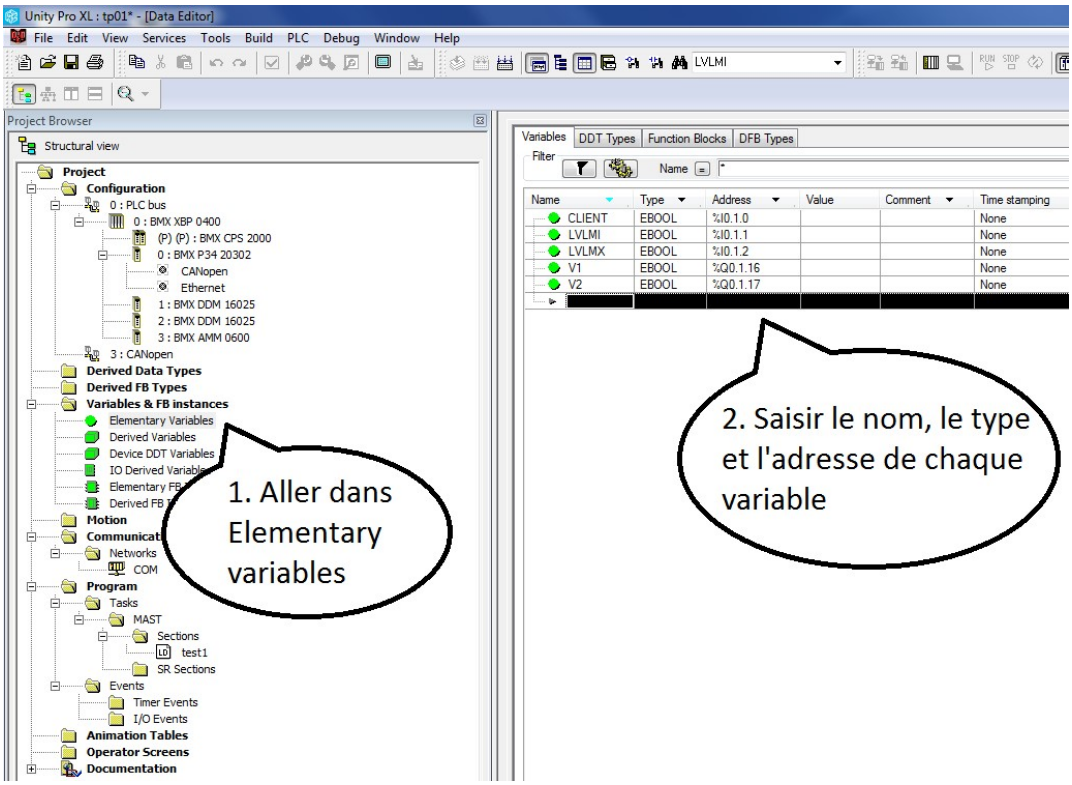
2.3 Configuration des variables

Configuration des variables

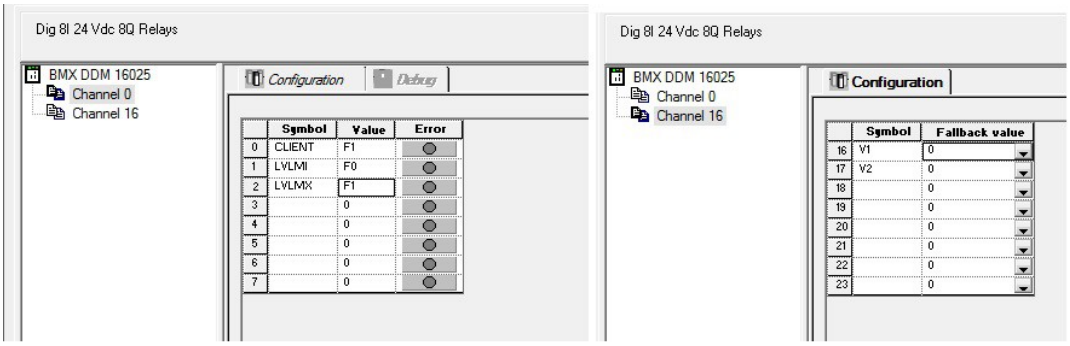
Après la configuration de l'API, il faut définir les variables dont l'API a besoin. Pour ajouter une variable, il faut lui donner un nom, définir son type et connaître l'adresse qui lui correspond sur les ports de modules I/O de l'API. Pour définir cette dernière on peut utiliser la méthode suivante :



Il faut ensuite créer les variables (exemple ci-après) :



Les variables créées sont du type EDT qui est l'abréviation de Elementary Data Type (type de données de base), qui comprend des types de données simples plutôt que dérivées (tableaux, structures, blocs fonctions). Les types EDT disponibles sont les suivants :BOOL,EBOOL, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, DATE, TOD et DT. Dans notre cas nous utiliserons des variables de type EBOOL car ce type nous offre la possibilité de forcer la valeur voulue (figer une valeur) ce qui n'est pas vrai pour le type BOOL.
Après la création, nous pouvons vérifier que les variables sont parfaitement associées aux interfaces des entrée sortie de l'API :



Les entrées **Les sorties**

2.4 Compilation et envoi de la configuration à la CPU de l'automate

Il faut compiler votre projet (click droit sur le nom de l'automate).
Lorsque la compilation est correcte, il faut envoyer le programme dans la CPU de l'automate.
Attention à bien être rigoureux dans la mise en place des adresses IP !

QUESTION : A la compilation nous pourrons avoir un « warning » de sécurité car nous avons donné l'accès au serveur web, pourquoi ce warning ?

2.5 Mode debugging et exécution

On a la possibilité de se mettre en mode déboguage (mode qui permet de voir ce qui se passe réellement sur l'API lui-même pendant l'exécution des programmes).

2.6 Ecriture d'un programme, compilation, envoi, exécution

On peut programmer en ladder par exemple.

Faites votre programme respectant les spécifications du sujet (voir partie 3 page suivante).

QUESTION : Expliciter dans votre compte-rendu votre compréhension des spécifications, faire des choix technologiques (vannes normalement ouvertes, vannes normalement fermées…) puis expliquer votre programme. **N.B. :** Si vous êtes en retard, ne perdez pas trop de temps en séance, faites des programmes simples et testez-les sur la machine.

Il faut ensuite effectuer les actions suivantes :

- compilation,
- envoi à la CPU,
- Lancer l'exécution.

3. The control problem.

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).

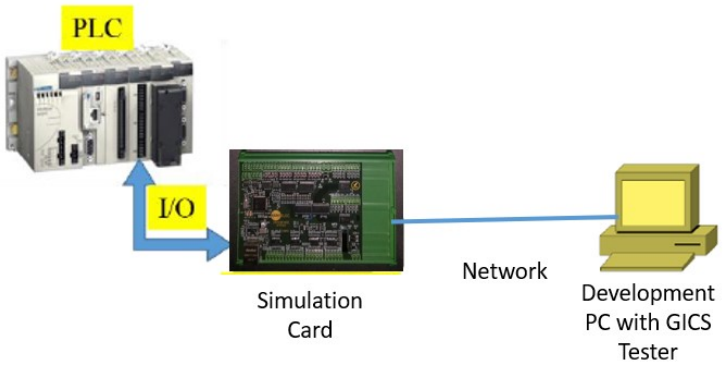


Figure 2. Hardware in the loop bench

The (very simple) control problem that you have to solve is the following. Consider a fluid tank (Figure 3) equipped with two level digital sensors (minimal and maximal levels) and two actuators (the two valves). There is also an external signal ("client request" for fluid supply). Your program has then three digital inputs and two digital outputs.

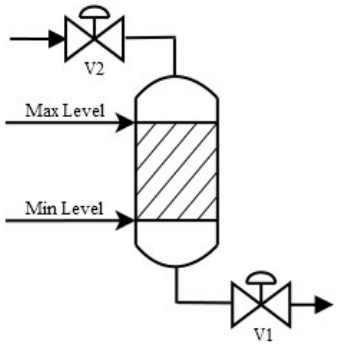


Figure 3. A simple tank

Control specification.

Your main objective is to keep the fluid level between the min and max levels. You'll also serve the client requests for the fluid supply. The control logic specification is:
A. If the tank level is between min and max and a client request is raised open the valve V1 as long as the client request is active.
B. If the tank level reaches level min (that will necessary happen when V1 is open) close V1 and open V2 until the tank level reaches level max. When the tank level reaches max close V2.

OPTIONAL: Supplementary diagnostic tasks: if level min and max are simultaneously activated (sensors failure) or level min is active raise an error (set an error bit in the PLC memory) write the error code (0x01) in a memory word and wait for an external error acknowledgement before going back to the normal state.

4. 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 two level sensors, the actuators controls, the error bit and error code from the PLC and a control allowing for error acknowledgement.

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

Communication setup (SEE ALSO APPENDIX B)

The communication setup program is started from the menu

"Configure->Communication->Equipment" (see figure B1)

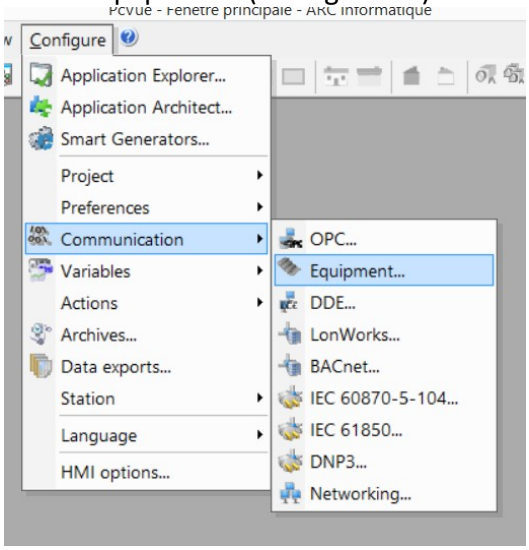


Figure B1: communication menu

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).

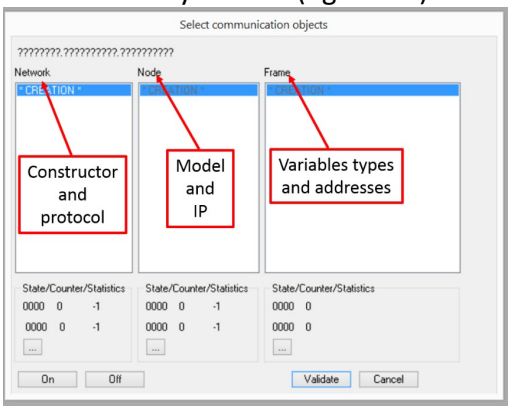


Figure B2: Communication setup

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 "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.
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.

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.

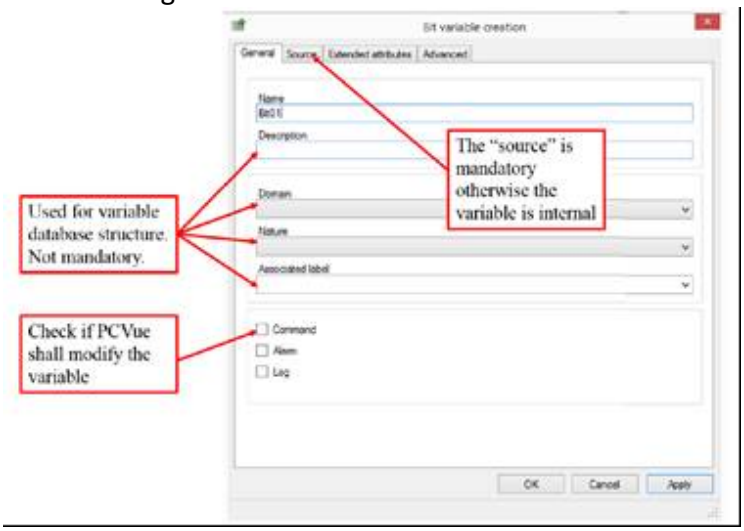


Figure B3: Variable creation dialog

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.

5. 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.

QUESTIONS : Sur la plate-forme, les éléments sont interconnectés via deux switches CISCO.

Comment fonctionne un switch ? Quel est son rôle ?

L'architecture décrite Annexe A propose un hub avec un lien de monitoring sur lequel sont reliées toutes les machines de supervision (10.10.3.x).

Comment fonctionne un hub ? Quel est le rôle d'un hub de manière générale ?

Quel est le rôle du hub dans ce schéma ?

Pour sortir sur internet, ce réseau dispose d'une passerelle (gateway), non visible sur le schéma général. Cette passerelle utilise l'adresse IP généralement utilisée pour les passerelles.

Quelle est donc cette adresse ?

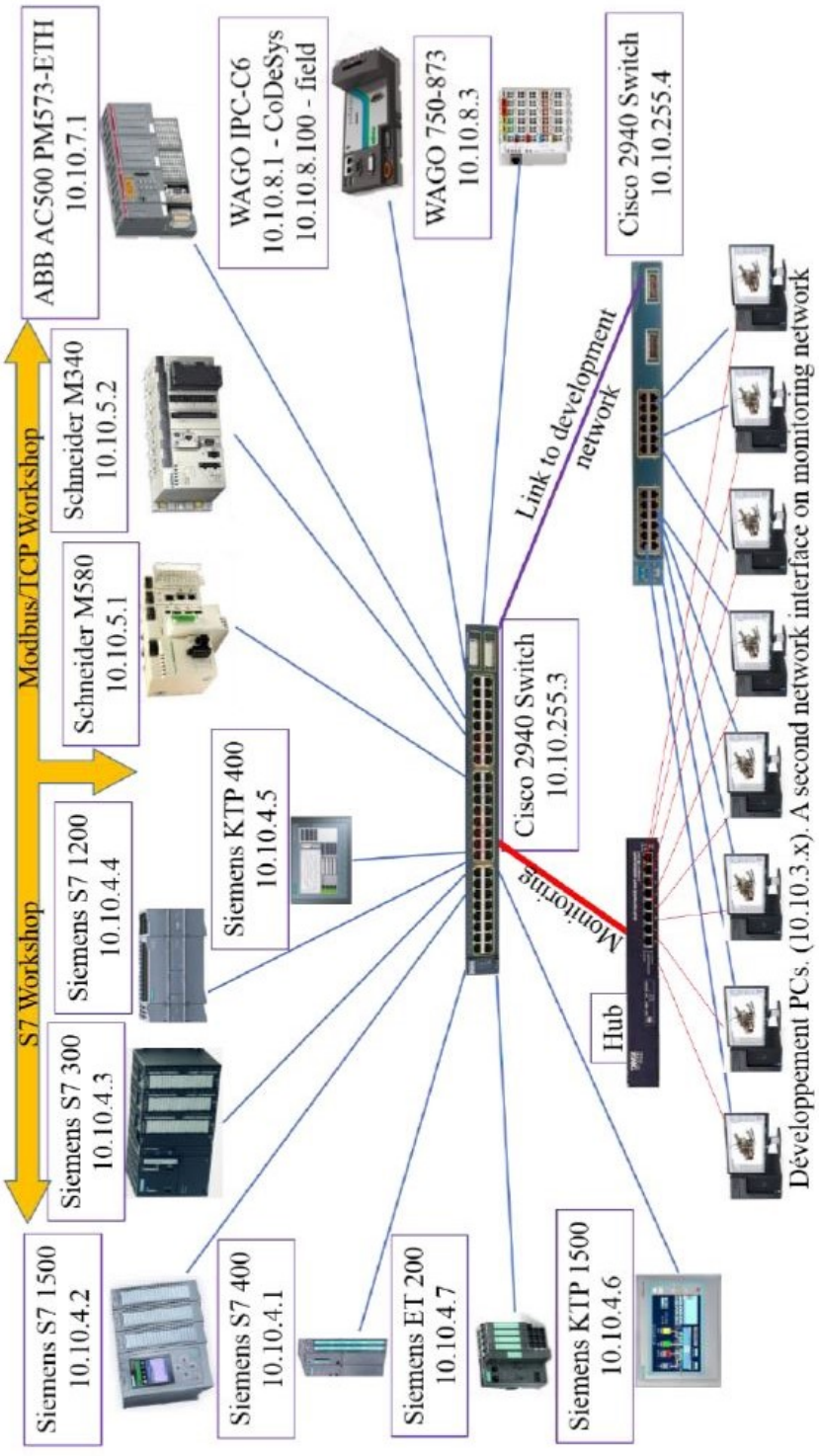
6. Conclusion

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

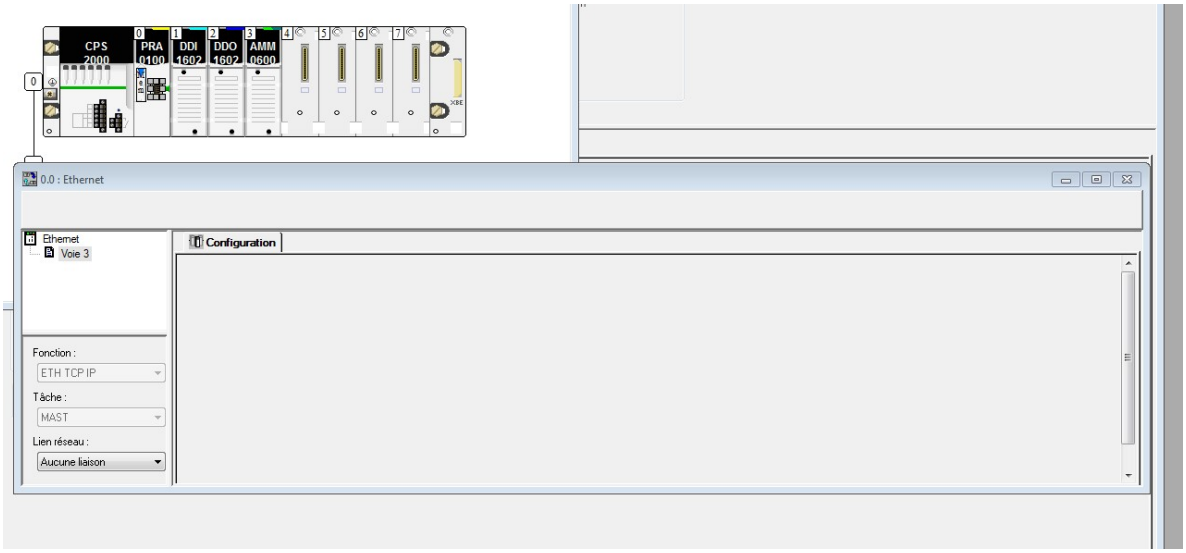
Only the devices used for the lab are included into the diagram



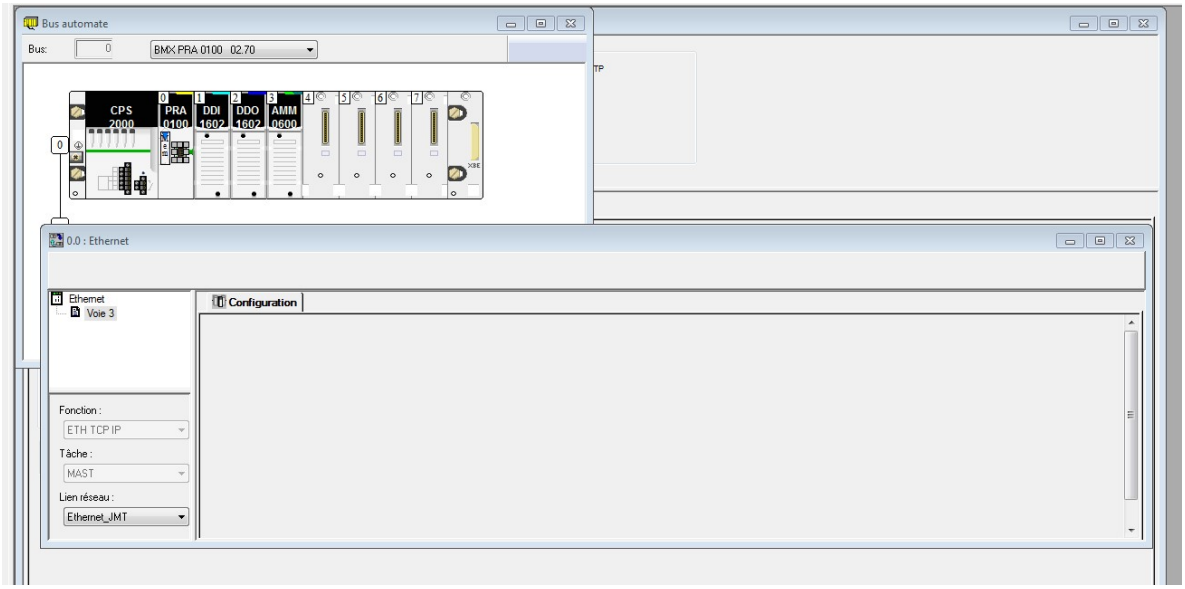
APPENDIX B. Documentation technique Réseau & IHM (SCHNEIDER & Autres automates, voir avec l'enseignant)

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)



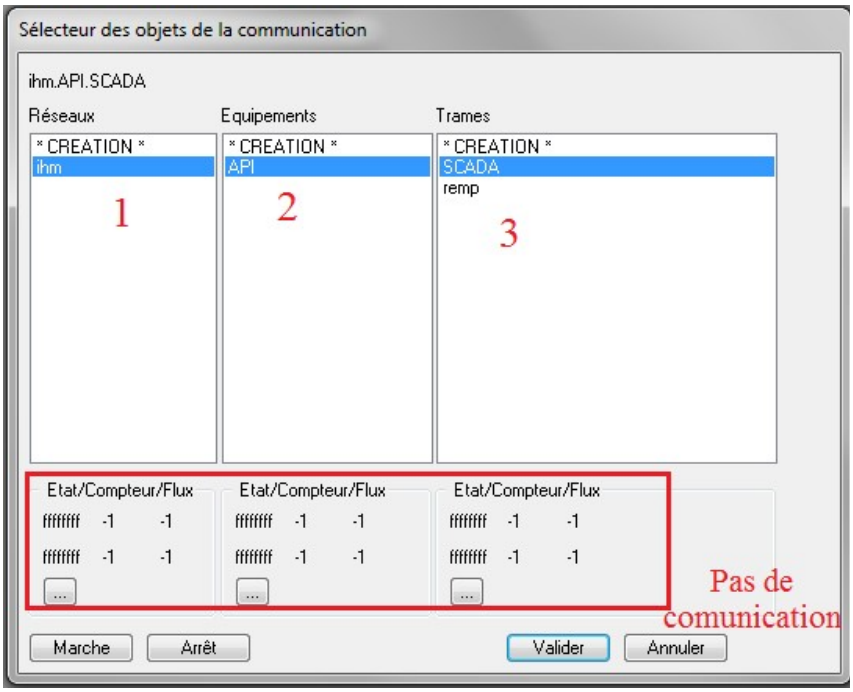
2. Etablir un « lien » avec un des nœuds qui a été configuré auparavant.



Documentation technique Réseau & IHM (Schneider)

3. IHM (PcVue)

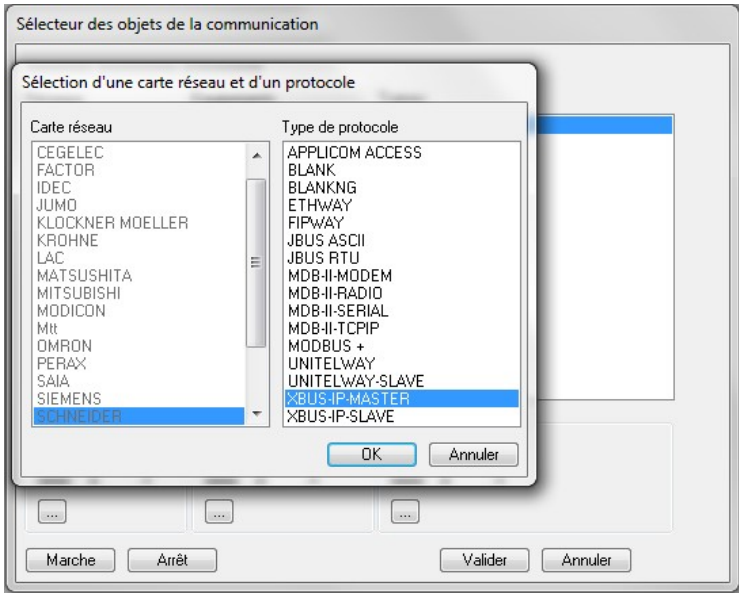
PcVue est utilisé pour les automates Schneider, ABB, WAGO et Siemens 300 et 400.
Dans le menu du logiciel PCVue nous commençons par **configure → communication → Equipement** pour obtenir le panel de configuration IHM de SCADA



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

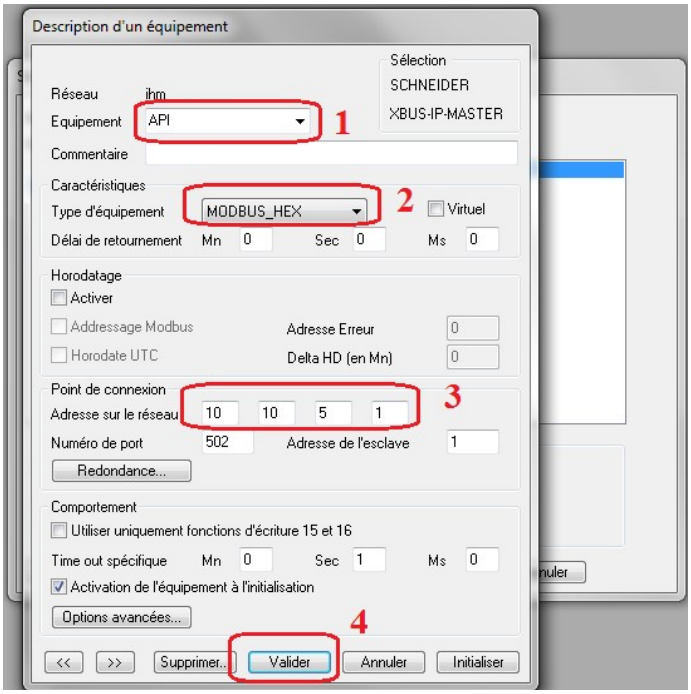
4. Création du réseau Ethernet : SCHNEIDER/ XBUS-IP-MASTER

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



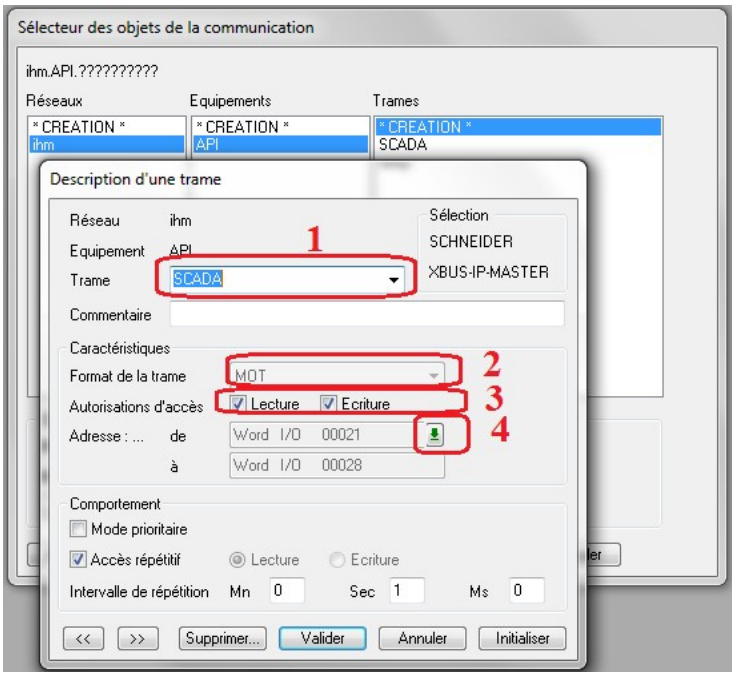
Il faut ensuite valider puis annuler.

Dans équipements, il faut définir le nom de l'automate ici nommée **API**, 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).



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.

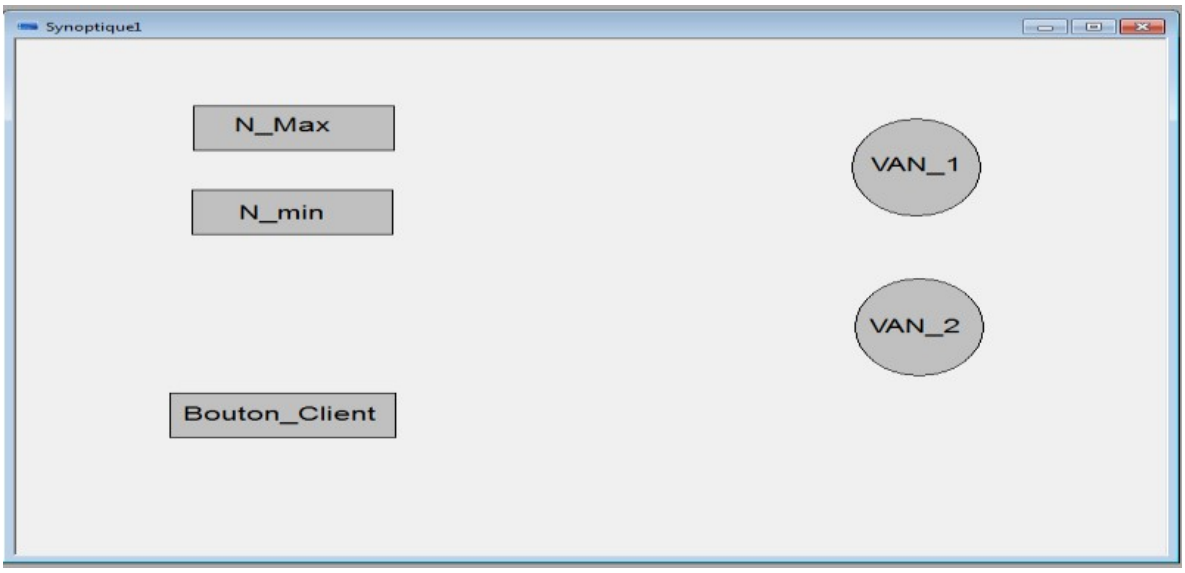


Il faut ensuite définir la taille de la trame, c'est-à-dire le nombre de données ou d'éléments binaire qui seront lu dans l'automate. Il faut aussi préciser dans quelle zone de l'automate (ou adresse) les données seront lues. Pour cela on clique sur la flèche verte et on définit l'adresse de départ,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

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