Industrial communication labs

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) Design a simple distributed application
3) Implement and observe communication flows
4) Analyze the flows, optimize the communication

General organization of the lab

The lab is organized in two workshops corresponding to the study of one major SCADA protocol each: S7 and Modbus/TCP. For each workshop four PLC are available together with development computers and a SCADA software or HMI. The class will be split in two teams corresponding to the two workshops, each team consisting of at most four groups.

Your first task will be to decide to which team and group you'll belong.

PART I: Communication between PLC and HMI/SCADA

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

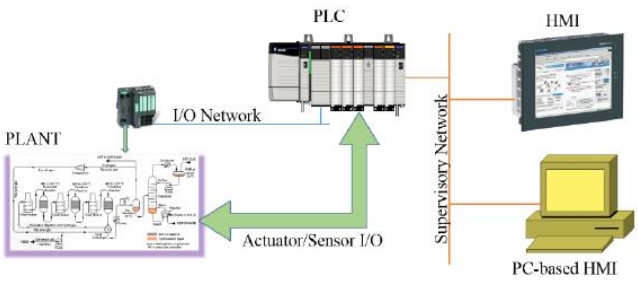


Figure 1.The simple SCADA system

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 PLC. Then your group will be completely autonomous. 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

The control problem.

For practical (available space) reasons, it is not possible to have 8 plants to be controlled by the 8 PLC. Therefore the plant is simulated by a computer program. However the PLC receives "true signals" on the I/O card while the computer simulation is interfaced with an electronic card. The Figure 2 displays the lab configuration (it is called a Hardware in the Loop simulation).

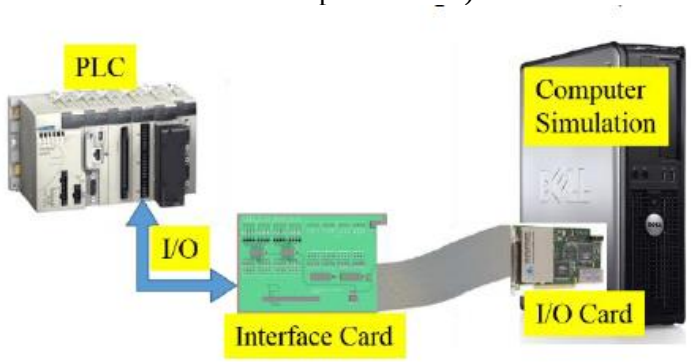


Figure 2. Hardware in the loop bench

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 level) 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.

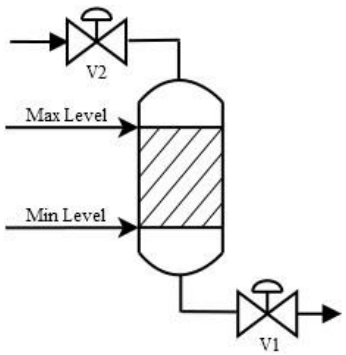


Figure 3. A simple tank

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.

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.

SCADA interfacing.

Depending on the hardware available for your group the HMI will be an industrial touch screen or a computer based HMI using the industrial supervisory control software PCVue. 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 group. An appendix describing the technical details for each lab bench is available.

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.

PART II: The distributed application

For the second part of the lab you'll have to collaborate with your colleagues while the control objectives change. We consider now that the four tanks will share a common pipe (each tank provides a different color of paint and we don't allow mixes – see Figure 4).

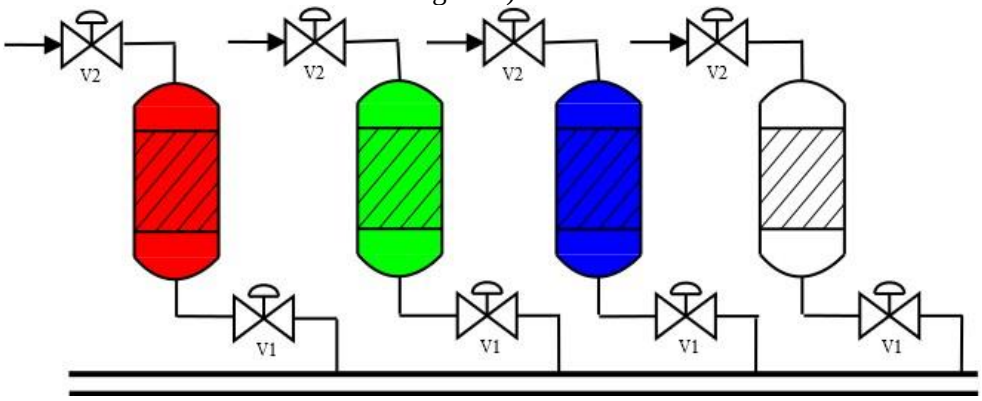


Figure 4. Color supplying tanks through a shared pipe

Then the first control specification is modified as follows:

If the tank level is between min and max, when a client request is raised *and there is no other request raised to any of the other tank controllers*, open the valve V1 as long as the client request is active.

There will be also a supplementary diagnostic objective: if your controller is blocked by another supply task on another tank, raise the error bit and write the error code (0x02). If any other tanks are also blocked add the error code (0x04).

Work to do.

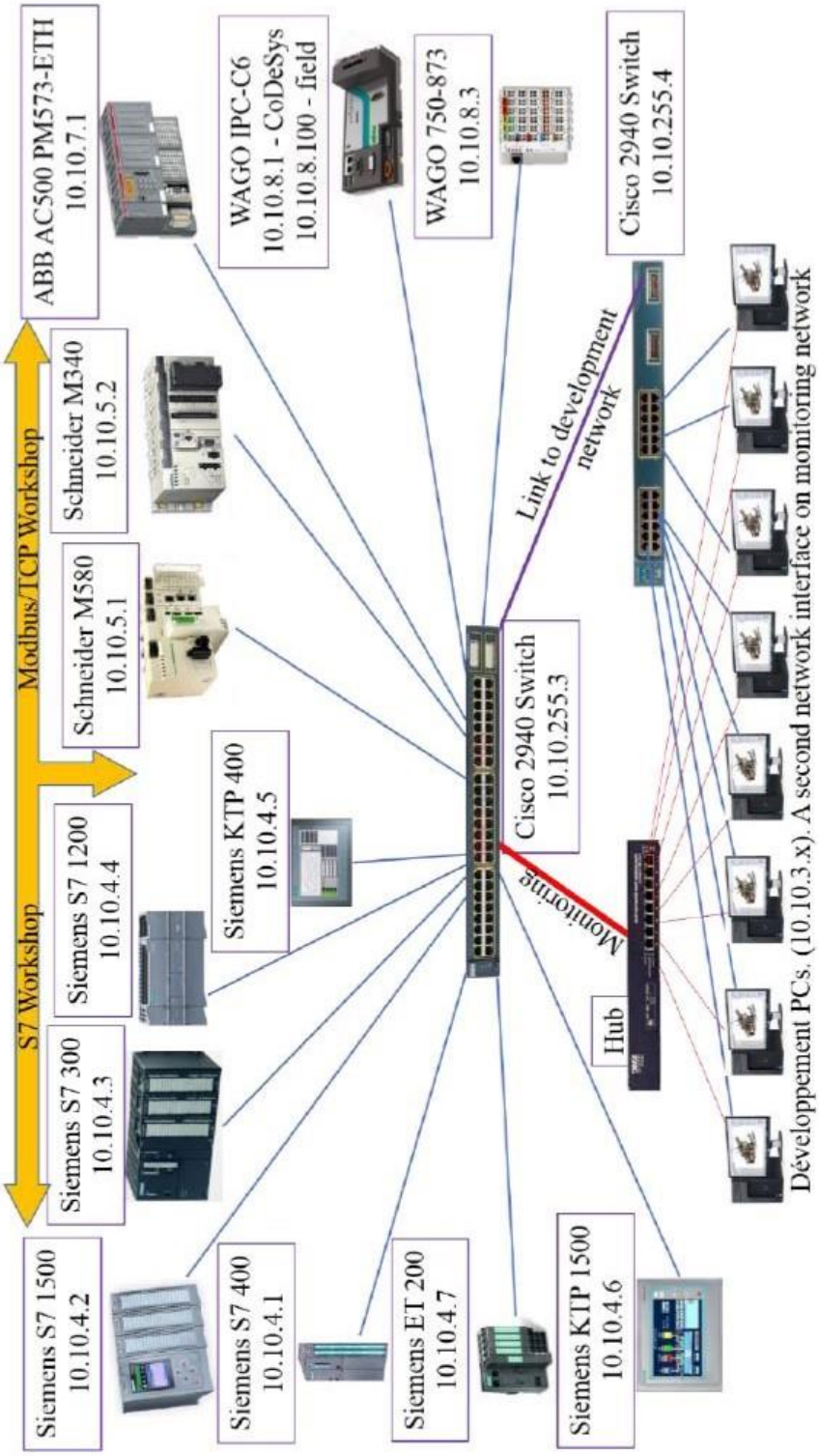
1) Establish the list of new data flows to be added to your system
2) Implement the communication in your PLC program (programming is PLC dependent, an appendix is describing the various approaches).
3) Use Wireshark to check and identify the new data flows.
4) Compare the physical data flows with the functional data flows. Are there any differences?
5) Compute the efficiency of the protocol (ration of bits of information over total number of protocol bits).
6) Comment out the use of network bandwidth. Can you estimate the maximal number of variables that can be transferred?
7) Are there any improvements in communication part in the PLC programming that can be made?
8) Propose an alternate control logic specification that achieves the same objective (not allowing color mix). Comment out the advantages of the two solutions.

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)
- Based on the previous point you shall be able to propose an optimal design of the data exchange in order to minimize the network bandwidth use (and enforce the real-time performance).

APPENDIX A. G-ICS network architecture

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



APPENDIX B: 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

The communication setup program is started from the menu "Configure->Communication->Equipement"(see figure B1)

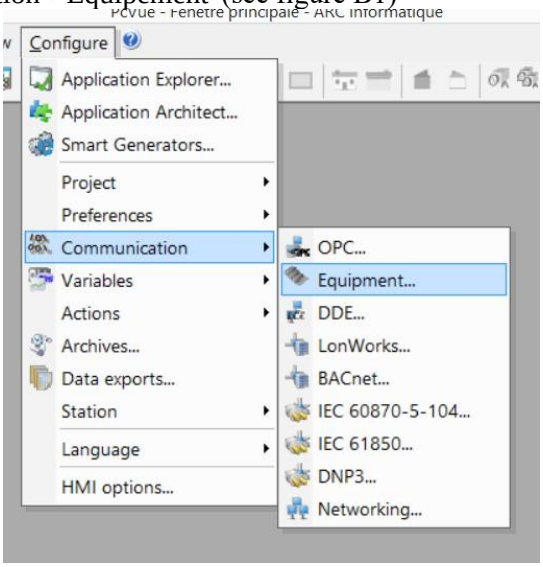


Figure B1: communication menu

Most of the parameters that 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 group of parameters used to identify the protocol you'll use, the equipment you'll talk to and the variables you scan (figure B2).

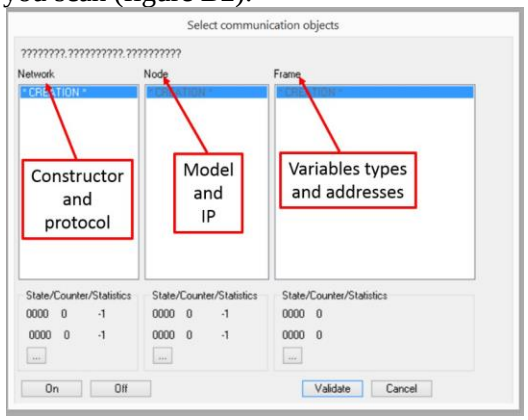


Figure B2: Communication setup

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

S7 communications setup

For the S7 communication you have to choose “Siemens” and “IP-ISO-S7” for the “Network” creation. Choose whatever name you want for the network and let the other values to default.

For the “Node” configuration choose “S7-300/400” for “Equipement Type” and add the IP number of your equipment into “Network address”. Let the other parameters to default.

Finally, into the “Frame” configuration you have to choose the type of variables you want retrieve (bit/byte/word/etc) and the variables addresses into the PLC memory. Whatever the type of your variables is, the S7 driver from PCVue is able to transfer only WORD. It means that, even if you declare your variable as a bit (%M0.0 for example), PCVue will still retrieve at least a WORD (16 bits) per frame (for the example, the bit you want will be the bit 0 of %MW0).

Then you have the choice between six types of variables addresses:
1) DBW: Data Block Words. Data Blocks are Step 7 data structures of free type. This choice let you to retrieve a PLC structured variable WORD by WORD. It is Siemens preferred data access because it does not rely on memory addresses, but only on variables names.
2) MW: memory words. Access to %MW addresses
3) EW: input memory addresses. Direct access to PLC inputs (E states for “Ein” meaning Input in German). Depending on PLC model and configuration, EW addresses access may be protected.
4) AW: output memory addresses. Direct access to PLC outputs (A states for “Aus” meaning Output in German). Depending on PLC model and configuration, AW addresses access may be protected.
5) ZW: counters memory addresses. Direct access to PLC counters (Z states for “Zähler” meaning Counter in German). Do not concern this lab.
6) Information: four bytes containing PLC CPU status information.

The choice of the type and the address of the variable depends on you and on your programming. The value of an inputs (for example) may be copied in a DB or a memory location. Then the access to the value of the input is possible by any of the three first methods (DBW, MW or EW).

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 “Equipement 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 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 of 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 of the 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. It, 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.

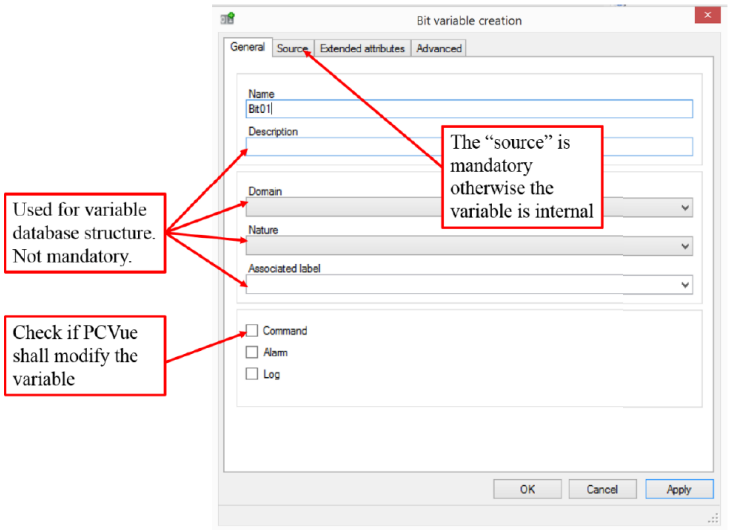


Figure B3: Variable creation dialog

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.

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 PCVCue 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".