# An Open-Source Hardware-In-The-Loop Virtualization System for Cybersecurity Studies of SCADA Systems

**Stéphane Mocanu,**
INRIA/Laboratoire d'Informatique de Grenoble

# PRESENTATION PLAN

- Industrial control systems
- Motivation
- Hardware in the loop simulation
- The demonstrator today
- The future project

# INDUSTRIAL CONTROL SYSTEMS (SCADA)

Cyberphysical Systems

IT

OT
Operational Technology

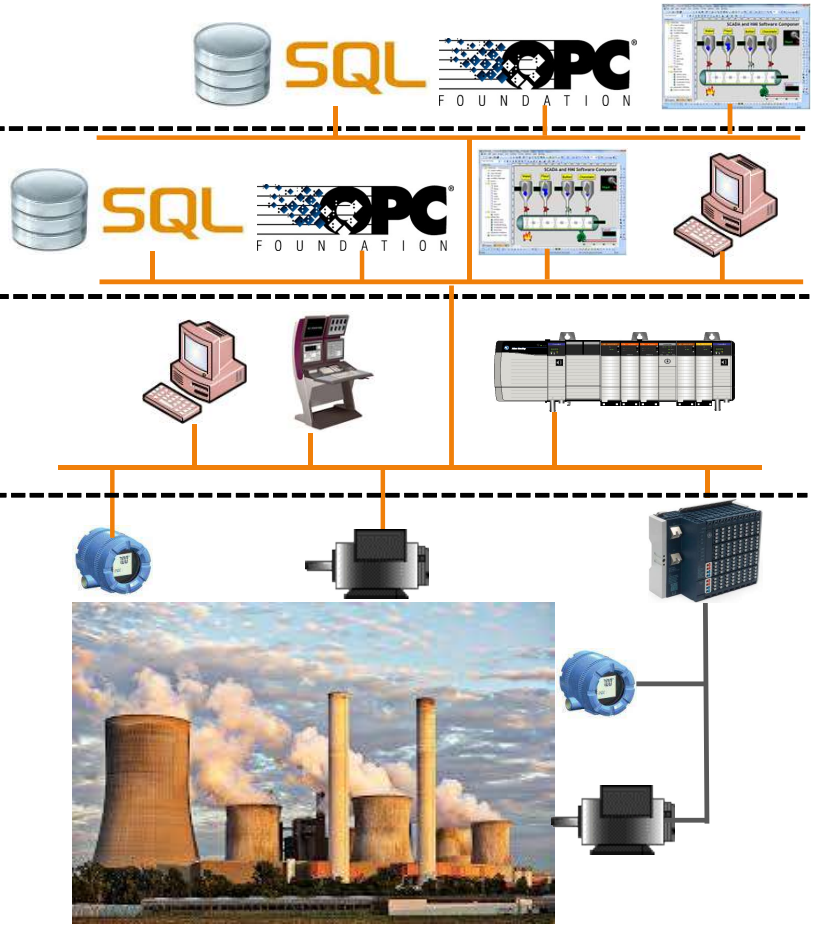
**Remote connections**

**Supervisory and journaling**
**No real-time**

**Controllers**
**Soft real-time communication**

**Intelligent sensors and actuators**
**Hard real time communication**

**Main asset : the physique plant**



# THE NEED FOR A SANDBOX

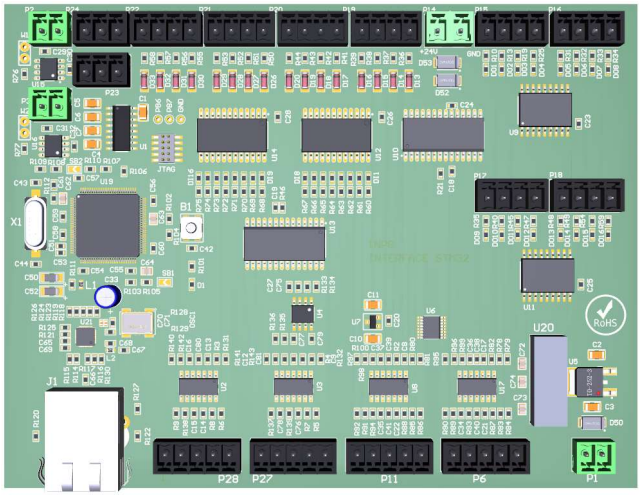
- **A proprietary world**
  - Real control hardware cannot be (realistically) simulated
  - Communication protocol stacks not free
  - Dedicated RTOS (pSos, VxWorks)
- **Research will never have access to a real SCADA system**
  - Industrial secret
- **Pentesting impossible on production SCADA systems**
  - Industrial risk
- **Only few data sets available on Internet for benchmarking**
- **Academic use-cases too simple to be relevant (one or two PLC)**

# THE SCADA SANDBOX PROJECT

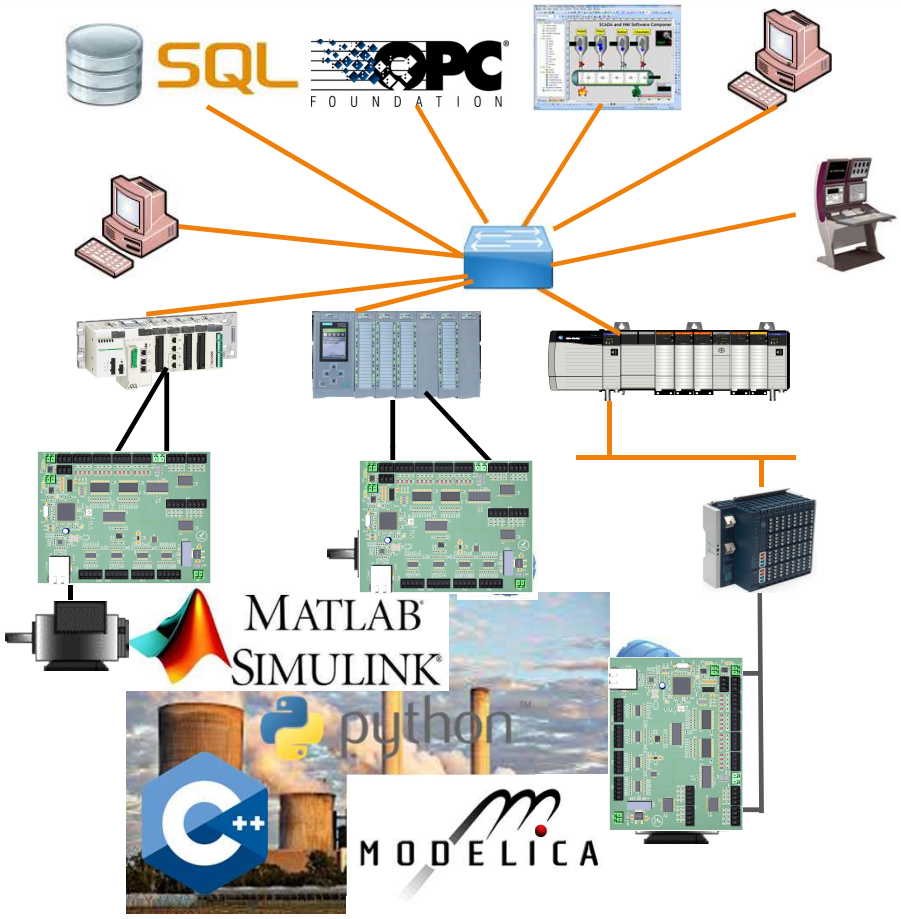
- **An open, scalable SCADA workbench**
  - Multi-protocol
  - Multi-vendor
  - Flexible (not limited to a single application)
  - Close to industrial size (~ 100 connected devices)
- **Must be “pentest proof”**
  - No physical process
  - A hardware-in-the-loop simulation approach
- **Reasonable cost**
  - Professional HiL systems cost is around 40k€/measurement point (8 sensors)
  - We target over 1000 simulated sensors
- **Remotely accessible**

# HARDWARE IN THE LOOP SYSTEM

- **Home made electronic interface card**

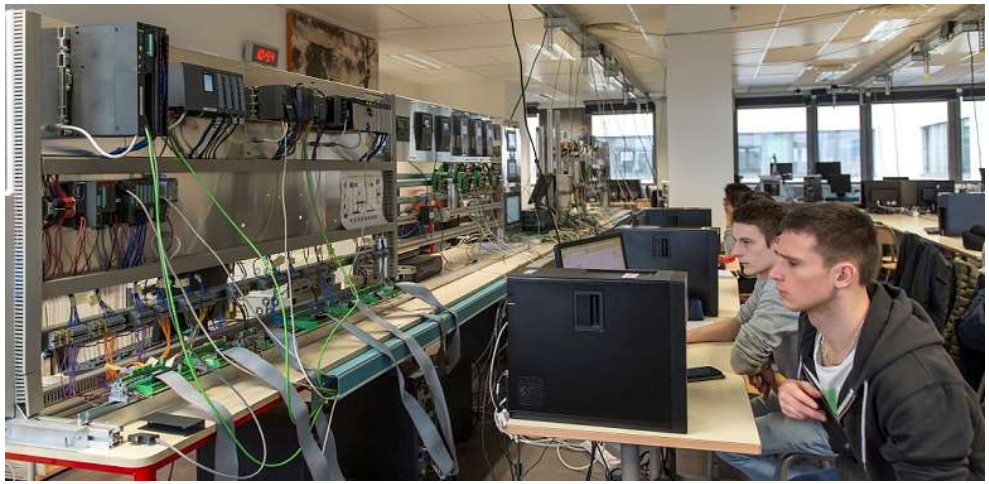


  - **24 sensors and 24 actuators**
    - 16 digital inputs / 16 digital outputs
    - 8 analog inputs / 8 analog outputs
  - **Less than 500€**
  - **Reasonable timing performance ( < 10 ms response time)**
  - **Easily chain (Ethernet addressing)**



# TWO MAIN APPLICATION FIELDS

**Industrial automation: PLC, SCADA et OPC**

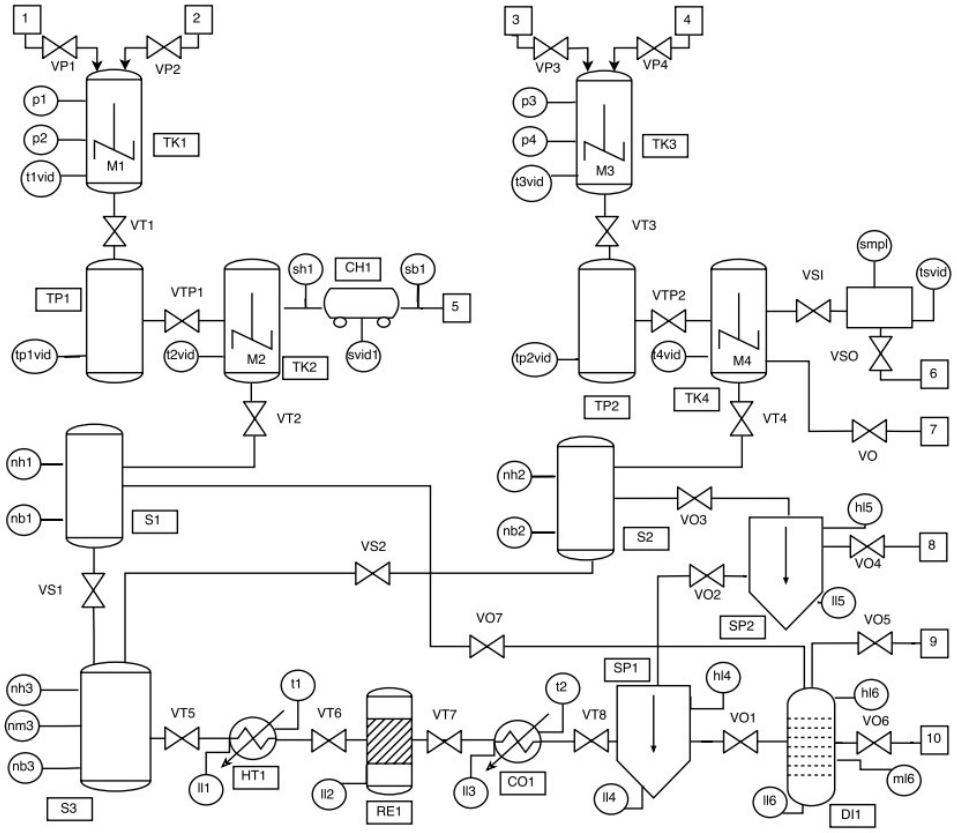


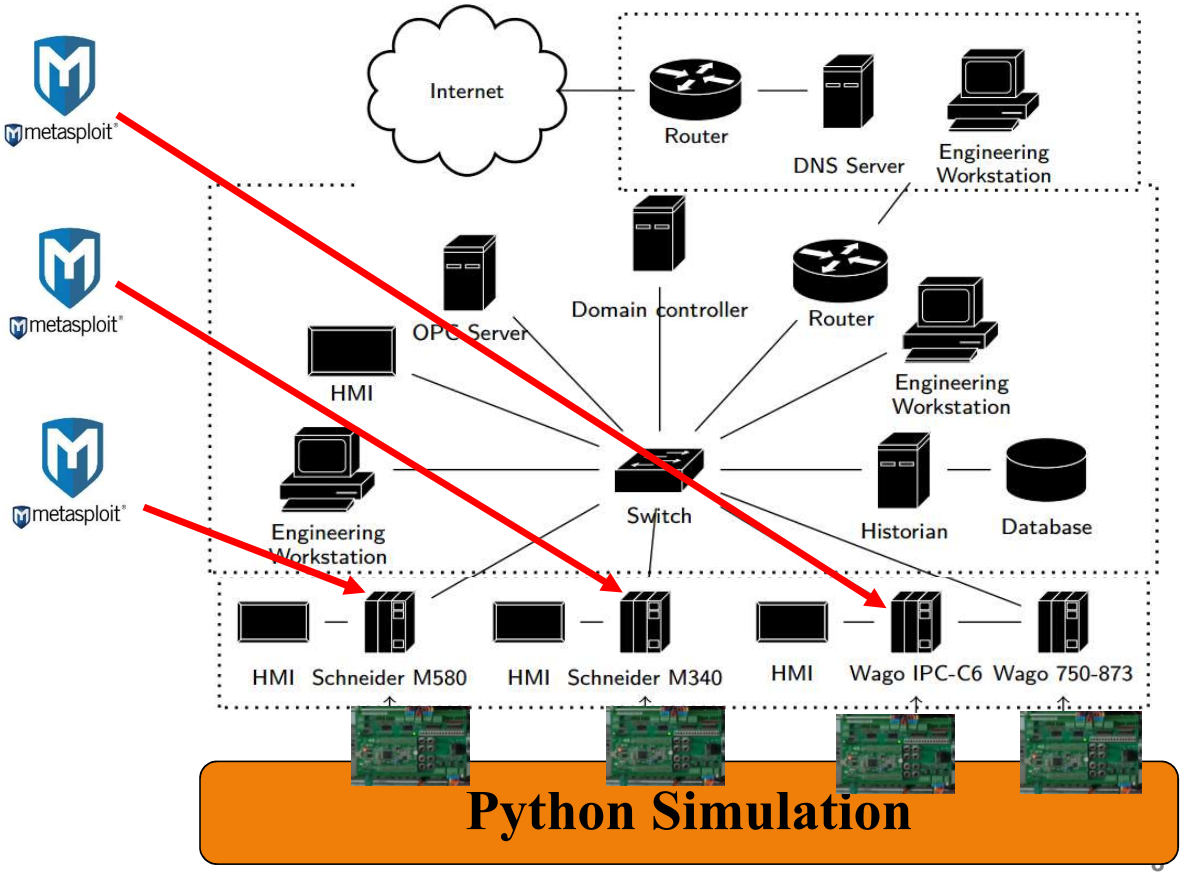
**Smart-grid communication**

- **IEC 61850**

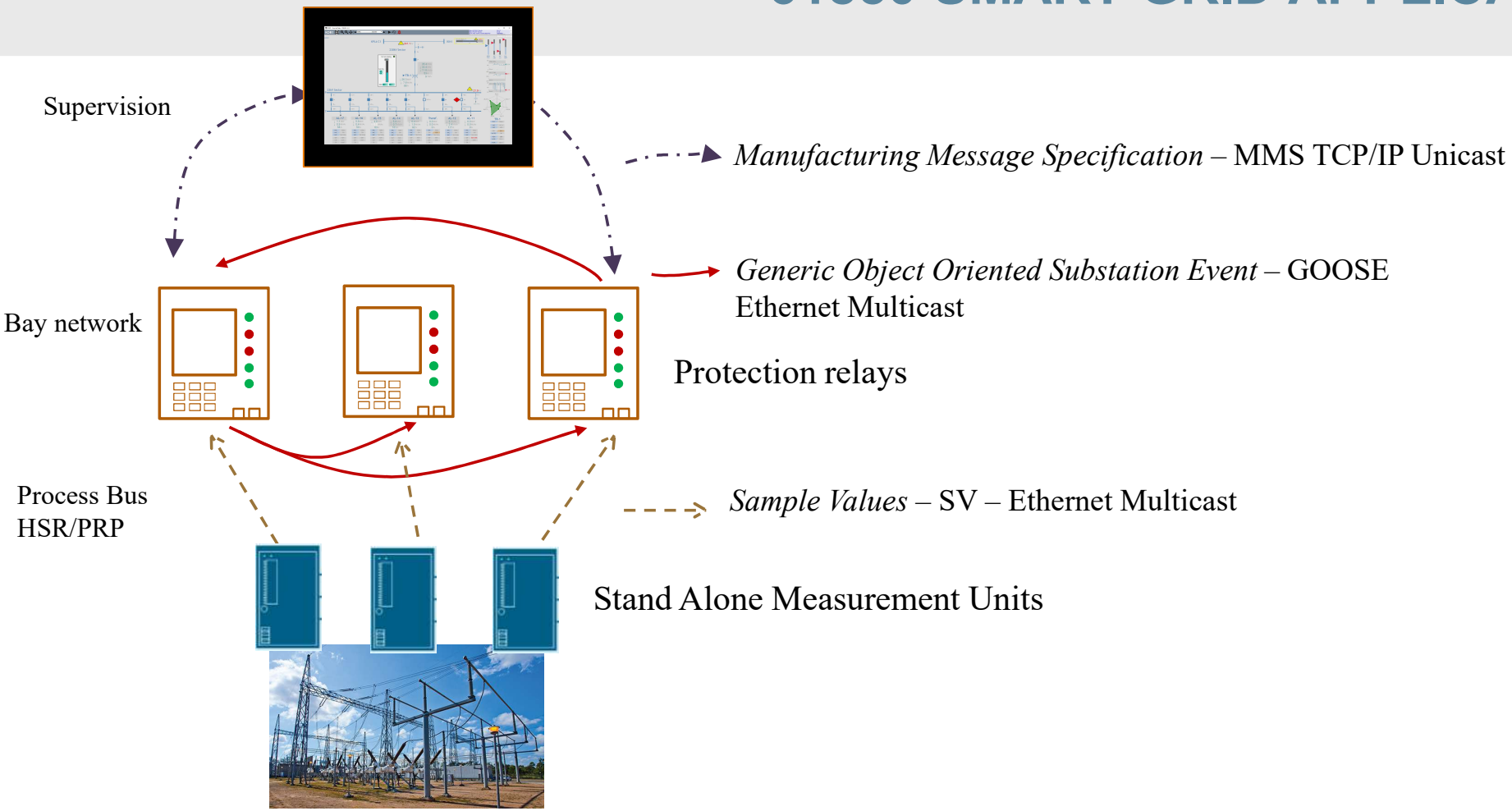
# A SIMPLE APPLICATION EXAMPLE ( ~70 I/O)

- **Tennessee Eastman Chemical Company (O. Koucham PhD benchmark)**

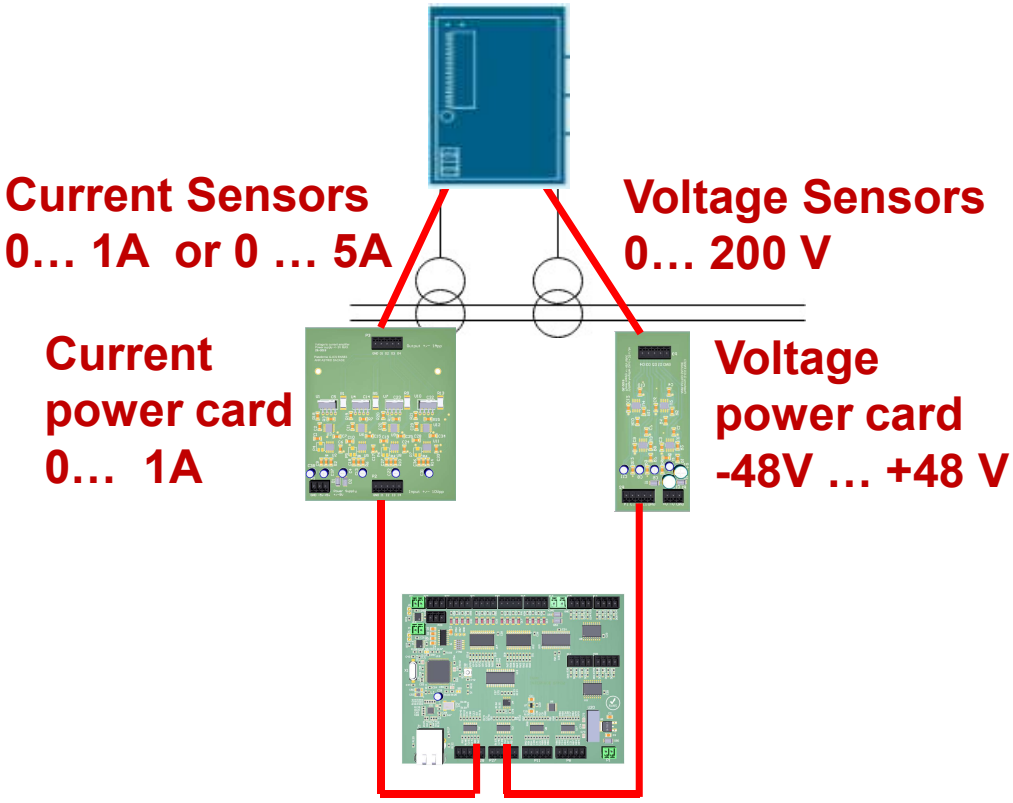


# 61850 SMART GRID APPLICATIONS

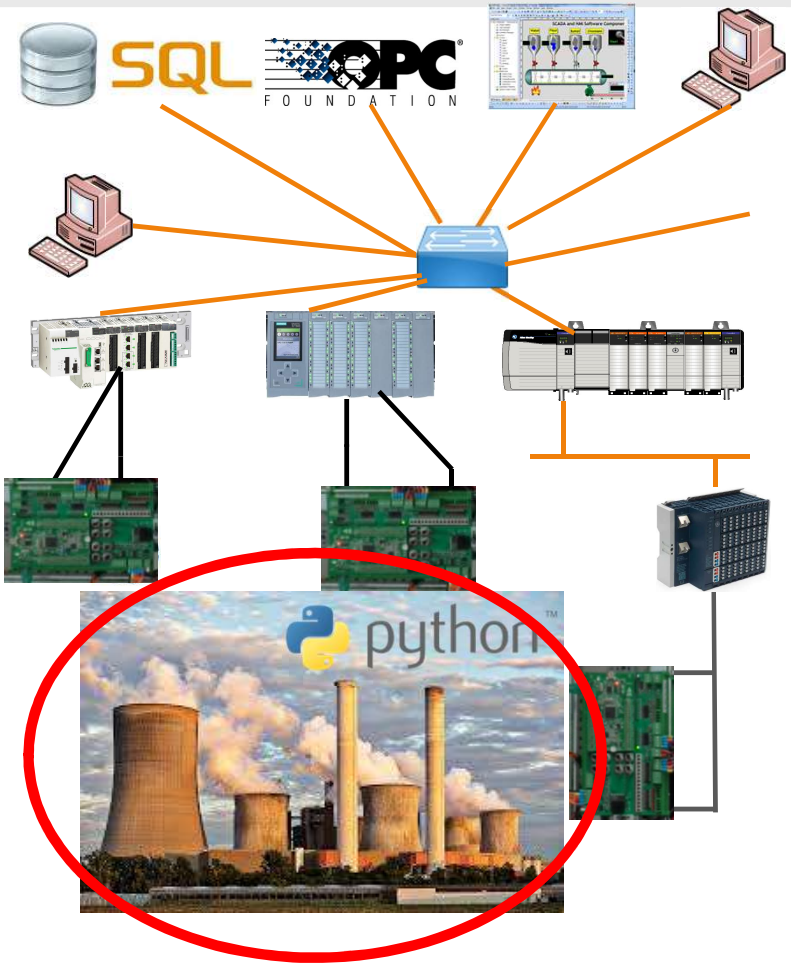


# 61850 SMART GRID APPLICATIONS




- **Cybersecurity applications (M. Kabir-Querrec PhD benchmark)**
  - ► False event injection (GOOSE attack)
  - ► False measures injection (SMV attack)
  - ► Resilient architectures
    - *C&ESAR 2015*

# INDUSTRIAL PROCESS SIMULATOR



# NEEDS FOR PROCESS SIMULATOR

- **Genericity**
  - Ability to model any industrial process (e.g., electricity distribution, chemical factory)
  - Not focused only on a single application type
  - Easily customizable
- **Fine grain modeling**
  - Not our priority at the moment /!\
  - We focus on basic functionalities (e.g., “opening a valve” and not on the intrinsic properties of the valve), yet finer modeling can be required.
- **Ability to communicate /!**
  - With interface cards (thus with PLC, SCADA, etc)
  - In TCP/RTU Modbus
  - With data bases, API, etc
  - With GPIOS
- **Ability to run on various systems**
  - Raspberry PI, computer, etc

# EXISTING PROCESS SIMULATORS

- **A real-time (time stepped simulation is needed)**
  - **Process dynamics simulated at real-life time not computer time**

- **Tested and validated simulators :**
  - **Modelica (with real-time) toolbox**
    - Native UDP communication implementation
  - **Factory I/O**
    - Engine I/O library -> interface provided
  - **Home I/O**
    - Engine I/O library -> interface provided
  - **Matlab/Simulink**
    - Native UDP communication implementation

# PROGRAM AND CONFIGURATION

- **Built on Atollic TrueSTUDIO® for STM32 V9.2.0**

- **One can configure**
  - Mode (PLC or protection relay)
  - Network configuration
  - Background task
- **Mode (gics.c)**
- `static int config_mode = GICS_API; // configure for PLC`
- `static int config_mode = GICS_IED; // configure for protection relay`

# NETWORK CONFIGURATION

- By default a card is idenfyied by a RACK_ID and a CARD_ID

- Ethernet config (ethernetif.c)
- MACAddr[0] = 0x02;
- MACAddr[1] = 0x61;
- MACAddr[2] = 0xc5;
- MACAddr[3] = RACK_ID;
- MACAddr[4] = 0x00;
- MACAddr[5] = CARD_ID;
  - You can use any values instead default ones but respect the rules :
    - Each card into the same network has a different MACAddr
    - First bit on byte 0 has to be 0 (otherwise it is a broadcast address).

# IP CONFIGURATION

- File lwip.c
- IP_ADDRESS[0] = 10;
- IP_ADDRESS[1] = 10;
- IP_ADDRESS[2] = 100;
- IP_ADDRESS[3] = RACK_ID*16+GICS_ID;
- NETMASK_ADDRESS[0] = 255;
- NETMASK_ADDRESS[1] = 255;
- NETMASK_ADDRESS[2] = 0;
- NETMASK_ADDRESS[3] = 0;
- GATEWAY_ADDRESS[0] = 10;
- GATEWAY_ADDRESS[1] = 10;
- GATEWAY_ADDRESS[2] = 255;
- GATEWAY_ADDRESS[3] = 254;

# CARD COMMUNICATION PROTOCOL

- **Simple requests derived from Modbus protocol**
- **Frame data defined in gics.h**
  - typedef struct GICSTransaction {
  - unsigned char function;
  - unsigned char magic; always GICS_MAGIC 0xd0
  - unsigned short length;
  - unsigned short data[255];
  - } GICSTransaction;
- **Elementary Functions**
  - #define GICS_READ 0x01
  - #define GICS_WRITE 0x02
  - #define GICS_DISCRETE 0x04
  - #define GICS_ANALOG 0x08
  - #define GICS_DA 0x10

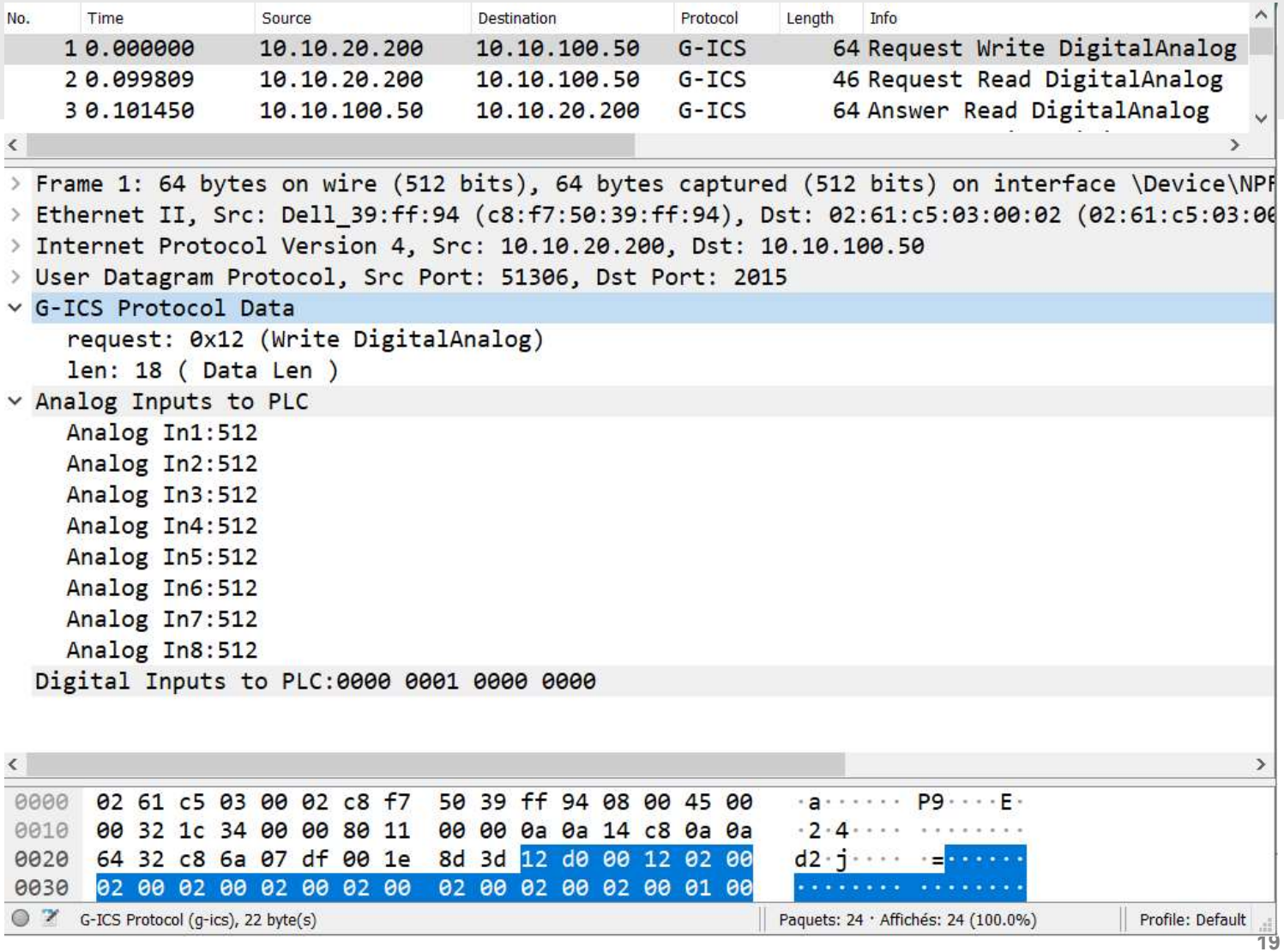
# CARD COMMUNICATION PROTOCOL

- **Request functions**
  - Combination of READ/Write and data type
  - Read Analog/digital = GICS_READ + GICS_DA = 0x11
  - Write Analog/digital = GICS_WRITE + GICS_DA = 0x12
- **Anwsers**
  - Write requests are not answered
  - Read answers have a magic number 0xd1
- **Default UDP port = 2015**

- **A Wireshark dissector exists**

# DATA EXCHANGE

- **Write request (D/A)**



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.10.20.200 | 10.10.100.50 | G-ICS | 64 | Request Write DigitalAnalog |
| 2 | 0.099809 | 10.10.20.200 | 10.10.100.50 | G-ICS | 46 | Request Read DigitalAnalog |
| 3 | 0.101450 | 10.10.100.50 | 10.10.20.200 | G-ICS | 64 | Answer Read DigitalAnalog |

```
> Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF
> Ethernet II, Src: Dell_39:ff:94 (c8:f7:50:39:ff:94), Dst: 02:61:c5:03:00:02 (02:61:c5:03:00
> Internet Protocol Version 4, Src: 10.10.20.200, Dst: 10.10.100.50
> User Datagram Protocol, Src Port: 51306, Dst Port: 2015
v G-ICS Protocol Data
    request: 0x12 (Write DigitalAnalog)
    len: 18 ( Data Len )
v Analog Inputs to PLC
    Analog In1:512
    Analog In2:512
    Analog In3:512
    Analog In4:512
    Analog In5:512
    Analog In6:512
    Analog In7:512
    Analog In8:512
  Digital Inputs to PLC:0000 0001 0000 0000
```

```
0000  02 61 c5 03 00 02 c8 f7  50 39 ff 94 08 00 45 00   ·a······ P9····E·
0010  00 32 1c 34 00 00 80 11  00 00 0a 0a 14 c8 0a 0a   ·2·4···· ········
0020  64 32 c8 6a 07 df 00 1e  8d 3d 12 d0 00 12 02 00   d2·j···· ·=······
0030  02 00 02 00 02 00 02 00  02 00 02 00 02 00 01 00   ········ ········
```

G-ICS Protocol (g-ics), 22 byte(s) | Paquets: 24 · Affichés: 24 (100.0%) | Profile: Default

# DATA REQUEST

- **Read D/A request**

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.10.20.200 | 10.10.100.50 | G-ICS | 64 | Request Write DigitalAnalog |
| 2 | 0.099809 | 10.10.20.200 | 10.10.100.50 | G-ICS | 46 | Request Read DigitalAnalog |
| 3 | 0.101450 | 10.10.100.50 | 10.10.20.200 | G-ICS | 64 | Answer Read DigitalAnalog |

```
> Frame 2: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface \Device\NPF
> Ethernet II, Src: Dell_39:ff:94 (c8:f7:50:39:ff:94), Dst: 02:61:c5:03:00:02 (02:61:c5:03:00
> Internet Protocol Version 4, Src: 10.10.20.200, Dst: 10.10.100.50
> User Datagram Protocol, Src Port: 51307, Dst Port: 2015
v G-ICS Protocol Data
    request: 0x11 (Read DigitalAnalog)
    len: 0 ( Data Len )
```

```
0000  02 61 c5 03 00 02 c8 f7  50 39 ff 94 08 00 45 00   ·a······ P9····E·
0010  00 20 1c 35 00 00 80 11  00 00 0a 0a 14 c8 0a 0a   · ·5···· ········
0020  64 32 c8 6b 07 df 00 0c  8d 2b 11 d0 00 00        d2·k···· ·+····
```

# DATA REQUEST

- **Read D/A Answer**

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.10.20.200 | 10.10.100.50 | G-ICS | 64 | Request Write DigitalAnalog |
| 2 | 0.099809 | 10.10.20.200 | 10.10.100.50 | G-ICS | 46 | Request Read DigitalAnalog |
| 3 | 0.101450 | 10.10.100.50 | 10.10.20.200 | G-ICS | 64 | Answer Read DigitalAnalog |

```
> Frame 3: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NP
> Ethernet II, Src: 02:61:c5:03:00:02 (02:61:c5:03:00:02), Dst: Dell_39:ff:94 (c8:f7:50:39:f
> Internet Protocol Version 4, Src: 10.10.100.50, Dst: 10.10.20.200
> User Datagram Protocol, Src Port: 2015, Dst Port: 51307
v G-ICS Protocol Data
    request: 0x11 (Read DigitalAnalog)
    len: 18 ( Data Len )
v Analog Outputs from PLC
    Analog Out1:2029
    Analog Out2:2049
    Analog Out3:2046
    Analog Out4:2046
    Analog Out5:2048
    Analog Out6:2049
    Analog Out7:2052
    Analog Out8:2048
  Digital Outputs from PLC:0000 0000 0000 0000
```

```
0000  c8 f7 50 39 ff 94 02 61  c5 03 00 02 08 00 45 00   ··P9···a ·····E·
0010  00 32 00 07 00 00 ff 11  2e a6 0a 0a 64 32 0a 0a   ·2······ ····d2··
0020  14 c8 07 df c8 6b 00 1e  50 87 11 d1 00 12 07 ed   ·····k·· P·······
0030  08 01 07 fe 07 fe 08 00  08 01 08 04 08 00 00 00   ···· ····
```

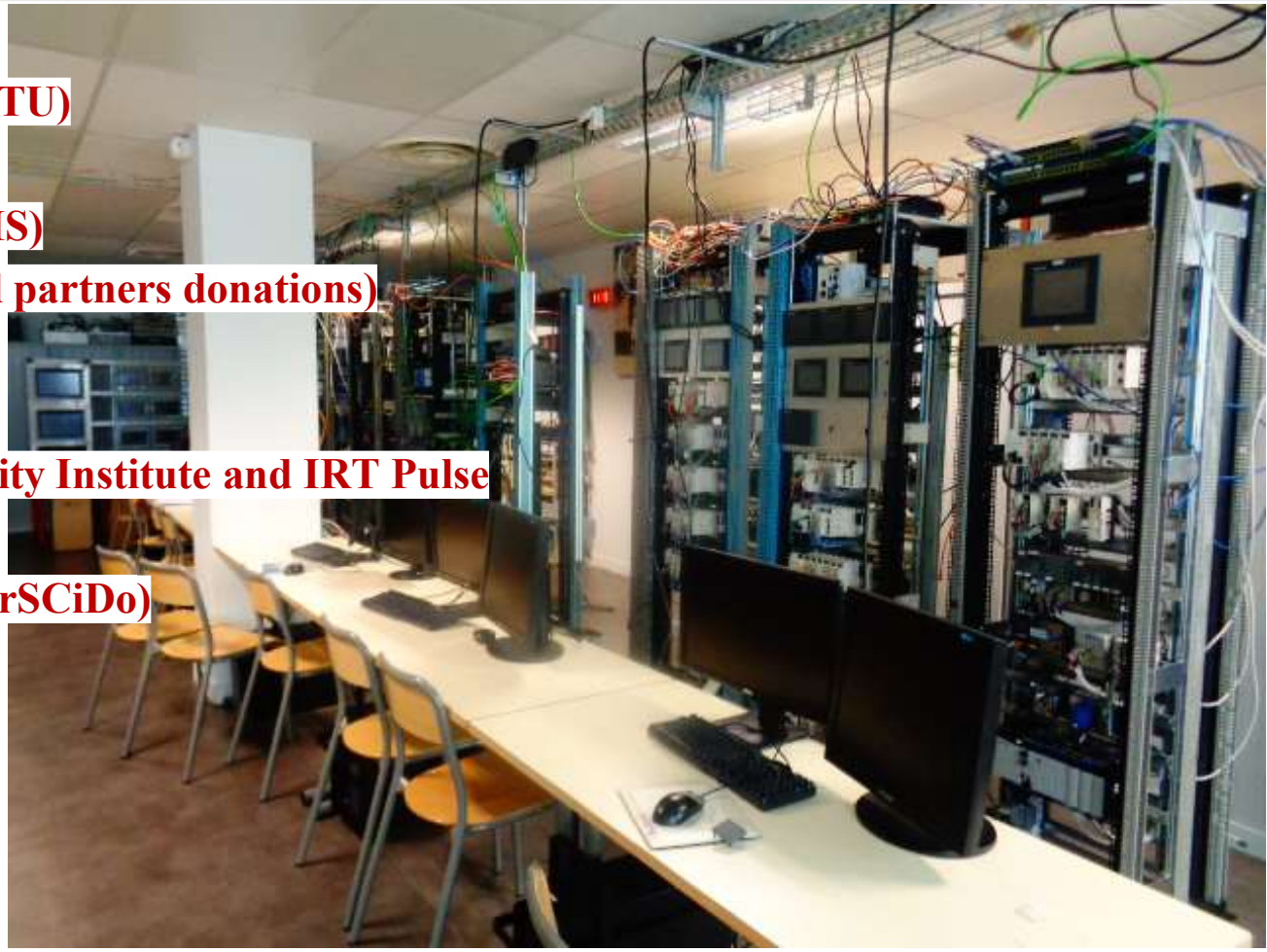
# PROGRAM MAIN LOOP (GICS.C)

- **Request decoding and DI/O**
  - ► gics_udp_server_receive_callback
- **Interrupt DA/C updater**
  - ► Update_DAC_Handler
- **Analog I/O : directly handled via memory transfer**
- **Protection relay signal adapter (sinusoïdal signals generation)**
  - ► gics_update_DAC_vars

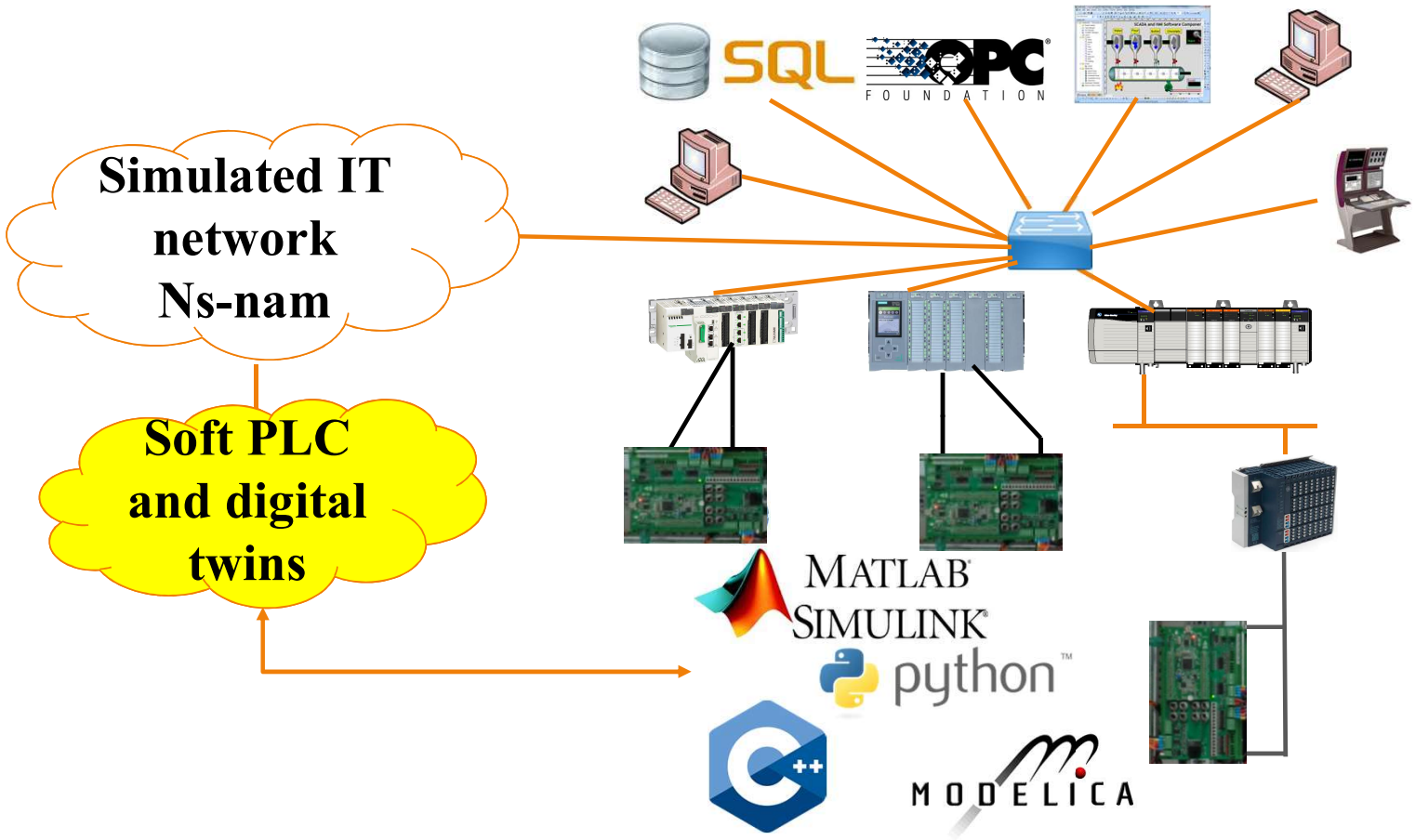
- **A/D and D/A conversion rules**
- **DAC are 10 bits precision. Therefore DAC inputs are 0…1023 for a +/- 10V output**
- **ADC are  12 bits precision. For à +/-10V input, the output will vary between 0 and 4095**
- **Note that a small bias is to be expected**

# THE WORKBENCH TODAY

- Over 100 devices connected (PLC, HMI, RTU)
- Over 1000 virtual I/O
- Around 20 OT servers (OPC, SCADA, EMS)
- Over 1M€ investment (including industrial partners donations)
- VPN accessible
- A network monitoring infrastructure
- Demonstration workbench for Cybersecurity Institute and IRT Pulse
- Presence at FIC 2018,  2019
- Some public datasets including attacks (PerSCiDo)
- Public system architecture
- Interface card PCB publicly available
- Driver for Home I/O, Factory I/O



# NEXT LEVEL: FULL SCALE SCADA CYBER-RANGE



# GET THE SOURCES : GICS-HIL PROJECT ON INRIA FORGE

- **Project web site : http://lig-g-ics.imag.fr/**

- **Git access to card schematics, PCB, embedded code, exemples ….**

`https://gitlab.inria.fr/mocanu/gics-hil`