OPCUA
HARDWARE INTEGRATION
Status Report

L.A.P.P : E. Chabanne, T. Le Flour, J.L Panazol

L.L.R : B. Khélifi, S. Chollet, Y. De Oliveira, F. Magniette

L.P.N.H.E : J. Bolmont

Outlines

- Reminders
- Hardware setup description
- Toward a generic OPCUA server
- Future plans

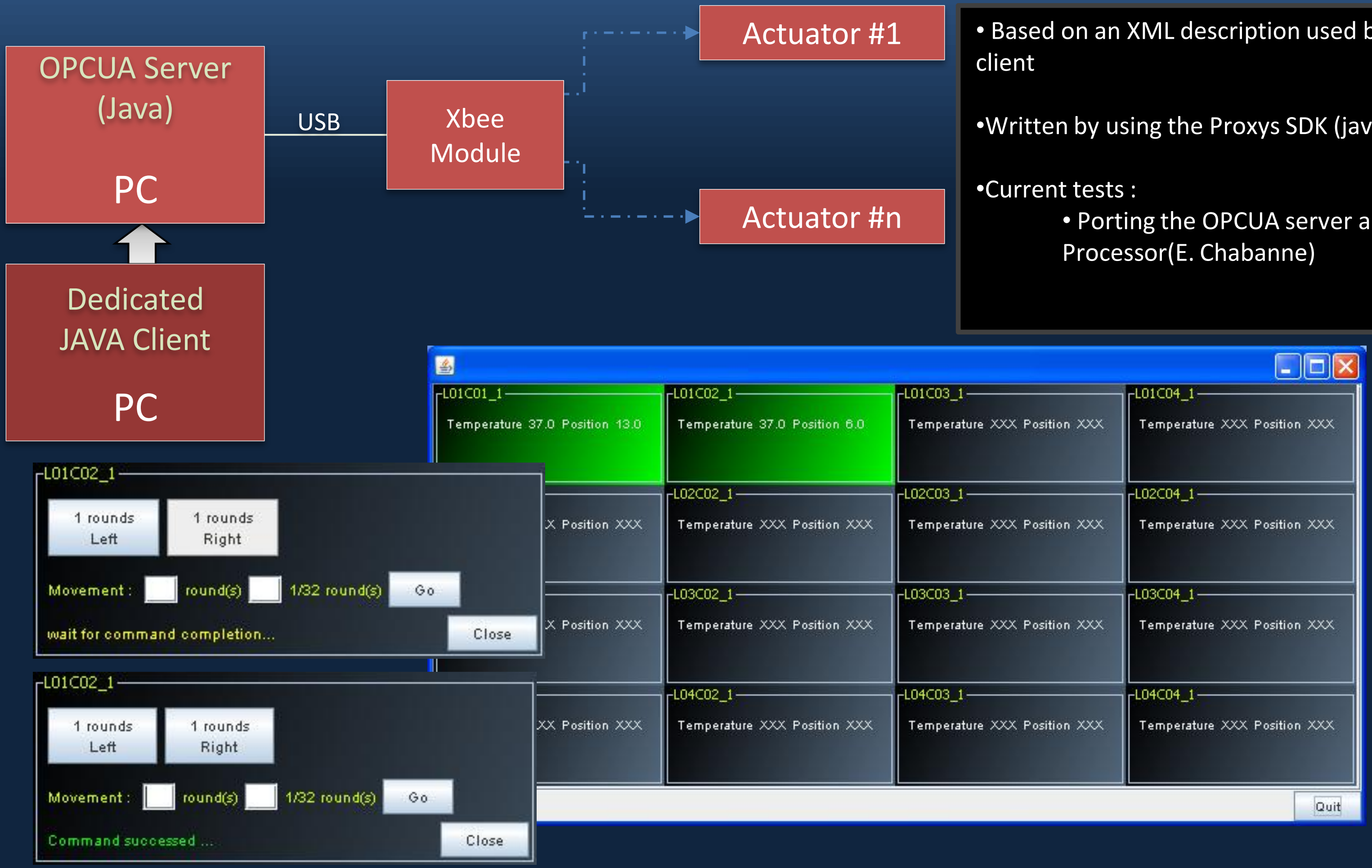
Reminders

- From Collaboration meeting in Amsterdam
  - ➢From the current developments(MST,…) and prototyping :
    - Device integration→ toward a generic description of the device(control points, commands, …)
    - Toolkit development and/or API to facilitate hardware integration in the environment OPCUA/ACS
    - Common slow control data format needed ?

OPCUA Prototypes in progress

- NectarCam slow control (LAPP)
- Mirrors Actuators (LAPP)
- LPNHE Test Bench (LPNHE)
- MST Dummy Camera (LLR)

LAPP OPCUA Prototypes (1)



L01C01_1	L01C02_1	L01C03_1	L01C04_1
Temperature 37.0 Position 13.0	Temperature 37.0 Position 6.0	Temperature XXX Position XXX	Temperature XXX Position XXX
X Position XXX	L02C02_1 Temperature XXX Position XXX	L02C03_1 Temperature XXX Position XXX	L02C04_1 Temperature XXX Position XXX
X Position XXX	L03C02_1 Temperature XXX Position XXX	L03C03_1 Temperature XXX Position XXX	L03C04_1 Temperature XXX Position XXX
XX Position XXX	L04C02_1 Temperature XXX Position XXX	L04C03_1 Temperature XXX Position XXX	L04C04_1 Temperature XXX Position XXX

L01C02_1

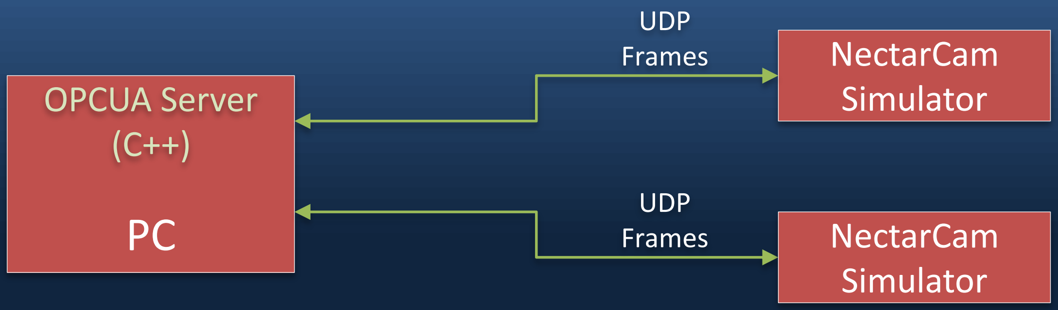
wait for command completion...

L01C02_1

Command successed ...

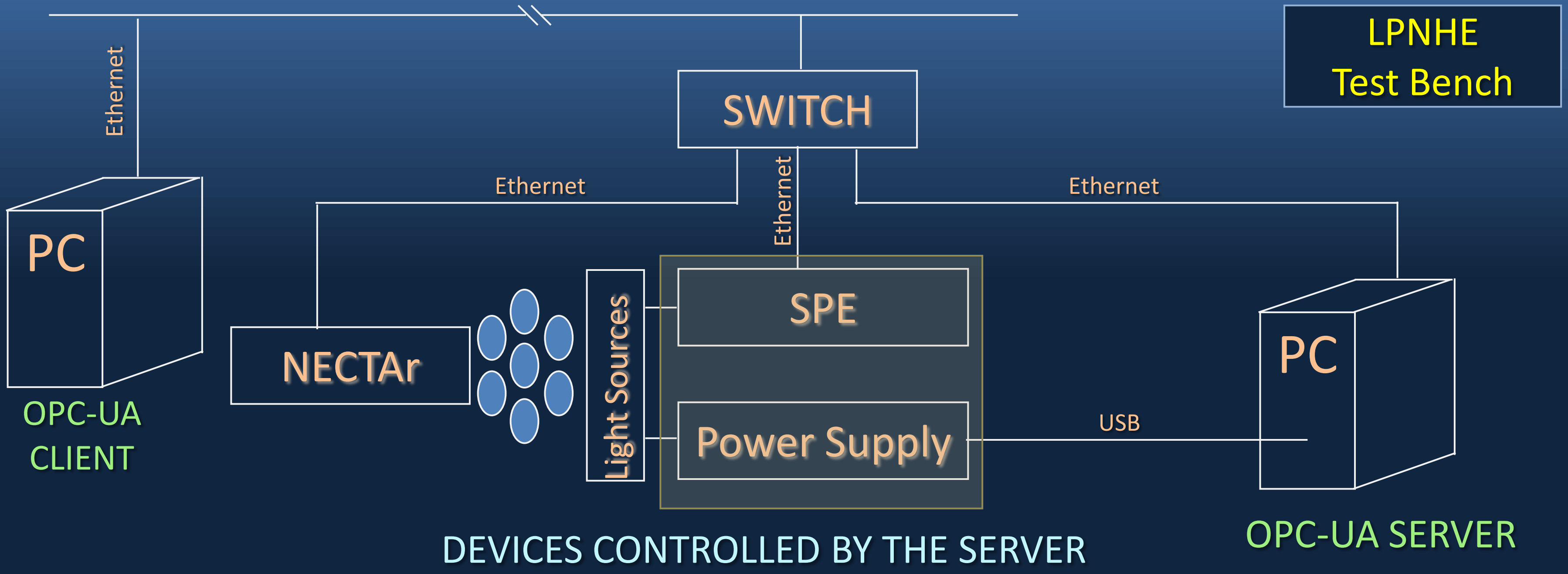
- Based on an XML description used by the server and the client
- Written by using the Proxys SDK (java)
- Current tests :
  - Porting the OPCUA server and client on ARM Processor(E. Chabanne)

LAPP OPCUA Prototypes (2)
NectarCam slow control



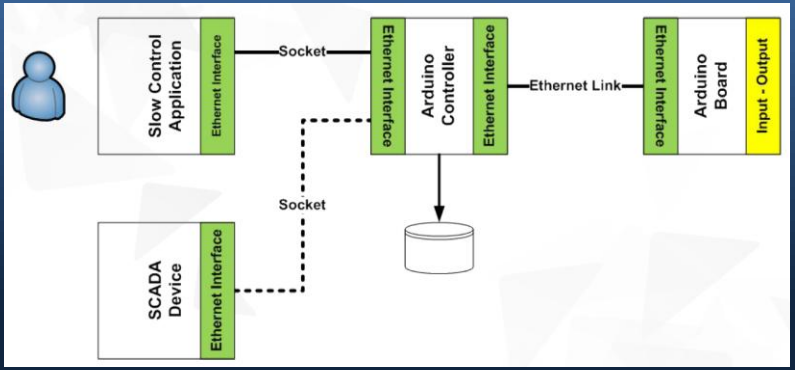
- Based on Nectar Simulator provided by CPPM(Julien)
- An XML File is used to describe
  - ➢The communication (IP address, protocol,…)
  - ➢The data structure coming out of the simulator

LPNHE Prototype

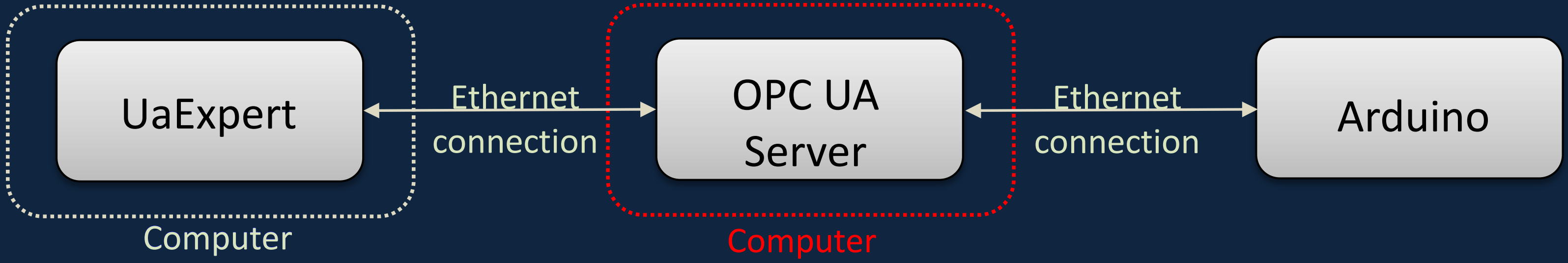


- Implementation of a test bench for :
  - To test the Nectar Frontend in CTA environment

LLR Prototype



- Dummy Camera
  - Host instrumentation for the structure mechanics measurements
    - CCD cameras, Temp sensors, pointing LEDs, accelerometers
  - Test of the Lids motorisation
    - Need to control it
  - Add of some electronics to control and monitor this instrumentation
    - Based on Arduino® device
  - And logically, need ACTL software to control this device


UaExpert

Ethernet connection

OPC UA Server

Ethernet connection

Arduino

Computer

Computer

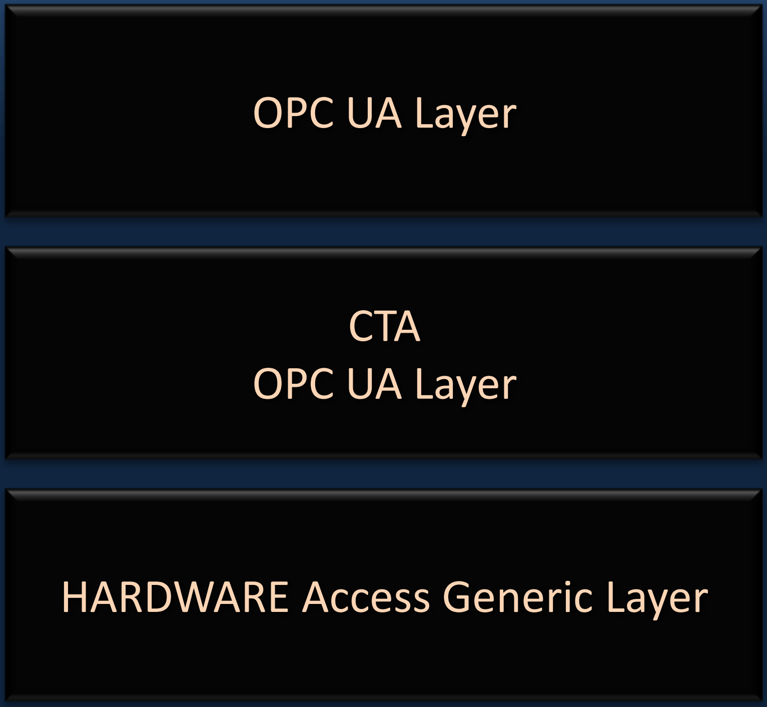
- Development of a Java emulator of the Arduino device
  - Socket Server and simulation of functions
  - Running on a distant machine

- OPC UA Server using the distribution of Prosys SDK
  - Demonstration version, in Java
  - From their examples, developed my own server of Dummy Camera
  - A pain, because OPC UA is powerful and thus complex

Our findings

- From a non expert point of view:
  - ➢ Writing an OPCUA server from scratch can be tedious
  - ➢ Developers are not necessarily the hardware integrators.
- We have to make the hardware integration easier ➔
  - ➢ Hide the OPCUA knowledge
  - ➢ Limit the code to be written
  - ➢ Provide a formalized description of the HW to be integrated
    - Data points description (set, get, …)
    - Hardware connection type (USB , RS232, TCPIP, …)
    - Data Transport (UDP, TCP, …)
    - Data transfer (PUSH, PULL)
    - Start Conditions and End conditions (Security aspects ?)
    - Other things?
- Share the description between OPCUA server and client

OPCUA Server Implementation model

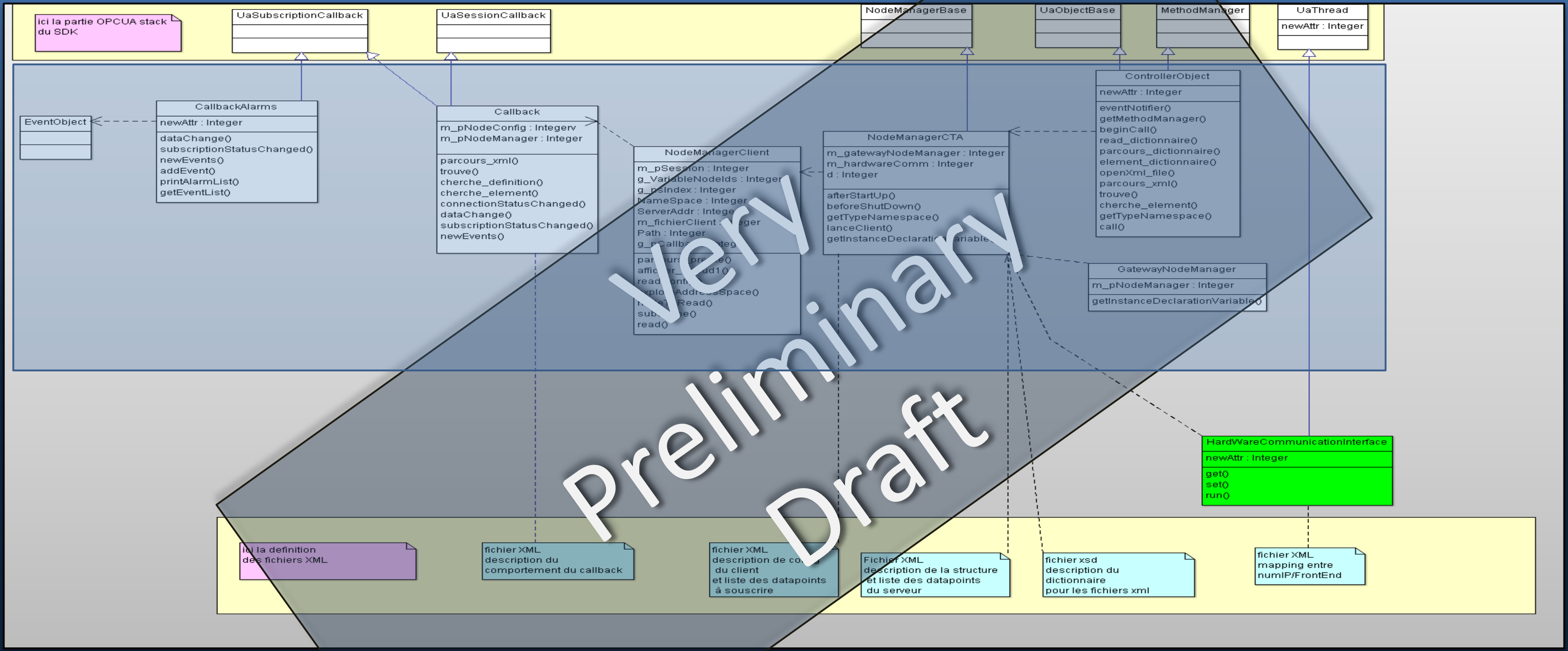

OPC UA Layer

CTA
OPC UA Layer

HARDWARE Access Generic Layer

- 2 options :
  1. The generic part can work with a description and some standard access method :
     1. connect(…), read(…), write(…), close(…)
     2. The input/output data stream is also described to analyze the data flow.
  2. The generic part is purely abstract and the hardware integration is done by inheriting the abstract part.
     1. connect(…), read(…), write(…), close(…),… have to be overridden.
- The “CTA OPCUA Layer” will manage always the same object types.

OPCUA Server :UML model



ici la partie OPCUA stack du SDK

UaSubscriptionCallback

UaSessionCallback

NodeManagerBase

UaObjectBase

MethodManager

UaThread
newAttr : Integer

EventObject

CallbackAlarms
newAttr : Integer
dataChange()
subscriptionStatusChanged()
newEvents()
addEvent()
printAlarmList()
getEventList()

Callback
m_pNodeConfig : Integerv
m_pNodeManager : Integer
parcours_xml()
trouve()
cherche_definition()
cherche_element()
connectionStatusChanged()
dataChange()
subscriptionStatusChanged()
newEvents()

NodeManagerClient
m_pSession : Integer
g_VariableNodeIds : Integer
g_nsIndex : Integer
NameSpace : Integer
ServerAddr : Integer
m_fichierClient : Integer
Path : Integer
g_pCallback : Integer
parcours_preuve()
afficher_add1()
read_config()
exploreAddressSpace()
nodeToRead()
subscribe()
read()

NodeManagerCTA
m_gatewayNodeManager : Integer
m_hardwareComm : Integer
d : Integer
afterStartUp()
beforeShutDown()
getTypeNamespace()
lanceClient()
getInstanceDeclarationVariable()

ControllerObject
newAttr : Integer
eventNotifier()
getMethodManager()
beginCall()
read_dictionnaire()
parcours_dictionnaire()
element_dictionnaire()
openXml_file()
parcours_xml()
trouve()
cherche_element()
getTypeNamespace()
call()

GatewayNodeManager
m_pNodeManager : Integer
getInstanceDeclarationVariable()

HardWareCommunicationInterface
newAttr : Integer
get()
set()
run()

ici la definition des fichiers XML

fichier XML description du comportement du callback

fichier XML description de config du client et liste des datapoints à souscrire

Fichier XML description de la structure et liste des datapoints du serveur

fichier xsd description du dictionnaire pour les fichiers xml

fichier XML mapping entre numIP/FrontEnd

- From such a model :
  - We are able to provide a OPCUA Server hiding the tedious part of the implementation
  - Based on a hardware description(XML/XSD, …), we are able to build the full representation of a device in the OPCUA Server.
  - We are able to write the implementation code (Java , C++, …) in coherent way with respect to the model (class description content and relationship)

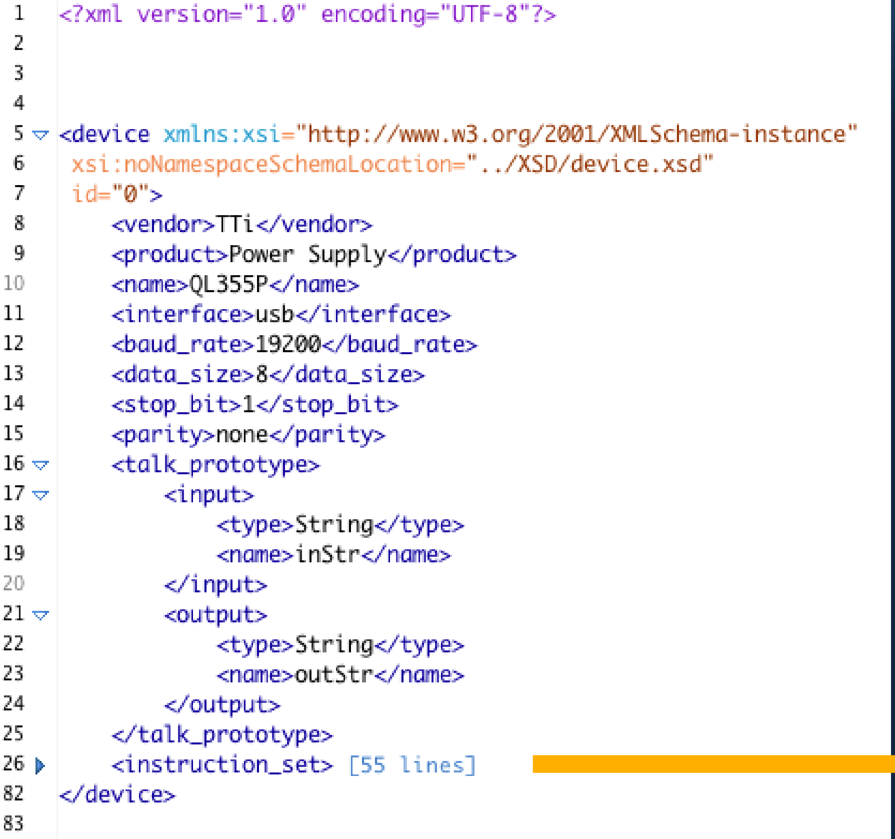
OPCUA Server Description

- Based on XML and XSD files
  - XML Schema description(XSD) files are the dictionary and validate the XML files.

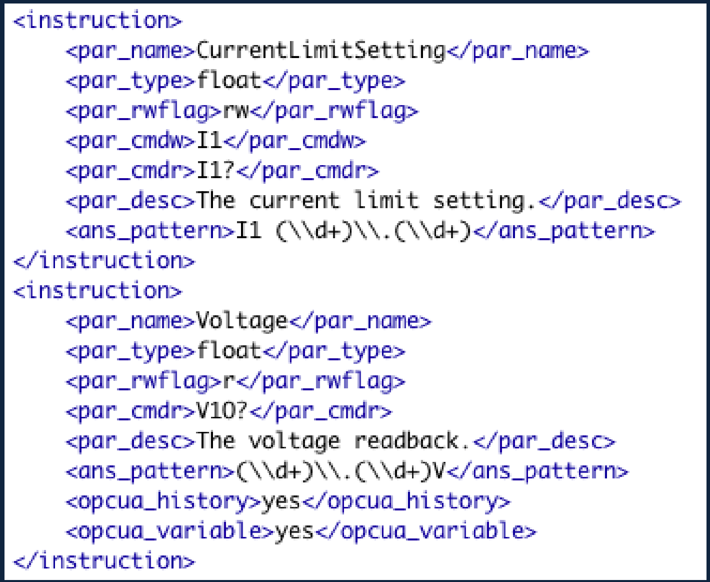
- The XML files are used as input to :
  - Describe the hardware connection
  - Populate the data points into the OPCUA server
  - Describe the hardware data stream
    - → Analyze the data stream
    - → Allow to locate the data point name and value.

Some details: XML files

- Device description: QL355P.xml by J. Bolmont



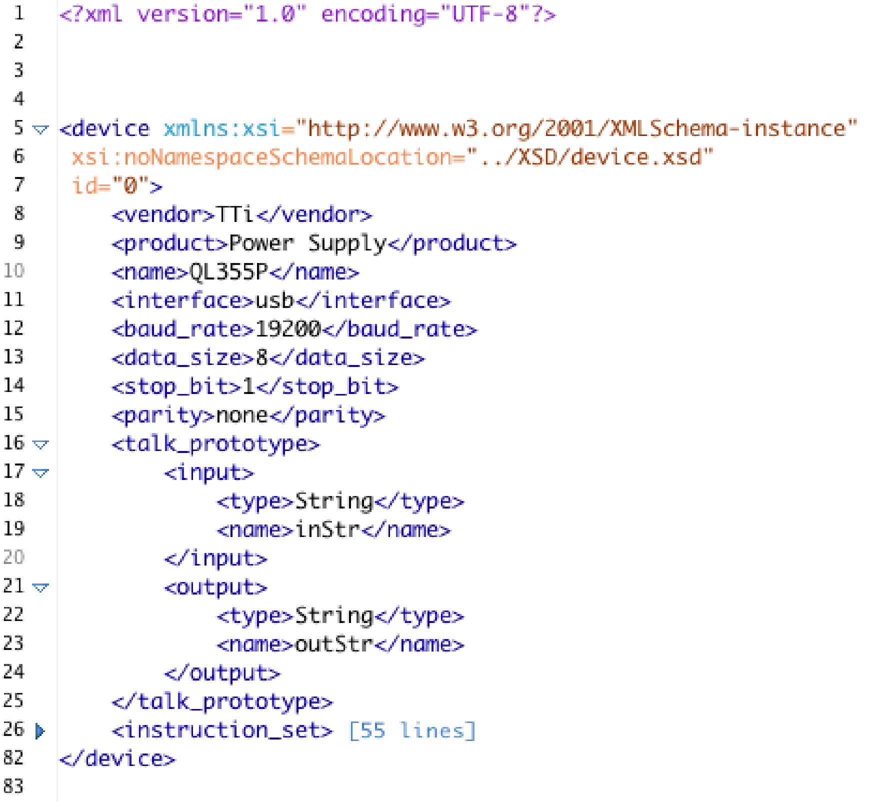
```xml
<?xml version="1.0" encoding="UTF-8"?>



<device xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../XSD/device.xsd"
 id="0">
    <vendor>TTi</vendor>
    <product>Power Supply</product>
    <name>QL355P</name>
    <interface>usb</interface>
    <baud_rate>19200</baud_rate>
    <data_size>8</data_size>
    <stop_bit>1</stop_bit>
    <parity>none</parity>
    <talk_prototype>
        <input>
            <type>String</type>
            <name>inStr</name>
        </input>
        <output>
            <type>String</type>
            <name>outStr</name>
        </output>
    </talk_prototype>
    <instruction_set> [55 lines]
</device>
```



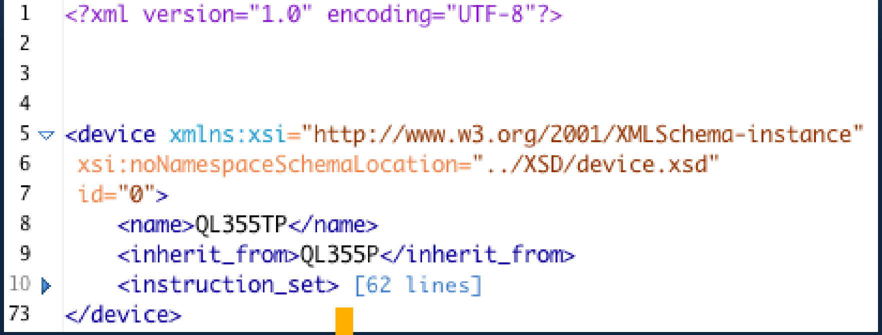
```xml
<instruction>
    <par_name>CurrentLimitSetting</par_name>
    <par_type>float</par_type>
    <par_rwflag>rw</par_rwflag>
    <par_cmdw>I1</par_cmdw>
    <par_cmdr>I1?</par_cmdr>
    <par_desc>The current limit setting.</par_desc>
    <ans_pattern>I1 (\\d+)\\.(\\d+)</ans_pattern>
</instruction>
<instruction>
    <par_name>Voltage</par_name>
    <par_type>float</par_type>
    <par_rwflag>r</par_rwflag>
    <par_cmdr>V1O?</par_cmdr>
    <par_desc>The voltage readback.</par_desc>
    <ans_pattern>(\\d+)\\.(\\d+)V</ans_pattern>
    <opcua_history>yes</opcua_history>
    <opcua_variable>yes</opcua_variable>
</instruction>
```

Some details: XML files

- Example of a device (QL355TP) inherited from another device (QL355P)



```
<?xml version="1.0" encoding="UTF-8"?>



<device xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../XSD/device.xsd"
 id="0">
    <vendor>TTi</vendor>
    <product>Power Supply</product>
    <name>QL355P</name>
    <interface>usb</interface>
    <baud_rate>19200</baud_rate>
    <data_size>8</data_size>
    <stop_bit>1</stop_bit>
    <parity>none</parity>
    <talk_prototype>
        <input>
            <type>String</type>
            <name>inStr</name>
        </input>
        <output>
            <type>String</type>
            <name>outStr</name>
        </output>
    </talk_prototype>
    <instruction_set> [55 lines]
</device>
```

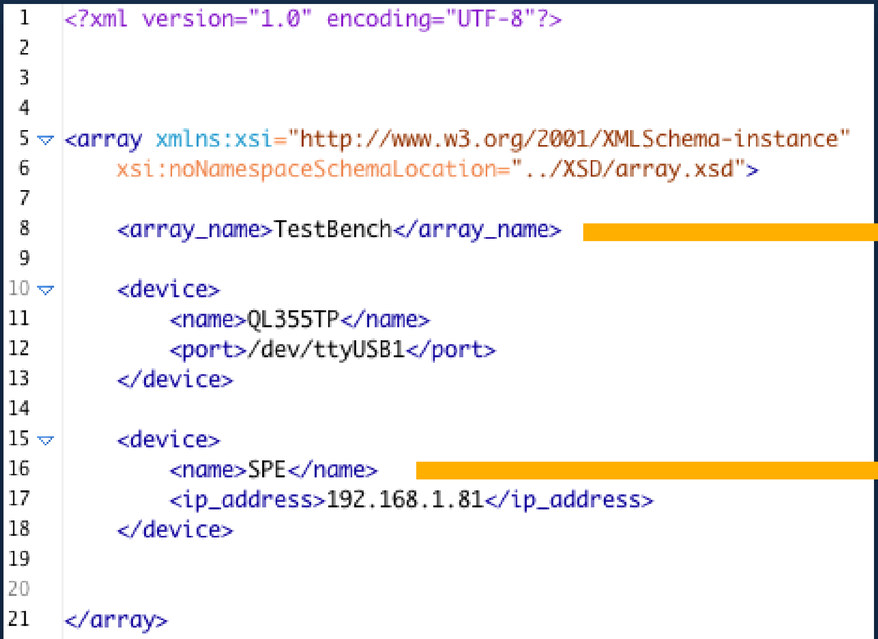


```
<?xml version="1.0" encoding="UTF-8"?>



<device xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../XSD/device.xsd"
 id="0">
    <name>QL355TP</name>
    <inherit_from>QL355P</inherit_from>
    <instruction_set> [62 lines]
</device>
```

Supplementary instructions for functions specific to the child device. All the other attributes are inherited.

Some details: XML files

- Testbench description: testbench.xml



```
<?xml version="1.0" encoding="UTF-8"?>



<array xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../XSD/array.xsd">

    <array_name>TestBench</array_name>

    <device>
        <name>QL355TP</name>
        <port>/dev/ttyUSB1</port>
    </device>

    <device>
        <name>SPE</name>
        <ip_address>192.168.1.81</ip_address>
    </device>


</array>
```

1 array =
1 OPC-UA server

For each device, a description file is expected, here “SPE.xml”

Conclusions

- Does this approach fit with the current status and orientation of the development?
  - ➢ If yes :
    - Future plan :
      - – Enrich the UML model :
        - ○ Should be the result of the part of ACTL group working on this topic(Hardware integration)
      - – Improve the XSD if needed.
      - – Try to use this environment for the current prototypes in progress.
- OPCUA current limit : license and pricing :
  - ➢ How can we share licenses among CTA Group ?
  - ➢ Price negocation …