Plant Floor Communications Integration using a Low Cost CPPS Architecture

Marcelo V. García[1], Edurne Irisarri[1], Federico Pérez[1], Elisabet Estévez[2], Darío Orive[1], Marga Marcos[1]
[1]University of the Basque Country, UPV/EHU, Spain
{mgarcia294, edurne.irisarri, federico.perez, dario.orive, marga.marcos}@ehu.eus
[2]University of Jaen, Spain
eestevez@ujaen.es

***Abstract*- Currently, factory automation systems need to cope with very different challenges, such as technological advances happening very fast, global and competitive market or product customization. These challenges lead to a new generation of automation systems based on the so-called Cyber-Physical Production Systems (CPPS) globally connected to form a flexible System of Cyber-Physical Production Systems (SoCPPS). CPPSs require acquisition of production system data and smart data processing to extract information to improve the overall system performance. To achieve that it is needed to bridge the gap between the control systems and higher layers. This paper discusses an approach to use the IEC 61499 function block concept to exchange data between plant floor and higher layers using an industrial standard like OPC UA. The OPC UA server offers subscription mechanisms, making possible the integration of several resources residing at plant floor. As it runs on embedded devices, the proposal makes possible to acquire plant information at low cost, enabling at the same time, a component-based design for enterprise plant floor control with independence of the hardware platform used.**

Keywords—Cyber-Physical Production Systems (CPPS), OPC UA, IEC 61499, Industry 4.0.

I. Introduction

Current conventional centralized plant floor control systems are not using the latest promising technologies like message network protocols, smart sensors and actuators and new production technologies [1]. On the other hand, communication and information technologies such as cloud computing, intelligent data processing or big data, require new services from plant floor control systems to achieve global performance [2]. In the context of factory automation, traditional plant floor control runs autonomously making local decisions based on local data.

In order to achieve the goals that the Industry 4.0 concept envisions, secure communication links and cooperation of all participants across companies involved on the entire lifecycle of the product are needed.

In this context, the OPC UA (IEC 62541) standard is a promising alternative being a service oriented architecture, offering data security and reliable information models. Unfortunately, as far as authors know, it is not extensively used probably because of a lack of accessible platforms. [3]

The Industry 4.0 concept is based on the global integration of CPPS devices. In smart factory both Industrial IoT (IIoT) and CPPS concepts are converging to the Internet of Services, which uses the cloud-based manufacturing for creating, publishing, and sharing the services that represent manufacturing processes, and could be offered by virtual enterprises [4]. IIoT defines a series of technologies, being Machine to Machine communication (M2M) the one that enables data exchange between two remote machines. The use of M2M would allow to implement context-aware machines with an increase of efficiency in different industrial processes, leading to cost savings and economies of scale [5].

On the other hand, the IEC 61499 standard promotes a model based development of distributed control systems. In Industry 4.0 will allow to model and develop software and hardware for independent distributed control systems [6]. The key entity in the IEC 61499 standard is the Function Block (FB), which wraps the control and communication algorithms in different programming languages, including IEC 61131, Java, C++, or virtually any other programming language. The Service Interface Function Block (SIFB) is one of the types of FBs provided that allows wrapping and abstracting the hardware access as well as the resources from the Application Programmer Interface. In summary, currently available technology is mature enough as to achieve the goal of M2M but there is still a gap between technology and real industries.

This work presents a low cost platform and an OPC UA server and clients easily configurable. Thus, it allows an easy to deploy software platform aiming at collecting process data via M2M communications. The approach focuses on integrating traditional Modbus/TCP industrial networks within an OPC UA server over the IEC 61499 standard. In order to fill the gap between the technology and real factories, a low-cost platform in which it is possible to prototype inter CPPS communications based on the proposed OPC UA server and client.

The layout of the paper is as follows, Section II proposes a set of SIFB for IEC 61499, describing in detail the configuration XML file for the OPC UA server. Section III illustrates its use in a case study where the OPC UA server and the set of SIFBs are implemented for plant floor communication. Finally, some conclusions and ongoing work are drawn in Section IV.

II. SIFB SET FOR IEC 61499

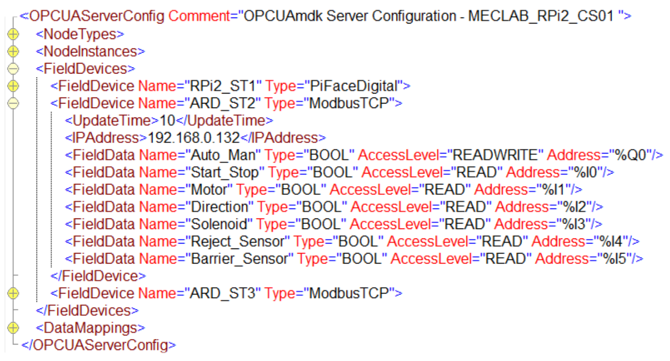
The 4DIAC-IDE framework has been used to create a set of SIFBs, which is based in previous researches [7], [8]. In this work the new version of the OPC UA server is configured using an information model, implemented using a XML configuration file, which improves the integration with field devices and their process data. In this way, the server includes a driver for Modbus TCP/IP protocol. By other hand, a subscription mechanism over OPC UA is implemented.

A. SIFB OPCUA_SERVER

This SIFB allows configuring the OPC UA server through a XML file. This file includes all essential parameters for the OPC UA server such as URL address, vendor name, server name, version, etc. Also, this configuration file states Address Space with their Node Types and Node Instances, among others. Within the configuration two more elements are defined:

a) *Field Devices*: This element groups the definition of the devices accessible by the server, characterized by the supported communication and related information. Process data elements (*FieldData*) supplied for the device are also defined.

b) *Data Mapping*: This element defines the existing relationships between *DataVariables* declared in the address space section of the *FieldDevice*, and the associated *FieldData*.

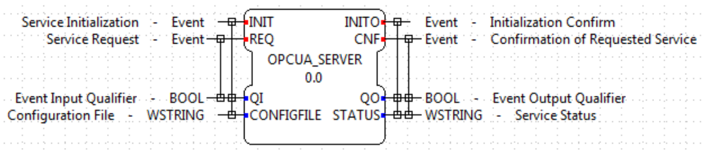


```
<OPCUAServerConfig Comment="OPCUAmdk Server Configuration - MECLAB_RPi2_CS01 ">
  <NodeTypes>
  <NodeInstances>
  <FieldDevices>
    <FieldDevice Name="RPi2_ST1" Type="PiFaceDigital">
    <FieldDevice Name="ARD_ST2" Type="ModbusTCP">
      <UpdateTime>10</UpdateTime>
      <IPAddress>192.168.0.132</IPAddress>
      <FieldData Name="Auto_Man" Type="BOOL" AccessLevel="READWRITE" Address="%Q0"/>
      <FieldData Name="Start_Stop" Type="BOOL" AccessLevel="READ" Address="%I0"/>
      <FieldData Name="Motor" Type="BOOL" AccessLevel="READ" Address="%I1"/>
      <FieldData Name="Direction" Type="BOOL" AccessLevel="READ" Address="%I2"/>
      <FieldData Name="Solenoid" Type="BOOL" AccessLevel="READ" Address="%I3"/>
      <FieldData Name="Reject_Sensor" Type="BOOL" AccessLevel="READ" Address="%I4"/>
      <FieldData Name="Barrier_Sensor" Type="BOOL" AccessLevel="READ" Address="%I5"/>
    </FieldDevice>
    <FieldDevice Name="ARD_ST3" Type="ModbusTCP">
  </FieldDevices>
  <DataMappings>
</OPCUAServerConfig>
```

Fig. 1: Field Device Configuration

This SIFB provides common events like INIT, REQ, INITO and CNF (Fig. 2) as well as the following input and output data:

- QI (BOOL): This input data works jointly with INIT to start-up or shut-down the OPC UA server. On INIT event, if QI is TRUE, OPC UA server starts; otherwise, if OI is FALSE, OPC UA server shutdowns.
- CONFIGFILE (WSTRING): This input data holds the XML configuration filename.
- QO (BOOL): Informs about the performance of last procedure executed.
- STATUS (STRING): Reports status information of the server.

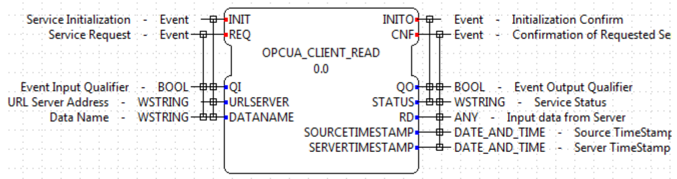


Service Initialization - Event
Service Request - Event
INIT
REQ
INITO
CNF
Event - Initialization Confirm
Event - Confirmation of Requested Service
OPCUA_SERVER
0.0
Event Input Qualifier - BOOL
Configuration File - WSTRING
QI
CONFIGFILE
QO
STATUS
BOOL - Event Output Qualifier
WSTRING - Service Status

Fig. 2: SIFB OPCUA_SERVER

B. SIFB OPCUA_CLIENT_READ

This SIFB implements an OPC UA client able to read in variables from the OPC UA server.



Service Initialization - Event
Service Request - Event
INIT
REQ
INITO
CNF
Event - Initialization Confirm
Event - Confirmation of Requested Se
OPCUA_CLIENT_READ
0.0
Event Input Qualifier - BOOL
URL Server Address - WSTRING
Data Name - WSTRING
QI
URLSERVER
DATANAME
QO
STATUS
RD
SOURCETIMESTAMP
SERVERTIMESTAMP
BOOL - Event Output Qualifier
WSTRING - Service Status
ANY - Input data from Server
DATE_AND_TIME - Source TimeStamp
DATE_AND_TIME - Server TimeStamp

Fig. 3: SIFB OPCUA_CLIENT_READ

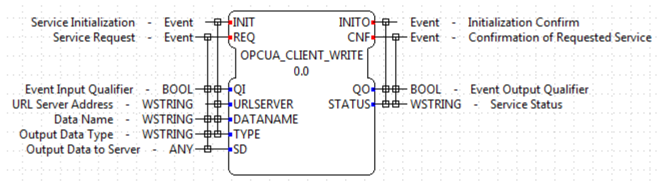
This SIFB includes the following data:

- URLSERVER (WSTRING): URL of the OPC UA server.
- DATANAME (WSTRING): Name of the variable data available from the OPC UA server.
- RD (ANY): Data provided by the OPC UA server to which the client is connected. The type of this parameter is ANY to improve reuse.
- SOURCETIMESTAMP (DATE AND TIME): Timestamp associated to the source resource item.
- SERVERTIMESTAMP (DATE AND TIME): Timestamp associated to the OPC UA server item.

C. SIFB OPCUA_CLIENT_WRITE

This SIFB provides variable writing service, having as parameters:

- TYPE (WSTRING): Data type for the value to be written.
- SD (ANY): Data value to be written. Again, the type of this parameter is ANY in order to increase generality.



Service Initialization - Event
Service Request - Event
INIT
REQ
INITO
CNF
Event - Initialization Confirm
Event - Confirmation of Requested Service
OPCUA_CLIENT_WRITE
0.0
Event Input Qualifier - BOOL
URL Server Address - WSTRING
Data Name - WSTRING
Output Data Type - WSTRING
Output Data to Server - ANY
QI
URLSERVER
DATANAME
TYPE
SD
QO
STATUS
BOOL - Event Output Qualifier
WSTRING - Service Status

Fig. 4: SIFB OPCUA_CLIENT_WRITE

D. SIFB OPCUA_CLIENT_SUBSCRIBE

This SIFB implements a single subscription to monitor variables in OPC UA servers. The subscription maintains a local copy of the parameters for the monitored item. These local copies can be altered by updating their properties without affecting the state on the server. This SIFB input and output parameters are:

- MODE (WSTRING): three monitoring modes are supported: "REPORTING", "SAMPLING" and "DISABLED".
- PERIOD (ULINT): Set the sampling period in millisecond for sampled monitored items.
- RD (ANY): Value of the monitored item provided by the OPC UA server.

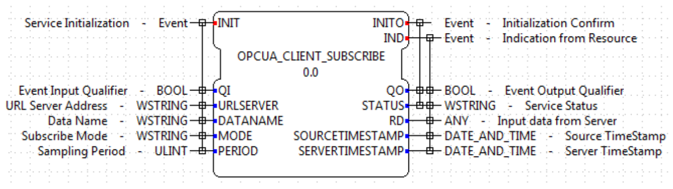

Service Initialization - Event
INIT
INITO
IND
Event - Initialization Confirm
Event - Indication from Resource
OPCUA_CLIENT_SUBSCRIBE
0.0
Event Input Qualifier - BOOL
URL Server Address - WSTRING
Data Name - WSTRING
Subscribe Mode - WSTRING
Sampling Period - ULINT
QI
URLSERVER
DATANAME
MODE
PERIOD
QO
STATUS
RD
SOURCETIMESTAMP
SERVERTIMESTAMP
BOOL - Event Output Qualifier
WSTRING - Service Status
ANY - Input data from Server
DATE_AND_TIME - Source TimeStamp
DATE_AND_TIME - Server TimeStamp

Fig. 5: SIFB OPCUA_CLIENT_SUBSCRIBE

III. Case Study

The proposed case study describes a laboratory system aiming at illustrating a scale factory automation application. In particular, the production plant is an assembly line with three FESTO® stations as depicted in Fig. 6. The *Handling station* gets the work pieces to be processed from an input position and leaves them in a ramp that feeds next station; the *Conveyor station* transports and selects the pieces; whereas the *Stack station* stores and separates work pieces.

The low-cost hardware and software architectures comprises a Raspberry PI 2 model B as process controller with an OPC UA server, a Beagle Bone Black as monitoring system, and two Arduino UNO boards as input/output distributed peripheral Modbus/TCP slaves.

A. Hardware Architecture

Raspberry PI 2 model B (RPi2) is a SBC (Single Board Compute) that allows running several Linux distributions. In this case, a Debian Jessie release (Raspbian) is used. This platform integrates a 900MHz quad-core ARM Cortex-A7 processor with 1 GB of RAM. This board runs a Debian Jessie release. The RPi2 also provides a GPIO with 40 pins for interacting with the physical world. Other features include 4 USB ports, 10/100 Ethernet port, Micro SD card slot.

Beagle Bone Black (BBB) is another SBC running a Debian Jessie Linux distribution. The BBB comprises a General Purpose I/O Port (GPIO) with 69 pins for interacting with the physical world. It is based on a 1GHz processor, a 512MB onboard DDR3 RAM, an AM3358 microprocessor. USB ports, 10/100 Ethernet among other features are included.

The Arduino UNO board is used to collect variables from plant floor. Arduino UNO has the ability to receive analog or digital inputs from a variety of sensors and to control actuators. This is possible by means of its 14 digital I/Os and 4 analog inputs.

However, in order to make possible working with industrial signals, expansion board is required. In this case, a board providing 8 digital inputs and 8 digital outputs of 24V has been used.

B. Software Platform

The main elements in the software platform are the IEC 61499 framework and the OPC UA client/server architecture.

1) IEC 61499

4DIAC is a IEC 61499 compliant framework for developing applications running on heterogeneous platforms [9]. In particular, in this work the 4DIAC-IDE has been used as development environment for distributed control applications. This framework provides portability, interoperability as well as configuration capabilities. The run-time used is FORTE (4DIAC-RTE), which allows executing IEC 61499 applications on top of small embedded devices. FORTE provides portability to several operating systems including Windows and Linux. Whereas 4DIAC-IDE can only run on the programming platform, the FORTE IEC 61499 run-time can run on any device of the distributed automation system.

2) OPC UA

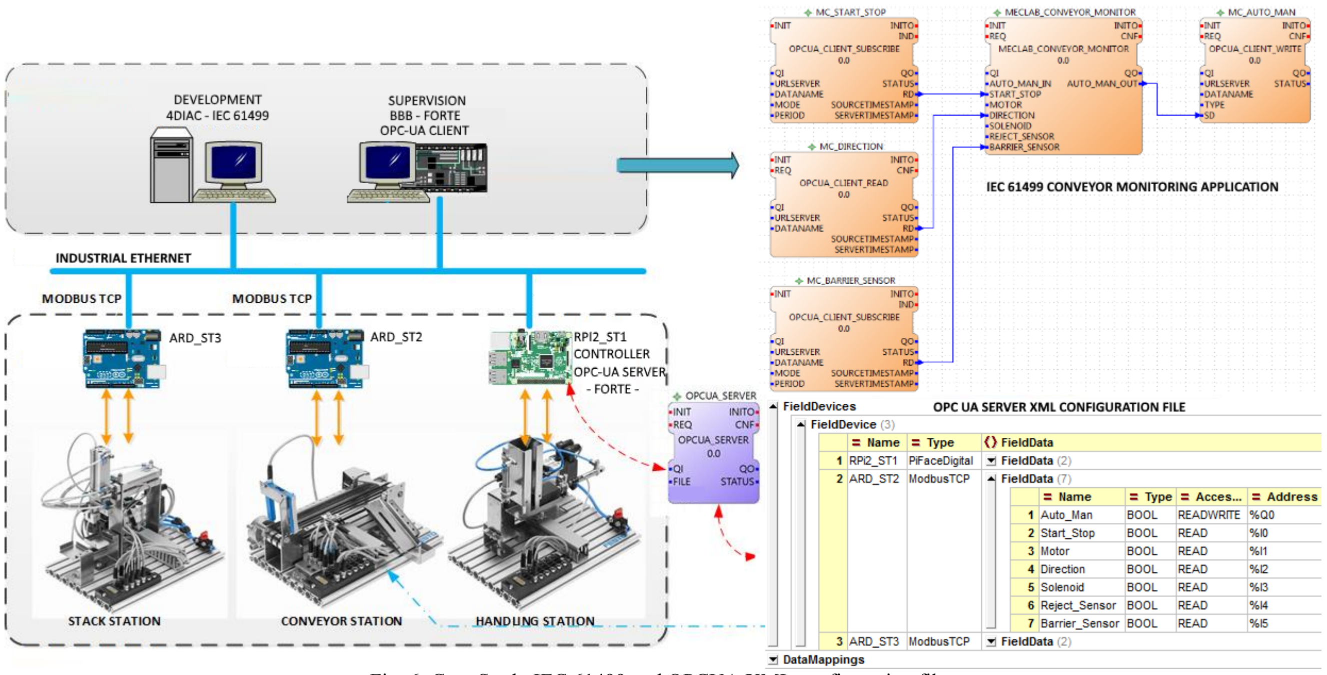
OPC UA represents the evolution of previous standards such as OPC DA, OPC A&E and OPC HDA in order to provide a vendor independent open architecture [10]. Relating to portability issues, the OPC UA server uses a portable communication stack that can be directly used in automation systems. OPC UA provides a reliable, robust and high performance communication means, suitable for industrial automation applications and devices, i.e. for CPPS.

OPC UA provides, more than a transport method, as it includes a modeling mainstay able to support plant floor models. OPC UA, besides a transport method, offers information access through plant floor models.

The OPC UA server uses a model-based representation of the process that consists of a set of objects that makes available to clients. These objects represent the underlying real-time process data [11]. This approach introduces the concept of process model for a better adaptation to the needs of modern industrial applications. Typically, a profile is designed for the vertical integration of automation systems. The use of an OPC UA architecture allows a full description of any system data, independently of their complexity.

C. Functionality

As depicts Fig. 6, the *Stack Station* and *Conveyor Station* have associated Modbus/TCP slaves implemented by Arduino boards. The RPi2 gets access directly to the *Handling station* I/Os and as Modbus/TCP master to the other stations I/Os. Fig. 6 also depicts the XML configuration file of the OPC UA server to get access the tags in local RPi2 and remote slaves as Modbus/TCP master.



FieldDevices
FieldDevice (3)

	Name	Type	FieldData
1	RPi2_ST1	PiFaceDigital	FieldData (2)
2	ARD_ST2	ModbusTCP	FieldData (7)
3	ARD_ST3	ModbusTCP	FieldData (2)

FieldData (7)

	Name	Type	Acces...	Address
1	Auto_Man	BOOL	READWRITE	%Q0
2	Start_Stop	BOOL	READ	%I0
3	Motor	BOOL	READ	%I1
4	Direction	BOOL	READ	%I2
5	Solenoid	BOOL	READ	%I3
6	Reject_Sensor	BOOL	READ	%I4
7	Barrier_Sensor	BOOL	READ	%I5

DataMappings

Fig. 6: Case Study IEC 61499 and OPCUA XML configuration file

The RPi2 controls the three stations and integrates the OPC UA server. It executes a FORTE runtime providing the commented SIFBs to perform the control of the whole process and the management of the OPC UA server. Remote OPC UA clients, like the supervisory application in the BBB, can read/write or subscribe process data by means the above SIFB set. For instance, Fig. 6 shows the IEC 61499 supervisory *Conveyor station* application.

The OPC UA server and client functionalities are integrated in an own library implemented using a C++ stack. The OPC UA library includes access to Modbus/TCP slaves and RPi2 GPIO. At the same time, this OPC UA library has been integrated in the FORTE runtime. In this way, server and client functionalities are embedded in the runtime.

IV. CONCLUSIONS

This work presents an approach for accessing field data in automation systems using OPC UA servers in low-cost CPPS architecture using IEC 61499 applications running over FORTE runtime. The use of this kind of systems helps to introduce new CPPS concepts within the Industry 4.0 paradigm. This architecture provides a M2M infrastructure for plant floor communications.

By other hand, an information model implemented using a XML configuration file, eases the integration between field devices and Address Space in OPC UA for factory automation systems.

A set of SIFBs are proposed that can be used to implement OPC UA servers and clients, including subscription mechanisms. These Function Blocks may be used for building new applications with IEC 61499 frameworks in a fast and easy way.

Future works focus on developing new functionalities such as integrate OPC UA models with other industrial standards like ISA 95 or AML what will improve CPPS integration.

ACKNOWLEDGMENT

This work was financed in part by the University of the Basque Country (UPV/EHU) under project UFI 11/28, by the MCYT&FEDER under project DPI2015-68602-R, and by the Ministry of Education and Research of Ecuador through grant SENESCYT-2014.

REFERENCES

[1] K. Takahashi, K. Yokoyama, and K. Morikawa, “Integrating Lean and Agile Strategies into the Production Control System for Mixed- model Production Lines,” vol. 246, pp. 405–412.

[2] K. Windt, F. Böse, and T. Philipp, “Autonomy in production logistics: Identification, characterisation and application,” *Robot. Comput. Integr. Manuf.*, vol. 24, no. 4, pp. 572–578, 2008.

[3] N. Jazdi, “Cyber physical systems in the context of Industry 4.0,” *Autom. Qual. Testing, Robot. 2014 IEEE …*, pp. 2–4, 2014.

[4] C. Scheuermann, S. Verclas, and B. Bruegge, “Agile Factory - An Example of an Industry 4.0 Manufacturing Process,” *3rd IEEE Int. Conf. Cyber-Physical Syst. Networks Publ.*, vol. 2008, p. 5, 2015.

[5] J. Kim, J. Lee, J. Kim, and J. Yun, “M2M service platforms: Survey, issues, and enabling technologies,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 61–76, 2014.

[6] “International Electrotechnical Commission Std. (2005) IEC 61499: Function blocks, Part 1-4,” 2014. [Online]. Available: http://www.iec.ch.

[7] M. V. Garcia, F. Perez, I. Calvo, and G. Moran, “Developing CPPS within IEC-61499 based on low cost devices,” *IEEE Int. Work. Fact. Commun. Syst. - Proceedings, WFCS*, vol. 2015-July, 2015.

[8] M. V. Garcia, F. Perez, I. Calvo, and G. Moran, “Building industrial CPS with the IEC 61499 standard on low-cost hardware platforms,” *19th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2014*, 2015.

[9] 4DIAC Consortium. PROFACTOR GmbH, “Framework for Distributed Industrial Automation and Control (4DIAC),” 2010. [Online]. Available: http://www.fordiac.org.

[10] D. Van Der Linden, H. Mannaert, W. Kastner, V. Vanderputten, H. Peremans, and J. Verelst, “An OPC UA interface for an evolvable ISA88 control module,” *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, 2011.

[11] A. Claassen, S. Rohjans, and S. Lehnhoff Member, “Application of the OPC UA for the Smart Grid,” *IEEE PES Innov. Smart Grid Technol. Conf. Eur.*, pp. 1–8, 2011.