Desarrollo de Comunicación OPC-UA en Sistemas CPPS de Bajo Costo

Marcelo V García Sánchez
DISA (Basque Country University)
48013 – Bilbao, España
mgarcia294@ikasle.ehu.es

***Resumen* – Los requisitos de flexibilidad y de reconfiguración de las fábricas y plantas industriales del futuro pueden ser parcialmente satisfechos por la adopción de tecnologías de comunicación nuevas como OPC-UA y normas de automatización como IEC-61499 que ya están disponibles en investigaciones. Actualmente los Sistemas de Producción Cyber-Físicos (CPPs) se consideran como el núcleo de los futuros sistemas de control y automatización distribuidos. Usarlos en plataformas hardware de bajo costo en donde converjan todas estas nuevas tecnologías será de gran importancia para su uso y comercialización en plantas industriales a futuro. En este trabajo se presenta el diseño de un sistema de control distribuido empotrado realizado en una placa Raspberry PI y una tarjeta de expansión PiFace, la cual entrega datos a un servidor OPC-UA todo dentro de la norma IEC-61499, aplicamos esta norma ya que provee un alto nivel de diseño, permitiendo combinar componentes de software de una manera sencilla independientemente del hardware utilizado. Este artículo aborda el diseño de aplicaciones IEC-61499 en la tarjeta Raspberry PI**

***Abstract* - Requirements of flexibility and reconfiguration of factories and industrial facilities of the future may be partially satisfied by the adoption of new communication technologies like OPC-UA and automation standards as IEC-61499 which are now available for research. Currently Cyber-Physical Production Systems (CPPs) are considered as the core of future control systems and distributed automation. Use in low-cost hardware platforms where converge all these new technologies will be of great importance for use in industrial plants and marketing in the future. This paper shows the design of an embedded distributed control system on a tiny single-board computer like Raspberry PI and using a expansion card like PiFace, which delivers data to an OPC-UA server, all within the IEC-61499 standard, we use this standard because give a high level of design, allowing re-use software's components in a easy way regardless of the hardware used. This article discusses the design of IEC-61499 applications in the Raspberry PI card**

Keyword: Industrial Cyber-Physical Systems, OPC-UA, IEC-61499, Raspberry PI.

I. INTRODUCCIÓN

La Industria 4.0 y el Internet de las cosas (IoT) son conceptos que requieren la existencia y utilización de redes de comunicación de alta velocidad, alta seguridad y de transferencia de datos especializada entre dispositivos de control y servicios asociados de alto nivel. Los protocolos de comunicación actuales, especialmente la norma en desarrollo OPC-UA, y sistemas de control basados en dispositivos CPPS a nivel industrial hacen que resulten idóneos para ejecutar los principios de la Industria 4.0.

Durante las últimas décadas, varios avances en las tecnologías informáticas y de la comunicación han llevado a la miniaturización de los equipos, la reducción de su costo y la disponibilidad de redes de gran ancho de banda. Algunos autores afirman que estos avances han traído una nueva revolución en el dominio de los sistemas de control y de las comunicaciones, siendo necesario la introducción de nuevas abstracciones y métodos que ayudan a los ingenieros a diseñar este tipo de sistemas, conocidos actualmente como Sistemas de Producción Cyber-Físicos (CPPS) [1]. Sin embargo, la adopción de estas técnicas en la industria se ha retrasado, en parte debido a la tendencia a utilizar hardware propietario, que dificulta la introducción de los últimos avances.

Los conceptos anteriormente descritos permiten una producción dinámica, la incorporación de una mayor flexibilidad e individualización de los procesos de fabricación, permitiendo que todos los sistemas estén en capacidad de comunicarse directamente entre sí: sensores, chips de comunicación, controladores, PLCs, HMIs, Sistemas de Ejecución de Producción (MES) y Sistemas de Planificación de Recursos Empresariales (ERP). Esto facilita a los ingenieros de automatización permitiéndoles desarrollar sistemas con estas nuevas técnicas y por otro lado se reduce el tiempo de adopción de estos conceptos a nivel industrial debido al fácil uso de prototipos.

Durante varios años el estándar IEC-61131 ha sido la principal norma en el ámbito de la automatización industrial, ya que estandariza los lenguajes de programación en este campo y ha permitido crear sistemas de producción que fuesen más flexibles y reconfigurables [2]. Sin embargo, esta norma se ha centrado en los algoritmos de control y no en problemas de comunicación. Es por esto, que como complemento se ha empezado a adoptar la norma IEC-61499 la cual tiene como objetivo modelar y desarrollar sistemas de control distribuido independientes del hardware utilizado [3]. La unidad central de esta norma es el Bloque de Función (FB) [4], el cual permite encapsular los algoritmos de control o comunicación y pueden ser escritos en leguajes propios de IEC-61131 o de alto nivel como JAVA, C++, etc [5]. Dentro de los varios tipos de FBs que esta norma provee se encuentra el Bloque de Función de Interfaz de Servicio (SIFB) como un mecanismo para encapsular y abstraer el acceso al hardware de la programación de la aplicación

Para esta investigación, se ha visto recomendable el usar comunicación OPC Arquitectura Unificada (OPC-UA) bajo la norma IEC-61499, OPC-UA contiene varias mejoras importantes, por ejemplo, arquitectura orientada a servicios, seguridad de datos y modelos de información configurables y

provee una conveniente vía para integrar sistemas de gestión de procesos industriales directamente con el nivel de control, todo a través de una interface común [6]. Sin embargo, todavía no hay muchas experiencias de cómo IEC-61499 debe utilizarse con OPC-UA y cuáles serían los beneficios reales de este.

El objetivo del presente artículo es desarrollar e implementar comunicación OPC-UA modelado bajo la norma IEC-61499 para su uso en sistemas de control CPPS especialmente en plataformas ARM los cuales estas siendo usados frecuentemente en sistemas de control industrial, para que interactúen con SCADASs, ERPs.

El presente artículo está organizado de la siguiente manera: en la sección II se indica las plataformas hardware y software usadas en la implementación, sección III muestra la implementación de un caso de estudio usando la norma IEC-61499 y OPC-UA para ilustrar la aplicación de la investigación y finalmente algunas conclusiones son mostradas en la sección IV.

II. IMPLEMENTACIÓN PROPUESTA

A. Plataforma de Hardware

La presente investigación está basada en la adopción de una plataforma hardware de bajo costo. En este trabajo se ha usado un Ordenador de Placa Reducida (SBC) como es la tarjeta Raspberry Pi [7]. Aunque en un inicio fue desarrollada con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas, actualmente se ve su posible uso a nivel del control industrial.

Las especificaciones técnicas son:

TABLE I. CARACTERÍSTICAS RASPBERRY PI MODELO B

SoC	Broadcom BCM2835
CPU	ARM 1176JZFS a 700 MHz
RAM	512 MB
Periféricos	8xGPIO, SPI, I^2C, UART
USB	2 x USB 2.0
Redes	Ethernet 10/100

Existen tres distribuciones de Linux como sistema operativo empotrado: Raspbian "wheezy" (basada en Debian), Arch Linux ARM y QtonPi. Posee un puerto de Entradas y Salidas de propósito general (GPIO) de 26 pines, el nivel de voltaje con el cual trabaja esta tarjeta es 3.3 V. La comunicación con el GPIO no es directamente soportado por el kernel de cualquiera de los sistemas operativos que existen para Raspberry PI, para ello hay librerías de licencia abierta las cuales son utilizadas para dicha comunicación utilizando el puerto SPI para este objetivo.

Debido a que Raspberry Pi no fue construida para interactuar con sistemas de control es por esto, que se utiliza la placa de entradas/salidas PiFAce (Figura 1) que posee: 8 salidas digitales de colector abierto, 8 entradas digitales, 2 relés, 4 interruptores. [8]

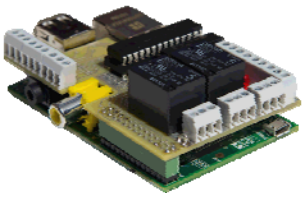

Figura 1: PiFace para Raspberry Pi

B. Plataforma de Software

1) IEC-61499

El enfoque del desarrollo de software se basa en el uso de 4DIAC [9] como framework para la implementación de la aplicación. La herramienta de desarrollo seleccionada es 4DIAC-IDE, que proporciona un entorno de ingeniería flexible para el modelado de aplicaciones de control distribuido a través de múltiples plataformas con la norma IEC-61499, destinada a proporcionar portabilidad, interoperabilidad y capacidad de configuración. El entorno de tiempo de ejecución es denominado FORTE (4DIAC-RTE), que es un runtime compilado de IEC-61499 para pequeños dispositivos embebidos, el cual es implementado en C++ y posee portabilidad a varios sistemas operativos como Windows, Linux, NET+OS®7 o eCos. Mientras 4DIAC-IDE sólo se ejecuta en la plataforma de desarrollo, FORTE se debe ejecutar en todos los nodos del sistema de automatización distribuido como runtime.

Para compilar FORTE o el Stack de Comunicaciones OPC-UA y que la aplicación sea compatible con otros dispositivos hardware que tengan un kerrnel de Linux independientemente de su versión se ha usado como toolchain de compilación Linaro para arquitecturas ARM. Básicamente Linaro aplica la esencia del desarrollo OpenSource, y nos permite brindar la portabilidad a nuestro código fuente.

2) OPC-UA

OPC-UA está dirigido a sustituir las diversas normas OPC como OPC-DA, OPC-HE, OPC-A&E por una arquitectura abierta. Similar a su predecesor OPC-UA sigue un protocolo Cliente/Servidor. En cuanto a la portabilidad, la principal diferencia es que el servidor está utilizando una pila de comunicación portátil que puede ser (y a menudo es) aplicado directamente en el Sistema de Automatización. OPC-UA permite una comunicación fiable, robusta, de gran rendimiento adecuado para la automatización industrial en CPPS. OPC-UA se diseñó para soportar codificación binaria para el intercambio de datos de alto rendimiento. Ofrece comunicaciones fiables ya que ha incorporado mecanismos capaces de afrontar problemas como pérdidas de mensajes, alta latencia, etc. La funcionalidad de las normas anteriores es compilada y agregada en un único estándar OPC-UA, que proporciona un conjunto reducido y único de accesos genéricos de servicio a toda la información. [10]

El contenido de un servidor OPC-UA es un sistema basado en modelos. Es compatible y soporta el concepto de modelos de datos que se pueden definir para que coincida con las necesidades de una aplicación. Mientras que la OPC clásica

tiene un meta-modelo muy sencillo que proporciona etiquetas en una jerarquía sencilla, OPC-UA ofrece un modelo de información rico con técnicas orientadas a objetos [11]. Un modelo de datos en OPC-UA puede ser publicado como un perfil de llamada, proporcionando una interfaz estandarizada para los clientes. Típicamente un perfil está diseñado para la integración vertical de Sistemas de Automatización.

Los perfiles amplían los medios de comunicación que OPC-UA define (es decir cómo conectar y comunicarse con el servidor), proporcionando información sobre cómo están organizados los datos y como se los puede acceder. Además de los modelos y perfiles, OPC-UA añade el concepto de máquinas de estado y los programas. Los programas son funciones que se pueden ejecutar mediante la invocación a través de OPC-UA. Este es un mecanismo de llamada a procedimiento remoto (RPC) o para ser más precisos, un mecanismo para la comunicación remota dispuesto por sistemas como CORBA, o COMAND. OPC-UA no es compatible con OPC clásica, ya que utiliza tecnologías diferentes, sin embargo OPC Foundation proporciona wrappers y proxies que o bien adaptan de forma automática los servidores existentes a los clientes OPC-UA o proporcionan un servidor proxy a los clientes de OPC clásica. [12]

La arquitectura OPC-UA permite suministrar el modelo de información, lo que quiere decir que la meta-información se proporciona junto con los datos. Un modelo de información puede incluir objetos, variables, eventos, descripciones de máquinas de estados, programas e incluso, el histórico de variables y eventos. De este modo, cualquier sistema, por complejo que sea, puede ser descrito en su totalidad mediante dichos mecanismos orientados a objetos. Por lo general, los sistemas CPPS no pueden implementar todas las opciones del modelado descrito anteriormente, pero con la información detallada en el Address Space puede ser suficiente para su uso a nivel industrial.

3) *Conjunto de SIFBs para IEC-61499*

El entorno de desarrollo 4DIAC-IDE ha sido usado para crear un conjunto de SIFBs los cuales encapsulan las operaciones de entrada/salida y la comunicación OPC-UA del sistema de control distribuido por medio del uso de la tarjeta de expansión PiFace para la Raspberry PI bajo la norma IEC-61499. Se tienen dos archivos de configuración uno para las entradas/salidas de la tarjeta PiFace y otro para la configuración inicial de la comunicación OPC-UA los cuales son realizados en formato XML.

Las entradas/salidas de la tarjeta PiFace son direccionadas de igual manera usando la configuración guardada en el archivo XML (Figura 2) el cual lista todas las variables del proceso que pueden accederse en la tarjeta. El grupo de los tags de las variables poseen la siguiente información: el nombre que permite acceder a la variable del programa, el pin del puerto GPIO de la Raspberry PI referido a dicha variable, el tipo de dato y una pequeña descripción del dato, con esto se realizará un referenciado dinámico y de acuerdo a los procedimientos utilizados actualmente por la mayoría de equipos de control a nivel industrial.

```
<?xml version="1.0" encoding="UTF-8"?>
- <esquema>
  - <variable1>
      <nombre>1S1</nombre>
      <pin>25</pin>
      <tipo>BOOL</tipo>
      <descripcion>Sensor Cilindro Horizontal 1</descripcion>
    </variable1>
  - <variable2>
      <nombre>1S2</nombre>
      <pin>24</pin>
      <tipo>BOOL</tipo>
      <descripcion>Sensor Cilindro Horizontal 2</descripcion>
    </variable2>
  - <variable3>
      <nombre>2S1</nombre>
      <pin>23</pin>
      <tipo>BOOL</tipo>
      <descripcion>Sensor Cilindro Vertical 1</descripcion>
```

Figura 2: Archivo XML de configuración de entradas/salidas de PiFace

a) *SIFB PiFace_DO*

Este SIFB es usado para acceder a los datos de entrada que están conectadas a la tarjeta de expansión PiFace. Posee los eventos usuales de entrada/salida como son INIT, CNF o QI (Figura 3). El significado de las entradas y salidas adicionales se describen a continuación:

- FILE (STRING): el nombre del archivo de configuración XML.
- TAG (STRING): Nombre de la variable que se necesita acceder en la placa RPI.
- DO (BOOL): El dato a ser despachado a la tarjeta PiFace.
- STATUS (STRING): Describe un código de error si se ha producido.

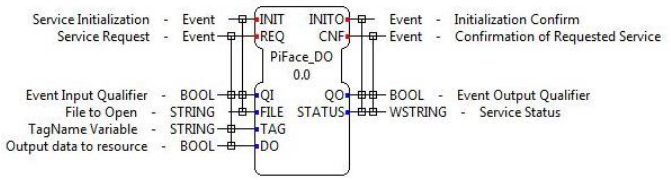

Figura 3: SIFB PiFACE_DO

b) *SIFB PiFace_DI*

Similar al SIFB previo, el PiFace_DI habilita el acceso de las variables de entrada conectadas a la tarjeta PiFace.(Figura 4) También encontramos los eventos que por norma se encuentran en SIFB como son INIT, CNF o QI así como conectores adicionales que se explican a continuación:

- FILE (STRING): Archivo de configuración XML.
- TAG (STRING): Nombre de la variable que se necesita acceder en la placa RPI.
- STATUS (STRING): Describe un código de error si se ha producido.
- DI (BOOL): El dato a ser leído de la tarjeta PiFace.

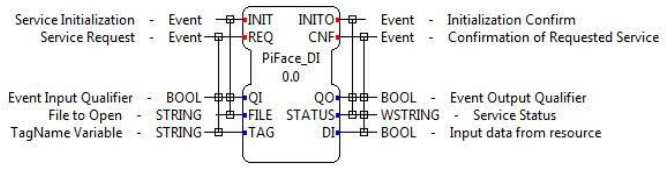

Figura 4: SIFB PiFACE_DI

a) *SIFB OPCUA_CONFIG*

Este SIFB permite la configuración básica del Servidor OPC-UA a ser utilizado, se realiza utilizando un archivo XML (Figura 5), en dicho archivo se lista lo esencial para el funcionamiento como: la dirección URL que usara el servidor con su puerto, nombre del fabricante, nombre del servidor, versión, tipo de encriptación, certificados de seguridad utilizados, número de clientes permitidos, etc

```
    <SamplingRate>5000</SamplingRate>
    <SamplingRate>10000</SamplingRate>
  </AvailableSamplingRates>
  <!--Settings for the thread pools used in the server application -->
  <ThreadPoolSettings>
    <MinSizeTransactionManager>1</MinSizeTransactionManager>
    <MaxSizeTransactionManager>4</MaxSizeTransactionManager>
    <MinSizeSubscriptionManager>1</MinSizeSubscriptionManager>
    <MaxSizeSubscriptionManager>4</MaxSizeSubscriptionManager>
  </ThreadPoolSettings>
  <!--Build information for the server application software -->
  <ApplicationUri>urn:OPCUA_DISA:OPCUADemoserver</ApplicationUri>
  <ManufacturerName>GSIS DISA ETSI</ManufacturerName>
  <ApplicationName>C++ OPCUADemoserver</ApplicationName>
  <SoftwareVersion>1.3.2</SoftwareVersion>
  <BuildNumber>200</BuildNumber>
  <!--Build information end -->
  <ServerUri>urn:[NodeName]:OPCUA_DISA:OPCUADemoserver</ServerUri>
  <ServerName>OpcUADemoServer@[NodeName]</ServerName>
  <!--OPCUA test Server for FORTE -->
  <RejectedCertificatesDirectory>[ApplicationPath]/PKI/CA/rejected</RejectedCertificatesDirectory>
  <UaEndpoint>
    <SerializerType>Binary</SerializerType>
    <Url>opc.tcp://[NodeName]:8080</Url>
    <IsVisible>true</IsVisible>
    <IsDiscoveryUrl>true</IsDiscoveryUrl>
```

Figura 5: Archivo de Configuración XML

El SIFB creado posee los eventos de entrada y salida usuales encontrados en la mayoría de SIFBs como son INIT, CNF o QI, el significado de las entradas y salidas es descrita a continuación (Figura 6):

- FILE (STRING): Nombre del archivo de configuración XML
- STATUS (STRING): Secuencia de caracteres que nos indica si el archivo fue leído o existe algún error.

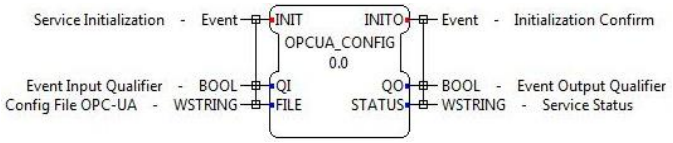

Figura 6: SIFB OPCUA_CONFIG

b) *SIFB OPCUA_SERVER_PIFACE*

Este SIFB implementa la comunicación OPC-UA para que la Raspberry Pi sirva los datos a la red de comunicación (Figura 7), e igual que todos los anteriores SIFBs poseen eventos de entrada/salida como INIT, REQ, CNF y datos de entrada y salida que se explica a continuación:

- NODENAME (STRING): Nombre del Nodo del servidor OPC-UA el cual será disponible para los clientes;
- VARIABLE_NAME(STRING): Nombre de la variable a ser creada dentro del Nodo del Servidor OPC-UA para ser servida a la red de comunicación, facilitando de esta manera su búsqueda por un cliente.
- IN_DATA (ANY): Variables de entrada del servidor OPC-UA que será despachada a la red de comunicaciones para compartir con los clientes suscritos. Es de tipo ANY ya que el servidor al ser genérico puede ser utilizado para despachar cualquier tipo de variable
- STATUS (STRING): Secuencia de caracteres que nos indica si el archivo fue leído o existe algún error.
- SERVERTIMESTAMP (DATE AND TIME): Indica la fecha y hora que se ha modificado el valor de los datos del servidor.
- OUT_DATA (ANY): Proporciona el dato que ha sido escrito en el servidor desde los clientes conectados al mismo. Al igual que con la entrada es de tipo ANY ya que puede ser utilizada para escribir cualquier tipo de variable dependiendo de las necesidades del control.

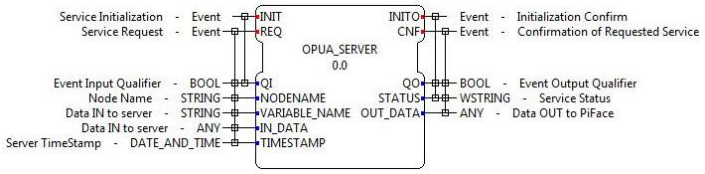

Figura 7: SIFB OPCUA_SERVER_PIFACE

III. CASO DE ESTUDIO

Después de crear y compilar los SIFBs previamente explicados con FORTE, se encuentran disponibles para su uso en aplicaciones de control distribuido bajo la norma IEC-61499 con la tarjeta Raspberry PI

En esta sección se explica cómo estos bloques pueden ser usados para crear sistemas distribuidos de comunicación usando la norma OPC-UA, la aplicación está compuesta por tres células de producción industrial y un cliente OPC-UA realizado en Windows que permite visualizar el estado de las entradas y salidas de las PiFace que realizan el control de las estaciones de producción. El diagrama del caso de estudio se indica en la Figura 8. Se tiene una estación de manipulación la cual recoge piezas y entrega a la cinta transportadora utilizando un brazo accionado por dos cilindros de doble efecto, y finalmente la estación de almacenamiento, lleva las piezas clasificadas anteriormente a un almacén para su posterior distribución.

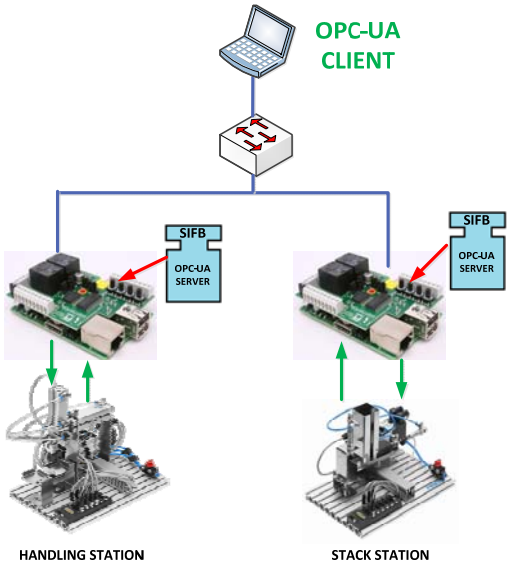

Figura 8: Diagrama del Caso de Estudio

Esta aplicación involucra dos runtimes FORTE que se está ejecutando en cada una de las Raspberry PI que realiza el control de cada estación, en dichas tarjetas adicionalmente se despacha las variables a un Servidor OPC-UA para el monitoreo por el cliente que reside en el ordenador de visualización bajo un ambiente Windows. Lo planteado podemos observarlo en el Diagrama de Bloques de la Figura 9

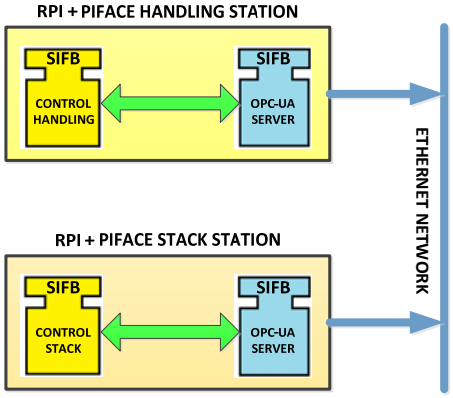

Figura 9: Diagrama de Bloques Comunicación OPC-UA en IEC-61499

Cada aplicación local de Control del Sistema Distribuido ha sido descargada a cada una de las tarjetas Raspberry PI que trabaja con el runtime FORTE. La Figura 10 muestra el proceso que se está ejecutando para el control y comunicación de las variables de la estación de cinta de transporte, el valor actualizado de las variables de entrada la está entregando al Servidor OPC-UA para que el cliente remoto pueda saber en qué posición se encuentra los elementos de control. Adicionalmente se puede escribir datos por parte del cliente al servidor y realizar cierto control desde el Cliente OPC-UA

En esta aplicación los SIFBs anteriormente descritos permiten una transferencia de la señales de los sensores al Raspberry PI y Servidor OPC-UA sin retrasos, permitiendo de esta manera una fácil construcción de sistemas de control

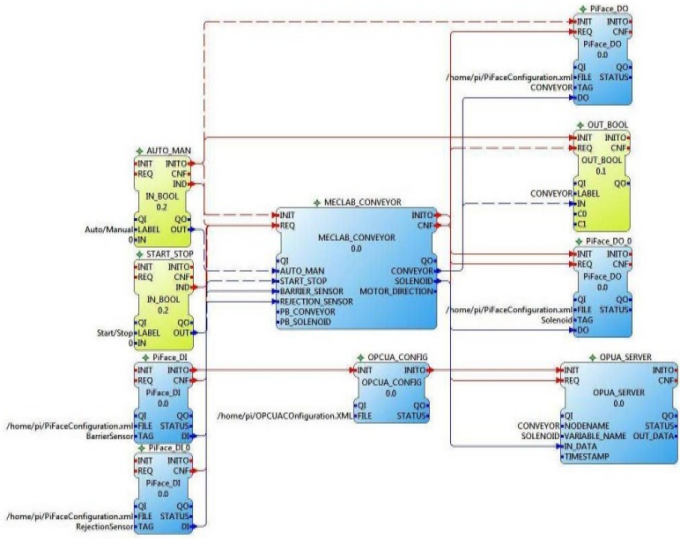

Figura 10: Aplicación IEC-61499 del servidor OPC-UA

Actualmente existen soluciones comerciales de servidores/clientes OPC-UA entre las cuales podemos citar los siguientes fabricantes: OPC-UA Matrikon suite, OPC-UA Unified Automation, OPC-UA Wapice, etc. Todos estos poseen varias características como uso de certificados de seguridad, múltiples conexiones de clientes, uso en sistemas bajo plataforma Linux, pero están desarrollados bajo lenguaje C++ y en la norma IEC-61131 que es la que actualmente la mayoría de sistemas de automatización utilizan. Es por esto que lo desarrollado y expuesto en el presente artículo es una solución en las comunicaciones industriales para que la norma IEC-61499 pueda empezar a ser utilizado a nivel de empresas o plantas de producción y ofrezca iguales capacidades o mejores que las soluciones comerciales ya que permite reutilización de los SIFBs, configuración usando archivos XML, fácil programación y uso por ingenieros de control, entre las principales características.

IV. CONCLUSIONES Y TRABAJO FUTURO

Se ha desarrollado un primer acercamiento de un Servidor OPC-UA bajo la norma IEC-61499 en una plataforma de bajo costo que puede actuar como un sistema CPPS. El uso de este tipo de plataformas a nivel industrial es una gran alternativa en mediciones rápidas de procesos reales donde el aplicar los métodos tradicionales es muy costoso y el prototipado rápido de este tipo de equipos ayuda a su uso en esos ámbitos. Al integrar este tipo de comunicaciones que se están abriendo campo en el área industrial permite que los sistemas CPPS puedan ya ser visto como una solución deseada para la construcción de sistemas de control distribuidos en empresas industriales

En este trabajo se ha logrado la integración de comunicación OPC-UA bajo la norma IEC-61499, utilizando el runtime FORTE en sistemas de bajo costo. Se ha diseñado un conjunto de SIFBs con los cuales se puede realizar la comunicación OPC-UA de las entradas conectadas a la tarjeta PiFace de la RPI. Estos SIFBs podrán ser usados para diseñar e implementar de una manera fácil nuevas aplicaciones de control usando 4DIAC-IDE de amplio uso a nivel educativo.

En el futuro se tiene mucho trabajo por desarrollar ya que la norma OPC-UA ofrece capacidades de modelado y seguridad que debe ser implementado de una manera eficaz en la norma IEC-61499, entre los principales retos se han identificados los siguientes: (1) Modelo del Adress Space del OPC-UA bajo norma IEC-61499, (2) Integrar seguridad en el servidor OPC-UA, (3) Integrar nuevas funcionalidades y SIFBS en sistemas CPPS bajo arquitectura ARM.

REFERENCES

[1] CPS Reference ARTIST2. (2006) Roadmap on Real-Time Techniques in Control System Implementation – Control for Embedded Systems Cluster. [Online]. www.artistembedded.org

[2] International Electrotechnical Commission. (1993) IEC 61131-3 Programmable Controllers - Part 3: Programming Languages. [Online]. www.iec.ch

[3] International Electrotechnical Commission Std. (2005) IEC 61499: Function blocks, Part 1-4. [Online]. www.iec.ch

[4] Lewis R. W, "Modeling control systems using IEC 61499," in *IEEE Publishing*, 2001, ISBN: 0 85296 796 9.

[5] Vyatkin V Zoitl A., "IEC 61499 Architecture for Distributed Automation: the "Glass Half Full" View," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 7-23, 2009.

[6] S. Rohjans, and S. Lehnhoff A. Claassen, "Application of the OPC UA for the smart grid," in *2nd IEEE PES*, Ny, USA, 2011.

[7] EBEN U. Raspberry Pi User guide. [Online]. http://www.myraspberry-pi.org/wp-content/uploads/2013/02/Raspberry.Pi_.User_.Guide_.pdf

[8] 14element, PiFace Digital, Started Guide, 2014, pp:1-2.

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

[10] D., Mannaert, H., Kastner, W., Vanderputten, V., Peremans, H., & Verelst, J. van der Linden, "An OPC UA interface for an evolvable ISA88 control module," *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference*, pp. 1-7, 2011.

[11] M. Uslar, and H.-J. Appelrath. S. Rohjans, "OPC UA and CIM: Semantics for the smart grid," in *IEEE PES Transmission and Distribution*, 2011.

[12] Ilkka Seilonen, Antti Tuomi, Kari Koskinen Jouko Virta, "SOA-Based Integration for Batch Process Management with OPC UA and ISA-88/95," *15th IEEE International Conference on Emerging Technologies and Factory Automation*, ISBN: 978-1-4244-6850-8/10, 2010.