UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA INDUSTRIAL EN PROCESOS DE AUTOMATIZACIÓN

Tema:

"DESARROLLO DE UN SISTEMA DE CONTROL INDUSTRIAL BASADO EN EL ESTÁNDAR IEC-61499"

Proyecto de Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero Industrial en Procesos de Automatización.

SUBLÍNEA DE INVESTIGACIÓN: Sistemas de control automatizados e instrumentos virtuales para procesos industriales de baja y alta potencia.

AUTOR: Sergio Patricio Bustos Pulluquitin.

TUTOR: Ing. Marcelo Vladimir García Sánchez, Mg.

Ambato – Ecuador

Octubre 2017

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de investigación sobre el tema: "Desarrollo de un Sistema de Control Industrial basado en el Estándar IEC-61499", realizado por el señor Sergio Patricio Bustos Pulluquitin, estudiante de la Carrera de Ingeniería Industrial en Procesos de Automatización de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato Octubre, 2017

EL TUTOR

MARCELO VLADIMIR GARCIA SANCHEZ

Firmado digitalmente por MARCELO VLADIMIR GARCIA SANCHEZ
Nombre de reconocimiento (DN): c=EC, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, l=QUITO, serialNumber=0000172836, cn=MARCELO VLADIMIR GARCIA SANCHEZ
Fecha: 2017.10.04 15:47:50 -05'00'

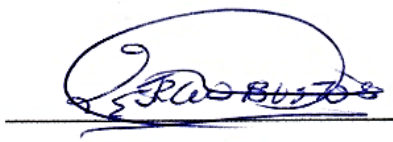
Ing. Marcelo Vladimir García Sánchez.

AUTORÍA

El presente trabajo de investigación titulado "Desarrollo de un Sistema de Control Industrial basado en el Estándar IEC-61499", es absolutamente original, auténtico y personal en tal virtud, el contenido, efectos legales y académicas que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato Octubre, 2017

EL AUTOR



Sergio Patricio Bustos Pulluquitin

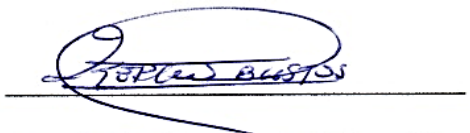
CC: 180465252-5

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato Octubre, 2017



Sergio Patricio Bustos Pulluquitin

CC: 180465252-5

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing.Mg. César Rosero e Ing. Mg. Benjamín Pusay, revisó y aprobó el Informe Final del Proyecto de Investigación titulado "DESARROLLO DE UN SISTEMA DE CONTROL INDUSTRIAL BASADO EN EL ESTÁNDAR IEC-61499", presentado por el señor BUSTOS PULLUQUITIN SERGIO PATRICIO de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. Mg. Pilar Urrutia

PRESIDENTA DEL TRIBUNAL

Ing. Mg. César Rosero

DOCENTE CALIFICADOR

Ing. Mg. Benjamín Pusay

DOCENTE CALIFICADOR

DEDICATORIA

El presente proyecto de investigación se lo dedico a mis queridos padres Patricio y Solanda por su apoyo incondicional, sus valores inculcados, sus consejos y por ser los pilares fundamentales de mi vida.

Sergio Patricio Bustos Pulluquitin

AGRADECIMIENTOS

A mi padre y madre por toda la confianza depositada en mí y estar conmigo en todo momento.

Al Ing. Marcelo García por encaminarme de una manera adecuada en el transcurso del proyecto.

A los docentes de Facultad de Ingeniería en Sistemas, Electrónica e Industrial por todos conocimientos enseñados durante mi ciclo universitario.

A la Unidad Operativa DIDE por darme la oportunidad de formar parte del proyecto de investigación.

Sergio Patricio Bustos Pulluquitin

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente proyecto se realizó la implementación de un Sistema de Control Industrial basado en el Estándar IEC_61499 integrando el entorno de desarrollo 4DIAC (4DIAC-IDE) y el runtime FORTE (4DIAC-RTE) en dispositivos embebidos de bajo costo, centrándonos en la BeagleBone Black (BBB), este sistema de control se implementó en el proceso de nivel de líquido en el módulo FESTO® MPS-PA Compact Workstation del laboratorio de hidráulica y neumática de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato. Con la presente implementación del sistema de control se pretende verificar los beneficios que ofrece la norma IEC-61499.

ABSTRACT

In the present project the implementation of an Industrial Control System based on the IEC_61499 Standard was made integrating the development environment 4DIAC (4DIAC-IDE) and the FORTE (4DIAC-RTE) runtime into low cost embedded devices, focusing on the BeagleBone Black (BBB), this control system was implemented in the liquid level process in the FESTO® MPS-PA Compact Workstation module of the hydraulic and pneumatic laboratory of the Faculty of Systems, Electronics and Industrial Engineering of the Technical University of Ambato. With the present implementation of the control system is intended to verify the benefits offered by the standard IEC-61499.

GLOSARIO DE TÉRMINOS Y ACRÓNIMOS

Sintaxis: Conjunto de reglas que debemos seguir para que el compilador sea capaz de reconocer nuestro programa

Semántica: Significado de un programa o funciones.

FB: Bloque de Función

FBB: Bloque de Función Básico

SIFB: Bloque de Función de Interfaz de Servicio

ECC: Gráfico Control de Ejecución

BBB: BeagleBone Black

IPMCS: Sistemas de Control y Medición de Procesos Industriales Distribuidos

HMI: Interfaz Hombre Máquina

INTRODUCCIÓN

Los procesos de fabricación y producción se realizan cada vez más por sistemas y soluciones automatizadas y en consecuencia el nivel de automatización en las fábricas y plantas aumenta de manera constante. A medida que el nivel absoluto de automatización aumenta, también lo hace la complejidad por el creciente número de sensores, actuadores, controladores o PLCs de diferentes fabricantes que se utilizan en las plantas y sistemas de fabricación, por lo tanto requisitos de interoperabilidad, capacidad de configuración y portabilidad son difíciles de alcanzar en los sistemas constituidos por elementos tan diversos[1].

En el campo de los sistemas de automatización industrial, las crecientes demandas de control descentralizado y el crecimiento exponencial de la complejidad de control han acelerado el paso de paradigmas de diseño monolíticos y centralizados hacia enfoques de ingeniería más reconfigurables y distribuidos. Esta tendencia se ha reflejado en el desarrollo de la norma IEC-61499 para sistemas de medición y control de procesos industriales distribuidos. La norma IEC-61499 tiene como objetivo establecer un marco de desarrollo abierto, orientado a componentes y plataforma independiente para mejorar la reutilización, re-configurabilidad, interoperabilidad, portabilidad y distribución de software de control para sistemas distribuidos complejos[2].

Los sistemas de control distribuido han demostrado ser ideales para la implementación de sistemas de control de variables complejas en procesos industriales. Su concepción se desliga del control centralizado, con el objetivo de dar autonomía a cada una de las partes constitutivas del proceso, al no depender de una única fuente de procesamiento de información[3].

CAPÍTULO 1

EL PROBLEMA

1.1. Tema

“Desarrollo de un Sistema de Control Industrial basado en el Estándar IEC-61499”.

1.2. Planteamiento del problema

Mundialmente los sistemas de automatización industrial se han basado durante muchos años en la Estándar IEC-61131[4], la cual es considerada como una de las más importantes en la automatización industrial[5] y ha permitido crear sistemas de producción más flexibles y reconfigurables[6]. La norma IEC-61131 es el primer paso en la estandarización de los autómatas programables y sus periféricos, el objetivo básico de esta norma durante varios años fue la estandarización de los lenguajes de programación para aplicaciones de automatización industrial [1]. Sin embargo, se afirma que actualmente la norma no aborda las nuevas exigencias de los sistemas industriales complejos, que incluyen la portabilidad, la interoperabilidad, el aumento de la reutilización y la distribución[5]. Además la portabilidad, la configurabilidad, la interoperabilidad, la reconfiguración y la distribución se han identificado en[7] y [8] como las exigencias y requisitos de alto nivel para futuros sistemas de automatización.

La norma IEC 61131-3 define guías para aplicación de bloques funcionales como elemento base para la comunicación de sistemas que procesan información, algunos sistemas industriales de manufactura con arquitecturas sencillas podrían modelarse con los bloques funcionales definidos en la IEC 61131-3, no obstante la complejidad que

caracteriza los sistemas de manufactura en la actualidad obligó a un replanteo y desarrollo de nuevos componentes de los bloques funcionales que respondan al dinamismo y versatilidad en las aplicaciones[3].

La norma IEC-61131 específica la sintaxis y semántica del lenguaje de programación, sin embargo la semántica de estos lenguajes no está definida estrictamente, por lo que esto conlleva a la incompatibilidad del software de control entre los diferentes fabricantes[1]. Por lo tanto no es posible transferir la configuración de una herramienta a otra y de esta manera preservar toda la información requerida para una ejecución correcta del algoritmo de control[1], Adicionalmente en el aspecto de la reconfigurabilidad la IEC-61131 no define los medios para crear dinámicamente nuevos recursos en una configuración, así como no hay definiciones para el intercambio de algoritmos sobre la marcha[1].

En la actualidad la industria ecuatoriana experimenta cambios en los procesos de automatización, impulsado por el mejoramiento constante de diferentes técnicas de producción, optimización de uso de materias primas, ahorro energético y mejoramiento de la calidad de los productos[9], todo esto se lograría con la implementación de sistemas automatizados de control distribuidos basados en el estándar IEC-61449 en las plantas manufactureras.

1.3. Delimitación del problema

1.3.1. De Contenido

Campo: Ingeniería Industrial en Procesos de Automatización.

Área académica: Electrónica.

Línea de investigación: Automatización.

Sublínea de investigación: Sistemas de control automatizados e instrumentos virtuales para procesos industriales de baja y alta potencia.

1.3.2. Espacial

La presente proyecto se desarrolló en la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato, ubicado en Av. Los Chasquis y Río Cutuchi del cantón Ambato de la provincia de Tungurahua.

1.3.3. Temporal

El presente proyecto se desarrolló en los períodos académicos octubre 2016 - marzo 2017 y abril 2017– septiembre 2017 luego que se aprobó por el H. Consejo Directivo de la Facultad.

1.4. Justificación

El estándar IEC-61499 fue creado para sistemas de control distribuido, incluyendo su arquitectura y los requisitos de herramientas de software. Se desarrolló como consecuencia del creciente interés en las nuevas tecnologías y arquitecturas para crear la próxima generación de sistemas industriales y teniendo como base el estándar IEC-61131[6]. Define una arquitectura genérica y una guía para el uso del FB en Sistemas de Control y Medición de Procesos Industriales Distribuidos (IPMCSs)[1]. El estándar propone una arquitectura basada en FB que se definen como la unidad estructural básica de los modelos[10], los FB han estado cada vez más presente en la programación de sistemas de automatización y control industrial. A esto ha contribuido en gran medida el uso cada vez más extendido del estándar IEC 61131-3, que tiene a los FBs como uno de sus conceptos de programación básicos[11].

Uno de los principales objetivos del estándar IEC-61499 es promover el desarrollo de sistemas heterogéneos compuestos de dispositivos de control de diferentes fabricantes y adicional permitiendo la reconfiguración dinámica, es decir cambiar la configuración de un sistema mientras la aplicación de control continúa ejecutándose. La IEC-61499 es visto como la siguiente generación de estándares en sistemas de automatización y está diseñado para cubrir interoperabilidad, portabilidad y reconfigurabilidad que no están contemplados en la norma IEC-61131-3[1].

Es factible realizar el presente proyecto porque los materiales necesarios para la implementación del sistema de control industrial se encuentran disponibles en los laboratorios de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, además es parte de proyecto de investigación del DIDE titulado: "MAXIMIZACIÓN DE PROCESOSO PRODUCTIVOS EN FÁBRICAS INTELIGENTES (SMART FACTORIES) BAJO LA NORMA ISA-95/88 E IEC-6199". Aprobado por Consejo Universitario con resolución: 1589-CU-P-2016.

Los beneficiarios de los resultados obtenidos en el presente proyecto son las empresas públicas y privadas con la necesidad de implementar sistemas industriales distribuidos, además de la comunidad de investigadores que se encuentran verificando la factibilidad de la implementación de la norma IEC-61499 en la industria.

1.5. Objetivos

1.5.1. Objetivo General

- Desarrollar un Sistema de Control Industrial basado en el Estándar IEC-61499.

1.5.2. Objetivos Específicos

- Diseñar Bloques de Funciones para manipular las entradas analógicas y salidas de modulación por ancho de pulsos (PWM) de la tarjeta Beaglebone Black bajo el entorno de desarrollo 4DIAC-IDE y el runtime 4DIAC-FORTE.
- Establecer algoritmos de control mediante Bloques de Funciones para la tarjeta de Beaglebone Black mediante el entorno de desarrollo 4DIAC-IDE y el runtime 4DIAC-FORTE.
- Implementar una aplicación de control industrial que utilice la tarjeta de desarrollo Beaglebone Black desde el entorno de desarrollo 4DIAC-IDE y el runtime 4DIAC-FORTE.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Antecedentes Investigativos

En la última década la aplicabilidad de la norma IEC-61499 en sistemas de control distribuido ha sido ampliamente estudiada en muchos proyectos como: redes inteligentes[8], mecanizado[12], control de fabricación[7], mecatrónica[13], control de procesos[14] y en una gran cantidad casos de estudios. Estos estudios de caso han confirmado muchas ventajas de IEC-61499 sobre la tecnología PLC basada en la norma IEC 61131-3 en términos de eficiencia de diseño, interoperabilidad y reutilización de código[2].

En el estudio realizado en[6] se ha comprobado que la implementación de la norma IEC-61499 sobre plataformas de bajo coste a nivel industrial puede considerarse una alternativa en mediciones rápidas de procesos reales donde el aplicar los métodos tradicionales es muy costoso, de la misma forma en[7] se investigaron las habilidades del entorno de automatización de código abierto 4DIAC, demostrando la aptitud así como los beneficios prometidos no sólo para la aplicación del ejemplo seleccionada sino también para aplicaciones de control industriales más grandes. Sin embargo, manifiesta que para obtener todo el potencial de IEC-61499 se requieren mejoras adicionales en 4DIAC para aumentar la flexibilidad y reducir el esfuerzo de desarrollo.

En el estudio de caso realizado en[8] se discute la aplicación de la IEC-61499 para diseñar e implementar aplicaciones Smart Grid basadas en eventos. Demostrando que la norma

IEC-61499 es muy adecuada en la implementación de Redes Inteligentes. Igualmente en el estudio realizado en [14] se demuestra que la arquitectura de referencia y los modelos para el desarrollo de sistemas de medición y control de procesos distribuidos propuestos en la norma IEC-61499 ayudan a cumplir los requisitos básicos para lograr la reconfigurabilidad del sistema.

La normativa IEC-61499 incluye numerosas características para futuros sistemas industriales, ofreciendo beneficios significativos con respecto a la reutilización y distribución de códigos, los cuales han sido estudiados en[15], conjuntamente en[16] manifiesta que el estándar IEC-61499 aporta a los desarrolladores de software de control una arquitectura de referencia para poder diseñar más fácilmente las aplicaciones de control distribuido, gracias a esta arquitectura es posible llegar a implantar controles holónicos o multi-agente que hasta ahora permanecían en un plano teórico.

Los primeros estudios realizados sobre la implementación de la norma han señalado algunas debilidades semánticas. Reportando algunas ambigüedades que están relacionadas con el tiempo de vida de los eventos en la ejecución del ECC y las diferentes posibilidades de programación en las estructuras compuestas de los FB[1], además en el estudio realizado en[5] se manifiesta que esta norma no fue aceptada por la industria a pesar de que es altamente promovida por la comunidad académica.

2.2. Fundamentación teórica

2.2.1. Estándar IEC-61499

- **Definición del estándar**

El estándar IEC-61499 es una arquitectura de referencia diseñada para facilitar el desarrollo de aplicaciones de control con lógica descentralizada. Para ello, propone una arquitectura basada en FB que se definen como la unidad estructural básica de los modelos. Cada bloque se caracteriza por sus entradas, salidas y funciones internas, igual que ocurre con los diagramas de bloques clásicos, a partir de las entradas y la definición del bloque (comportamiento interno) tendremos unas salidas. A través de la interconexión de bloques simples podemos conseguir modelar aplicaciones y modelos complejos. El

estándar busca aprovechar los amplios conocimientos de los ingenieros en el desarrollo de los diagramas de bloques mediante esta forma de modelado[12].

Otra de las nociones básicas de este estándar junto con la definición de FB, es la forma de ejecución de los FB. Los FB no se ejecutan hasta que reciben una señal de entrada, así estos permanecen en un estado de reposo hasta que es necesaria su ejecución, a diferencia de los componentes de la IEC-61131, donde las subrutinas son verificadas periódicamente. Esta forma de ejecución tiene por objetivo la portabilidad y reutilización de los FB. Gracias a que cada FB tiene su propio contexto y es ejecutado sólo ante una demanda, podemos considerar que este se comporta como una unidad totalmente independiente del resto[12].

- **Objetivos del estándar**

El estándar busca conseguir un alto grado de control distribuido siendo este su principal objetivo. A diferencia de los otros estándares de control más antiguos y no adecuados a los nuevos componentes (sistemas embebidos, PC industriales y redes más avanzadas) pueden ser utilizados para descentralizar la lógica de control y poder implantar controles holónicos o multiagente para obtener sistemas más flexibles. Además, la creciente importancia del software en el control hace que la reutilización de código sea vista como una cuestión fundamental, es decir poder diseñar un mismo software para dos máquinas que tienen que hacer lo mismo pero son de fabricantes diferentes[12].

Por estas razones, la arquitectura del IEC-61499 está fundamentada en la busca de tres aspectos claves[1].

- **Portabilidad:** Soportar e interpretar correctamente configuraciones y componentes software, creadas por otras herramientas software.
- **Interoperabilidad:** Los distintos dispositivos integrados pueden funcionar conjuntamente para llevar a cabo las funciones propias de las aplicaciones distribuidas.
- **Configurabilidad:** Cualquier dispositivo y sus componentes software pueden ser configurados por otras herramientas de software de múltiples proveedores.

En la Figura 1 se describen los tres aspectos claves de la arquitectura de la IEC-1499[1].

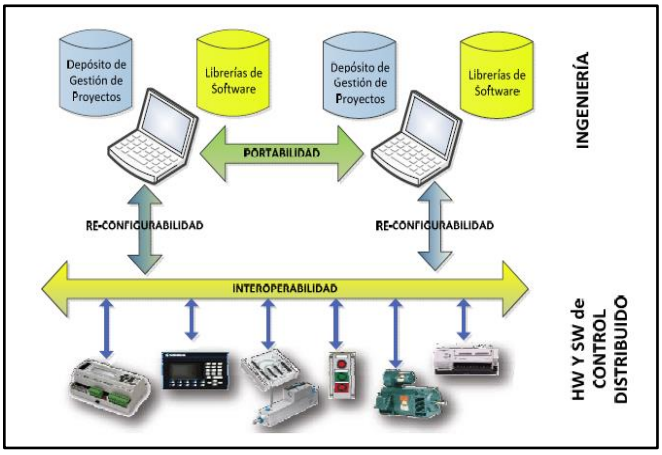


Fig. 1**.** Portabilidad, Re-configurabilidad e Interoperabilidad de la IEC-61499[1]

- **Estructura del Estándar**

En cuanto a la estructuración del estándar se encuentra distribuido en 4 partes, siendo cada una independiente de las otras y con objetivos diferentes[12].

- **Parte 1:** Define la arquitectura de referencia para el desarrollo, reutilización e implementación de los bloques funcionales. Se definen entre otros, los conceptos de FB básico, FB compuesto, aplicación, interfaz, dato, señal, evento, etc.
- **Parte 2:** En esta parte se definen los requisitos que deben cumplir las herramientas de desarrollo para poder implementar las arquitecturas propuestas en la parte 1. No obstante, los requisitos definidos no permiten por ellos mismo garantizar la interoperabilidad, portabilidad y la configurabilidad.
- **Parte 3:** Información didáctica que tiene por objeto mostrar las funcionalidades y ejemplos de aplicación de la IEC-61499 en un amplio conjunto de dominios.
- **Parte 4:** El objetivo de esta parte es definir unos perfiles de compilación de forma que se puedan cumplir los objetivos de la norma IEC-61499 en los dispositivos compatibles y las herramientas de desarrollo.

2.2.2. Modelos de la Norma IEC-61499

En esta sección desarrollaremos algunos de los concepto clave para entender la arquitectura del estándar IEC-61499. Estos conceptos se encuentran definidos en la parte 1 del estándar y constituyen el esqueleto de la misma.

- **Modelo de Bloque Funcional (FB)**

Es el elemento más pequeño en un sistema de control distribuido el cual se encuentra representado en la figura 2. El FB consiste en una cabeza que está conectada al flujo de eventos, el cual acepta entrada de eventos y genera salida de eventos, el cuerpo está conectado al flujo de datos, acepta datos de entrada y genera datos de salida. El comportamiento dinámico del FB está definido por la Gráfica de Control de Ejecución (ECC) que procesa entrada de eventos y genera salida de eventos[1], es decir el ECC describe el comportamiento interno de las instancias de los FBs básicos. Un FB en el IEC-61499-1 se mantiene pasivo hasta que es disparado por una entrada de evento, es decir estos eventos son usados para activar un bloque funcional.

La funcionalidad del FB esta proporcionada por medio de algoritmos. Un algoritmo puede ser escrito en cualquiera de los 5 lenguajes que menciona el IEC 61131-3: IL, ST, LD, FBD y SFC. También en otros lenguajes de alto nivel como: C, C++, Java y Delphi. El algoritmo procesa entradas y datos internos, generando datos de salida[1]. Se define tres diferentes tipos de bloques funcionales:

- **FB Básico:** Unidad más pequeña de programación. Consta de dos partes: ECC y algoritmos.
- **FB Compuesto:** Compuesto por una red de instancias de FBs interconectados.
- **FB Interfaz de Servicio:** Proporciona servicios a una aplicación, como interacción entre aplicación y recursos.

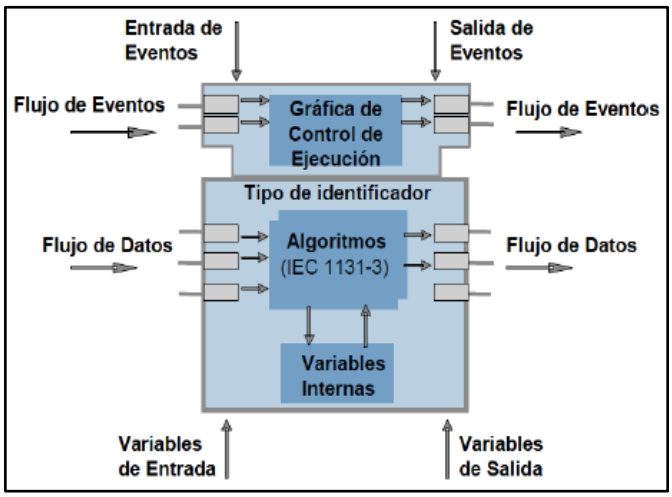


Fig. 2. Modelo de Bloque Funcional[1]

- **Bloque Funcional de Interfaz de Servicio (SIFB)**

El SIFB un bloque funcional que proporciona servicios a la aplicación, permitiendo la interacción entre aplicación y recursos. Proporciona una interfaz entre el software de control y el hardware con los sistemas de comunicaciones. Una de las funciones principales del SIFB es proporcionar servicios y el estándar IEC-61499-1 define los siguientes conceptos:

- **Servicio:** es la capacidad funcional de un recurso, la cual puede ser modelada por una secuencia de primitivas.
- **Primitiva:** es una representación abstracta de una interacción entre una aplicación y un recurso.

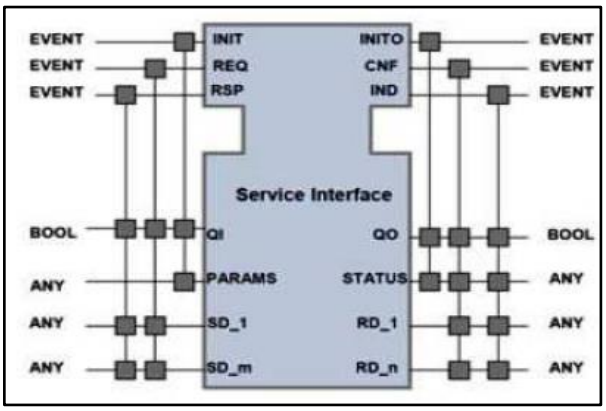


Fig. 3. Estructura general de un SIFB[1]

Para declarar los tipos de SIFBs, el estándar define el conjunto de entradas de eventos, salidas de eventos, entradas de datos y salidas de datos listados en la Tabla 1 y 2 respectivamente[1].

- **Especificaciones del SIFB**

Las interfaces externas de un SIFB tienen la misma apariencia que un FB. Sin embargo, algunas entradas y salidas de un SIFB tienen semántica especializada y el comportamiento de instancias de estos tipos de SIFBs está definido a través de una notación gráfica especializada para secuencia de primitivas de servicio.

En función del tipo de SIFB, se ofrecen diferentes servicios. El estándar IEC 61499-1 define dos tipos de SIFBs que son representados en la Figura 4 [1].

Tabla 1. Eventos y Datos de entrada de un SIFB bajo norma IEC-61499[1].

ENTRADA DE EVENTO	ENTRADA DE DATOS
INIT Esta entrada de evento debe ser asignada a una primitiva de petición la cuál solicita una inicialización del servicio proporcionado por la instancia del FB.	QI: BOOL Representa un calificador en la primitiva de servicio asignada a la entrada del evento. Si esta entrada es VERDADERA sobre la ocurrencia de un evento INIT, el servicio de inicialización es solicitado; si es FALSA, el servicio de finalización es requerido.
REQ Esta entrada de evento debe ser asignada a una primitiva de petición del servicio proporcionado por la instancia del FB.	PARAMS: ANY La entrada contiene uno o más parámetros asociados con el servicio, típicamente como elementos de una instancia de un tipo de dato estructurado. Cuando esta entrada está presente, la especificación del tipo de FB debe definir el tipo de dato y valor inicial por defecto.

Tabla 2. Eventos y Datos de salida de un SIFB bajo norma IEC-61499[1]

SALIDA DE EVENTO	SALIDA DE DATOS
INITO Esta salida de evento debe ser asignada a una primitiva de confirmación que indica la finalización del procedimiento de un servicio de inicialización.	QO: BOOL Esta variable representa un calificador en la primitiva de servicio asignada a la salida de eventos. Por ejemplo, si el valor de esta salida es VERDADERO sobre la ocurrencia de un evento INITO indica la inicialización exitosa del servicio; si esta salida tiene un valor FALSO indica la inicialización NO exitosa.
CNF Esta salida de evento debe ser asignada a una primitiva de confirmación del servicio proporcionado por la instancia del FB.	STATUS: ANY Esta salida debe ser un tipo de dato apropiado para expresar el estado del servicio sobre la ocurrencia de una salida de evento.

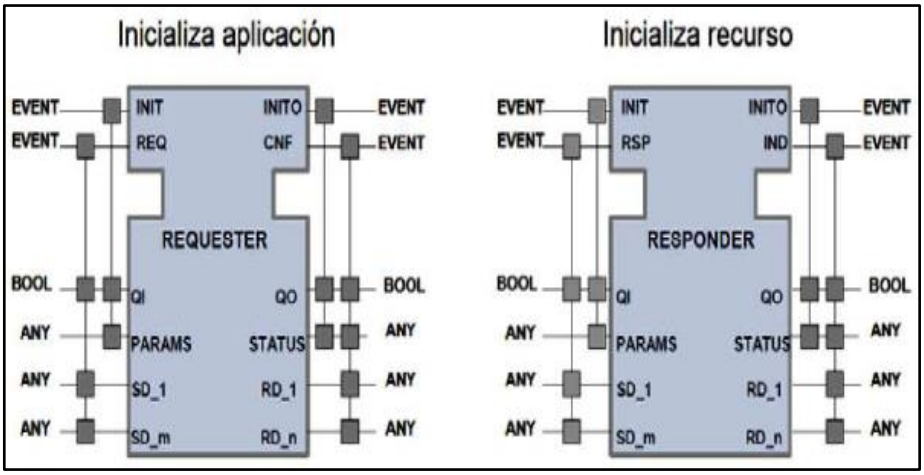


Fig. 4. SIFB Requester y Responder[1]

1. **Petición (REQUESTER).** Representa una interacción para inicializar una aplicación (ejemplo: envío de un mensaje). Este permanece pasivo hasta la llegada de una entrada de evento. Por otro lado, su ejecución es similar a la ejecución de un FB básico.
2. **Respuesta (RESPONDER).** Representa una interacción de inicializar un recurso. Si un servicio en reposo lanza su ejecución, este tipo de SIFB proporciona una salida de evento. También puede interrumpir la ejecución de cualquier otro FB.

En el estándar IEC 61499-1 la comunicación se implementa vía SIFBs de comunicaciones. Los dos paradigmas de comunicación empleados son publicador/suscriptor (PUBLISH/SUBSCRIBE) y cliente/servidor (CLIENT/SERVER).

La comunicación publicador/suscriptor transmite y recibe información en una única dirección, (ver la Figura 5) es decir, la transferencia de datos es unidireccional y se lleva a cabo por el uso del protocolo de datagrama de usuario (por ejemplo para UDP, (User Datagram Protocol)

El método publicador/suscriptor envía los datos en el bus los cuales se “publican” una vez, y todos los demás dispositivos que necesitan los datos escuchan o se “suscriben” a la misma transmisión. Por lo tanto, un parámetro específico puede ser usado por tantos dispositivos o funciones diferentes como el usuario requiera, sin incrementar el tráfico en el bus o afectar gravemente al rendimiento de control.

Este tipo de comunicaciones también son determinísticas. Esto significa que siempre ocurren sobre un programa predeterminado, así que es seguro que la información se transmitirá y recibirá precisamente cuando se necesita[1].

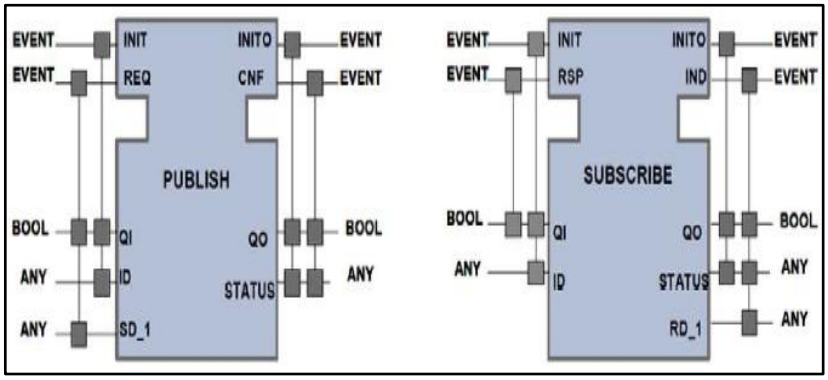


Fig. 5. Bloques de interfaz de comunicaciones Publish y Subscribe[1]

La comunicación cliente/servidor transmite y recibe información en dos direcciones, (ver Figura 6), es decir la transferencia de datos es bidireccional y se lleva a cabo por el uso del protocolo de control de transmisión (por ejemplo para TCP, Transmission Control Protocol)[1].

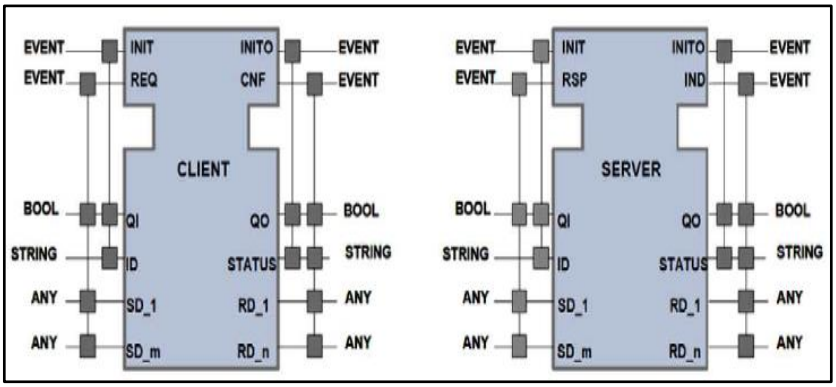


Fig. 6. Bloques de interfaz de comunicaciones Client y Server[1]

- **Modelo de Recurso**

La finalidad del recurso es actuar de interfaz entre los sistemas de comunicación y los procesos específicos del dispositivo, como por ejemplo los subsistemas de entradas/salidas del dispositivo. Con ellos se logra el correcto funcionamiento del conjunto del sistema de control basado en aplicaciones cuyo comportamiento ha sido modelado independientemente de donde vaya a ser ejecutado, además los recursos son

unidades que se ejecutan independientemente entre ellos aportando flexibilidad al sistema de control.[17]. En la Figura 7 se representa el Modelo de Recurso según la norma IEC 61499.

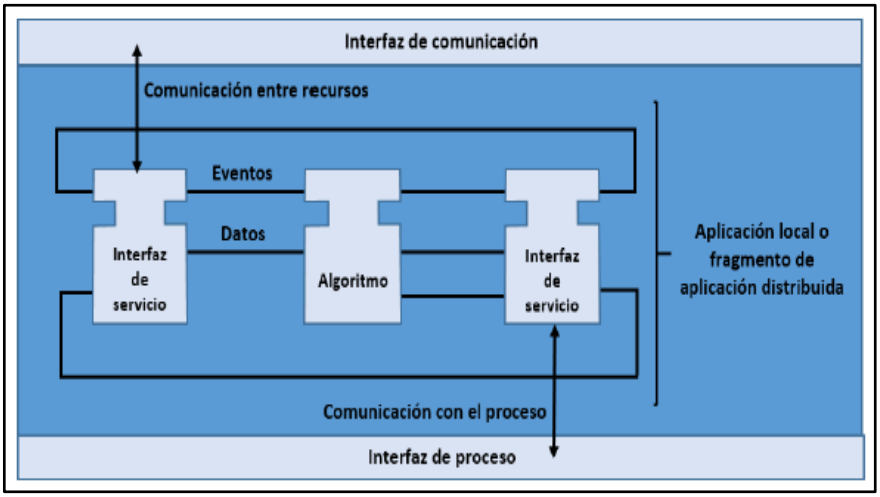


Fig. 7. Modelo de recurso en la norma IEC 61499[17]

- **Modelo de Dispositivo**

El modelo de dispositivo representa la entidad física que va a ejecutar nuestra red de bloques funcionales. Lo podríamos asimilar a un sistema embebido o a un PC. El dispositivo es capaz de comunicarse con otros dispositivos o con sus periféricos a través de interfaces de comunicación. Al mismo tiempo es igualmente capaz de comunicar con la red de bloques funcionales a través de una interfaz de proceso[12]. En la Figura 8 se representa el Modelo de Dispositivo del estándar IEC-61499.

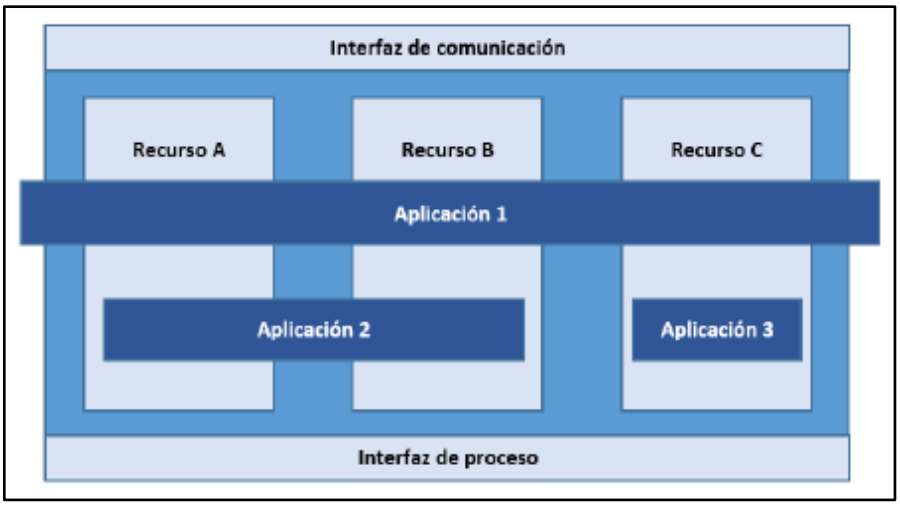


Fig. 8. Modelo de dispositivo del estándar EC-61499[17]

- **Modelo de Sistema**

El sistema es el elemento de mayor nivel que contempla esta arquitectura, engloba todos los dispositivos capaces de comunicarse y el conjunto de aplicaciones, así como las relaciones entre estos dos grupos. Desde el punto de vista del sistema, las aplicaciones son unitarias pero estas se pueden encontrar distribuidas entre los diferentes dispositivos del sistema[12]. En la Figura 9 se representa el Modelo de Sistema en la norma IEC-61499.

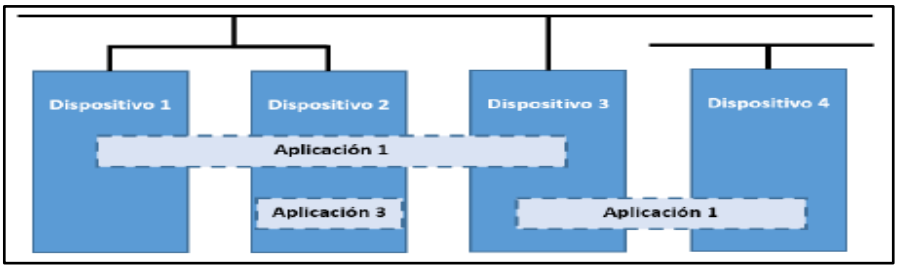


Fig. 9. Modelo de sistema en la norma IEC 61499[17]

- **Modelo de aplicación y sub-aplicación**

Una aplicación es una red de bloques de función conectados por flujos de eventos y datos que modela un comportamiento de un automatismo sin atender a donde se va a ejecutar. Posteriormente se distribuye esta aplicación entre los recursos disponibles para su correcta ejecución. Además se pueden definir sub-aplicaciones, que son grupos de bloques de función interrelacionados que se comportan como un bloque de función ya que tienen entradas y salidas de datos y eventos [17]. En la Figura 10 se representa el Modelo de Aplicación en la norma IEC 61499.

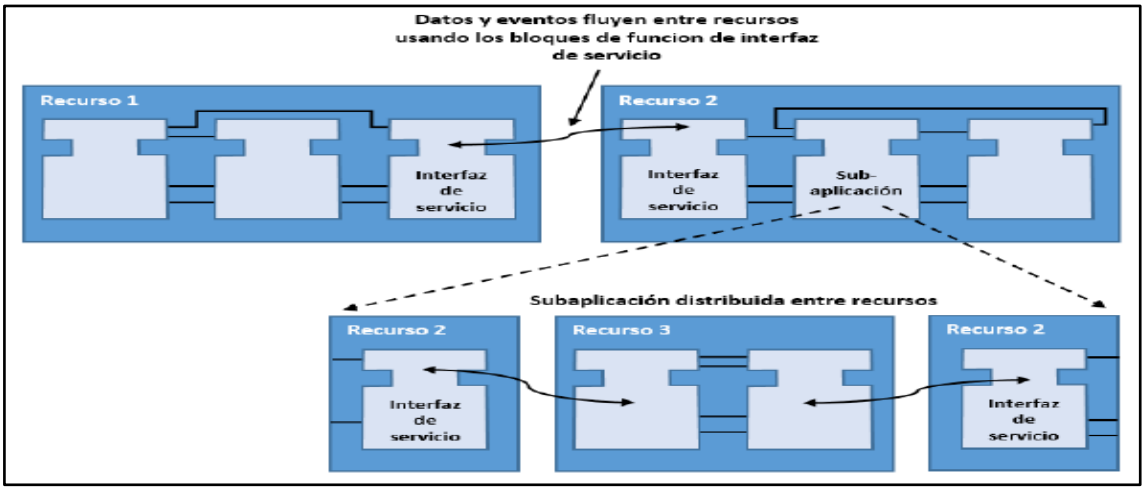


Fig. 10. Modelo de aplicación en la norma IEC 61499[17]

2.2.3. Herramientas de Desarrollo de los Modelos

En la parte 2 de la norma se especifican las características que deben cumplir las herramientas de desarrollo para lograr la interoperabilidad, uno de los principales objetivos de esta norma. A continuación presentamos brevemente las herramientas de desarrollo más importantes.

- **Functional Block Development Kit (FBDK)**

Este es el primer entorno de desarrollo que se creó, por uno de los miembro creadores del IEC-61499. Su principal objetivo era permitir la visualización de los FB, aunque con los años ha ido evolucionando y ahora permite además de representar, testear los bloques e intercambiarlos en forma de XML como se define en la parte 2 de la norma.

En cuanto a la herramienta, se encuentra desarrollada en java, posee un interfaz simple y poco amigable para el usuario. Los distribuidores sacan actualizaciones continuas para reflejar los constantes cambios y mejoras en la norma. En la Figura 11 se representa el entorno de desarrollo de FBDK.

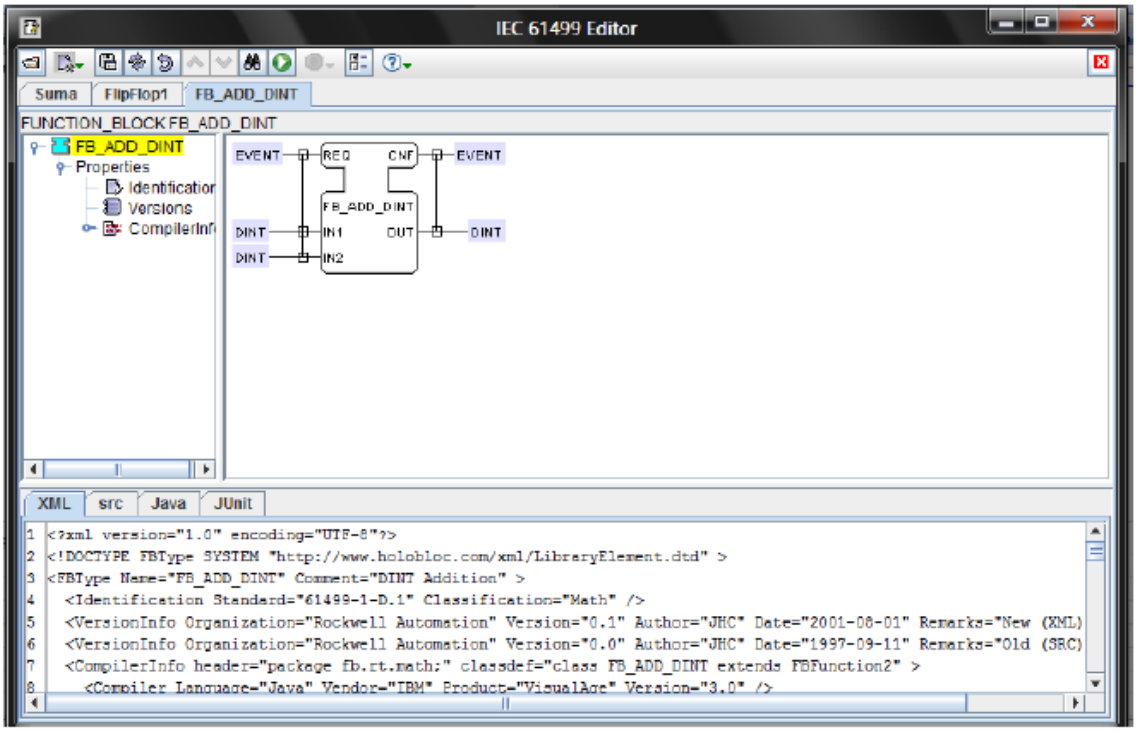


Fig. 11. Entorno de desarrollo de FBDK[1]

- **4Diac-IDE**

4diac es un entorno de desarrollo opersource extensible a través de plugins, basado el reconocido entorno de desarrollo Eclipse. La aplicación es muy robusta e intuitiva, tiene soporte oficial y saca dos actualizaciones del entorno por año. Además, permite distribuir muy fácilmente las aplicaciones entre diferentes recursos de forma gráfica. En las últimas versiones ha añadidos las funcionalidad de test y debug por medio de un display online que permite fijar y leer los valores de las variables de forma remota. Se ha demostrado su portabilidad con FBDK[12]. En la Figura 12 se representa el entorno de desarrollo de 4Diac.

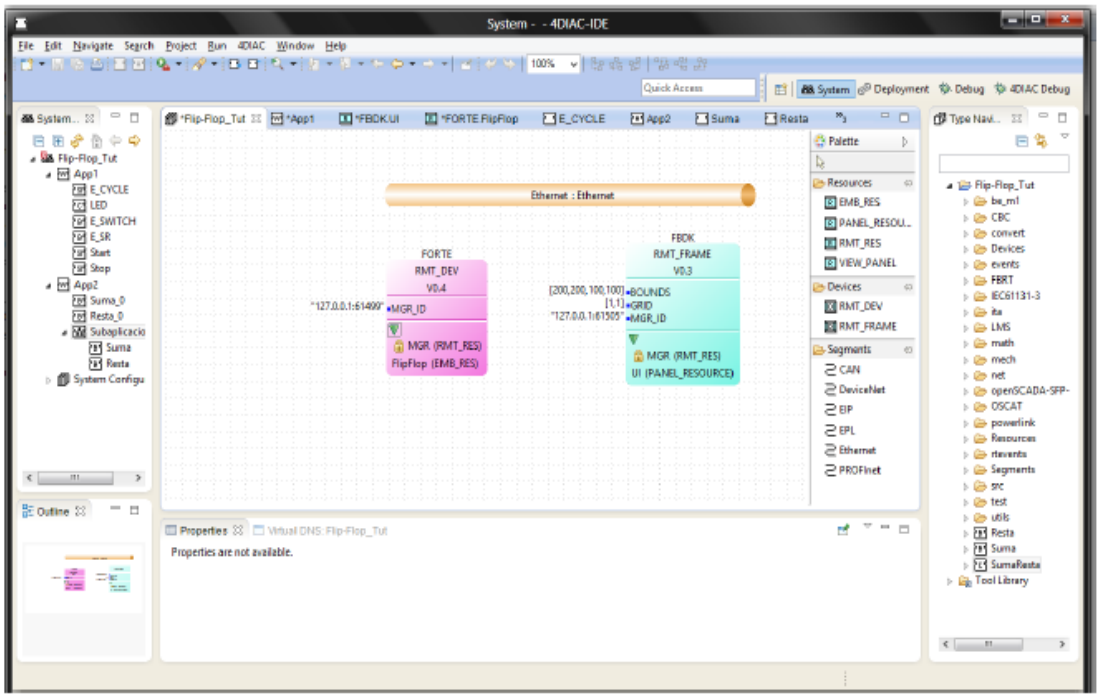


Fig. 12. Entorno de desarrollo de 4DIAC-IDE[1]

2.2.4. Plataformas de Ejecución

La plataforma de ejecución se encargara de gestionar el Hardware del dispositivo sobre el que se ejecuta nuestro modelo, entre otras cosas deberá gestionar la memoria, las comunicaciones y las entradas/salidas del dispositivo. A estos efectos podemos asimilar la plataforma de ejecución a un sistema operativo que se encarga de ejecutar nuestro programa[12].

- **Funtional Block Runtime (FBRT)**

Al igual que la herramienta de software FBDK, esta plataforma de ejecución fue desarrollada en el lenguaje Java y se utiliza para las pruebas de viabilidad y la demostración de la arquitectura IEC-61499[1].

- **Forte**

Es una pequeña plataforma de ejecución implementada en C++ y de código abierto, desarrollada por el mismo equipo que 4DIAC. Esta especialmente desarrollada para ejecutarse en pequeños sistemas embebidos y sistemas de tiempo real, ya que tiene los mecanismos necesarios para poder asegurar la ejecución en tiempo real y la gestión de prioridades. Otro de los puntos fuertes de FORTE es que ha sido diseñada para ser independiente de la plataforma sobre la que se ejecute, para poder ser ejecutada en diferentes hardwares y sistemas operativos. En la actualidad FORTE se ha llevado a diferentes sistemas operativos como POSIX, windows32, threadX y eCos. FORTE se puede configurar con 4DIAC pero también es compatible con otros entornos como FBDK[12].

2.3. Propuesta de solución

Con la investigación, se propone implementar un Sistema de Control Industrial basado en el estándar IEC-61499 para la variable de nivel de líquido de la maqueta analógica FESTO® MPS-PA Compact Workstation del laboratorio de hidráulica y neumática de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato.

CAPITULO 3

METODOLOGÍA

3.1. Modalidad de investigación

Proyecto de investigación aplicada, debido a que se tomó información, conocimientos previos y se los aplica para resolver una problemática específica.
En la elaboración del presente proyecto de investigación se utilizará las siguientes modalidades de investigación:

3.1.1. Investigación de Campo

Debido a que la implementación del Sistema de Control se la realizó en los laboratorios de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato, utilizando la respectiva maqueta FESTO.

3.1.2. Investigación Bibliografía

Porque se buscó información en libros, revistas, artículos científicos, tesis doctorales y en la web, que ayudaron al cumplimiento de los objetivos planteados.

3.1.3. Investigación experimental

Debido a que se realizó varias pruebas hasta obtener el correcto funcionamiento del sistema de Control Industrial implementado.

3.2. Población y muestra

Por las características de la presente investigación. No se requiere población y muestra.

3.3. Recolección de información

Para el presente proyecto de investigación se recopiló información de artículos científicos, tesis, revistas, manuales, libros, páginas de internet, etc.

3.4. Procesamiento y análisis de datos

Una vez obtenida la información se procedió a realizar los siguientes pasos:

a) Análisis, revisión e interpretación de la información recopilada.
b) Selección de alternativas para dar solución al problema planteado.
c) Pruebas de funcionamiento.
d) Control de errores.
e) Interpretar y analizar los resultados.

3.5. Desarrollo del proyecto

El desarrollo del proyecto se basa en las siguientes actividades:

1. Revisar la normativa IEC-61499.
2. Seleccionar una variable para nuestra aplicación en la maqueta FESTO® MPS-PA Compact Workstation.
3. Adquirir entorno de desarrollo 4DIAC-IDE y el runtime 4DIAC-FORTE.
4. Obtener la tarjeta de desarrollo BBB y examinar el datasheet de la misma.
5. Diseñar los circuitos necesarios para la protección de la BBB y acondicionamiento necesarios para la aplicación.
6. Diseñar bloques de funciones para controlar las entradas analógicas y las salidas tipo PWM de la BBB.
7. Diseñar un bloque de función para implementar el algoritmo de control.
8. Implementar el runtime en la tarjeta BBB.
9. Exportar los archivos .cpp y .h correspondientes a los FBs diseñados en entorno de desarrollo 4DIAC-IDE al runtime FORTE de la BBB.

10. Editar los archivos .cpp y .h correspondientes a los FBs diseñados anteriormente con los algoritmos y librerías necesarias.
11. Realizar un diseño en conjunto con los FBs creados y los existentes para crear el sistema de control.
12. Implementar una aplicación de control de la variable seccionada.
13. Pruebas de funcionamiento y corrección de errores en la aplicación.
14. Realizar un informe final con los resultados obtenidos.

CAPÍTULO 4

DESARROLLO DE LA PROPUESTA

Hoy en día la necesidad de automatizar los procesos industriales con el fin de obtener una mayor eficiencia en los productos y servicios en cualesquier área desempeñada por el hombre ha sido motivo de llevar a cabo el presente proyecto de investigación mediante la implementación de un Sistema de Control basado en el Estándar IEC-61499. El cual busca conseguir un alto grado de control distribuido, pudiéndose utilizar sistemas embebidos existentes para descentralizar la lógica de control flexibilizando así los sistemas, y a la vez normalizar el desarrollo de código de manera que este sea independiente de la máquina sobre la que se ejecuta.

4.1. Selección de maqueta y proceso para la implementación del sistema de control.

Para la implementación del sistema de controlo industrial basado en el estándar IEC-61499 se seleccionó la maqueta analógica FESTO® MPS-PA Compact Workstation (Ver Figura 13) del laboratorio de hidráulica y neumática de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato, debido a los diferentes procesos existentes en la misma, ideales para implementar sistemas de control continuos en el tiempo.

La maqueta analógica FESTO® MPS-PA Compact Workstation dispone de procesos didácticos para realizar controles de nivel, caudal, presión y temperatura, los cuales se puede operar individualmente. El diseño de los sensores y actuadores permite el uso de controladores continuos (por ejemplo, P, I, PI, PID) y discontinuos (por ejemplo, controladores de dos puntos).

4.1.1. Diseño básico MPS PA® Compact Workstation

- **Componentes mecánicos:** 2 depósitos de líquido, depósito de presión, sistema de tubos, marco de montaje, placa de perfil.
- **Sensores:** 2 sensores capacitivos, 2 interruptores de flotador, sensor de ultrasonidos, sensor de flujo, sensor de presión, sensor de temperatura PT100.
- **Actuadores:** bomba, válvula de control proporcional direccional, válvula de bola con actuador de proceso neumático, calefacción.
- **Componentes eléctricos:** Placa de conexión de E/S con transductor de medición, controlador de motor, terminal de E/S, SysLink, 8I/8O, terminal analógico,15 pines
- **Media:** Documentación técnica con libro de trabajo.

Fig. 13. Célula analógica FESTO MPS PA® Compact Workstation

El objetivo de este trabajo investigativo es la implementación de un control proporcional, integral y derivativo (PID) en el el proceso de nivel de líquido de la célula analógica FESTO® MPS-PA Compact Workstation.

4.1.2. Diagrama de Bloques del control de nivel

El sistema de control de nivel se representa gráficamente en la Figura 14 con un diagrama de bloques, los cuales indican la interrelación existente entre los distintos componentes del sistema de control a implementarse.

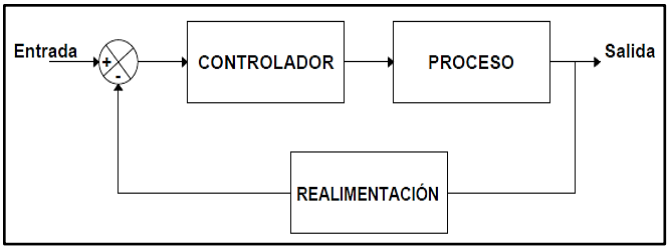


Fig. 14. Diagrama de bloques lazo de control

4.1.3. Diagrama PI&D de nivel

En el diagrama PI&D de la Figura 15 se representa los componentes relacionados al proceso de control de nivel implementado, donde la bomba P101 suministra un fluido desde un depósito de almacenamiento B101 a un depósito de depósito B102 a través de un sistema de tuberías. El nivel del fluido dentro del tanque B102 se monitoriza con un sensor analógico ultrasónico B101 en el punto de medición LIC B101 y se lee como valor real. El valor debe mantenerse en un cierto nivel incluso cuando se producen perturbaciones o cambios en el punto de consigna o setpoint. Las válvulas manuales V101 y V110 se encuentran abiertas.

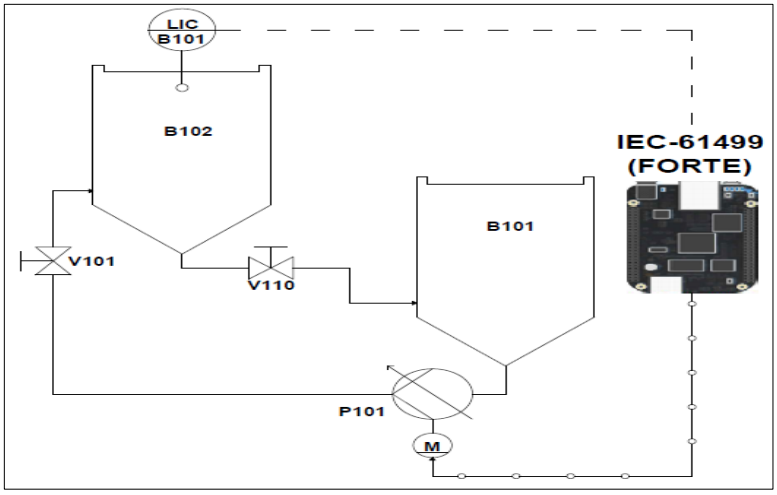


Fig. 15. Diagrama PI&D de nivel

4.2. Selección del Software

4.2.1. Selección del Entorno de Desarrollo

Se ha optado por 4DIAC (4DIAC-IDE) versión 1.8.2, el cual es un entorno de desarrollo Open Source, además de ser una aplicación robusta, intuitiva y tener soporte oficial. Posee la funcionalidad de test y debug por medio de un display online que permite fijar y leer los valores de las variables de forma remota. Las aplicaciones modeladas se pueden descargar a dispositivos de campo distribuidos según los medios definidos por la norma IEC-61499, además es compatible con la mayoría de las herramientas actualmente existentes y ha demostrado su portabilidad con nxtStudio y FBDK.

4.2.2. Selección de la plataforma de ejecución (runtime)

Como plataforma de ejecución para la implementación del sistema de control se ha seleccionado 4DIAC (4DIAC-RTE, FORTE) debido a la versatilidad que ofrece para elegir el hardware, ya que es independiente de la plataforma en la que se ejecute y funciona sobre varios sistemas operativos como Windows o Linux, demás es compatible con una gran cantidad de tarjetas embebidas y Controladores Lógicos Programables (PLC). FORTE soporta la reconfiguración en línea de sus aplicaciones y la ejecución en tiempo real de todos los tipos de FB proporcionados por la norma IEC-61499, también es compatible con FBDK y nxtOne, además es de código abierto.

4.3. Selección de hardware

Una vez seleccionado el software que se empleará para la implementación del sistema de control, se procede a la selección del sistema embebido o al Controlador Lógico Programable compatible con el software antes seleccionado.

Los sistemas embebidos y controladores lógicos programables compatibles con el runtime FORTE son expuestos en la Tabla 3.

Tabla 3. Sistemas embebidos recomendados por el desarrollador de 4DIAC_IDE y FORTE.

Tarjetas embebidas	Controladores Lógicos Programables (PLC)
• BeagleBone Black • Digi Connect ME (ARM7) • KIPR's CBC v2 robot controller • Lego Mindstorms EV3 (ARM7) • Raspberry PI	• Bachmann electronic M1 PLC • CONMELEON C1 • Raspberry-SPS • WAGO PFC200

De la Tabla 3 se eliminaron las opciones de los Controladores Lógicos Programables (PLC) para la implementación de nuestro sistema de control ya que se necesita un dispositivo que no soporte ambientes industriales.

En la Tabla 4 se indica los precios de las tarjetas embebidas.

Tabla 4. Precios de las tarjetas embebidas.

Tarjetas embebidas	Precio($)
BeagleBone Black	56.65
Digi Connect ME (ARM7)	149.35
KIPR's CBC v2 robot controller	225.00
Lego Mindstorms EV3 (ARM7)	278.35
Raspberry PI 2 Model B	66,88

La tarjeta embebida seleccionada fue la BBB de Texas Instruments debido a que es uno de los dispositivos más económicos y además consta con pines de entrada y salida analógicas y digitales, los cuales son necesarios para el sistema de control desarrollado en la presente investigación. En la Figura 16 se representa la tarjeta BBB.

Fig. 16. Tarjeta BeagleBone Black

4.3.1. Características de la tarjeta BBB

La BBB es un sistema embebido basado en el procesador XAM3359AZCZ100 Cortex A8 ARM de Texas Instruments a 1 GHz, 512 MB de RAM y 2 GB de almacenamiento interno, además de una ranura microSD. El sistema operativo de la BBB viene con la distribución de Linux Debian. Muchas otras distribuciones de Linux y sistemas operativos también son compatibles con la BBB, incluyendo: Ubuntu, Android, Fedora, Ångström, ArchLinux, Gentoo, Sabayon, Buildroot, Erlang. Además dispone de una conexión USB, una mini USB y Ethernet, también cuenta con dos conectores de 46 pines cada uno entre los que se dispone de 65 entradas/salidas tipo (GPIO), 7 entradas analógicas, 8 salidas EHRPWM y pines para comunicaciones como I2C, SPI y CAN Bus.

4.4. Desarrollo de capa interfaz de entradas analógicas y salidas EHRPWM

Para la aplicación desarrollada se manipularán las entradas analógicas y las salidas EHRPWM de la BBB, las entradas analógicas admiten un rango de 0 a 1,8V y las señales PWM funcionan a 3,3V respectivamente, por lo que se plantea la necesidad de usar una capa interfaz que permita la conexión de los equipos que conforman el sistema de control.

Además se considerará algunas limitaciones de uso que tiene la tarjeta BBB las cuales son: superar el valor de 1,8V en las entradas analógicas y aplicar tensiones a los pines de entrada analógica mientras el dispositivo esté apagado o en proceso de arranque, como consecuencia de la misma existiría una destrucción del procesador.

Para la implementación del sistema de control se conectarán un sensor analógico con rango de medida de 0-5V y un actuador a manipularse con una señal PWM a 3,3V, además se protegerá la BBB frente a la situación de aplicar tensiones en sus pines mientras este está apagado o en proceso de arranque.

A continuación se describen los circuitos que compone la capa:

4.4.1. Circuito de protección

El circuito representado en la Figura 17 tiene la finalidad de evitar que se aplique una tensión a la entrada analógica mientras la BBB esté apagada o en proceso de arranque. La BBB dispone de una salida digital (SYS_RESETn) para indicar cuando el sistema está en

marcha, esto es después de que se haya realizado el arranque del sistema. Hasta este momento no debe de haber tensiones en ninguno de sus pines, ya que esto dañaría el procesador.

Para solucionar este problema se activará un relé para el paso de corriente en la entrada del sensor analógico desde un circuito que corta el suministro de energía mientras la salida SYS_RESETn esté en estado lógico 0.

El circuito consta de un transistor BJT con el que se polariza un MOSFET de canal P. Es necesario que este último sea de canal P, ya que se desea cortar el suministro de tensión en los componentes.

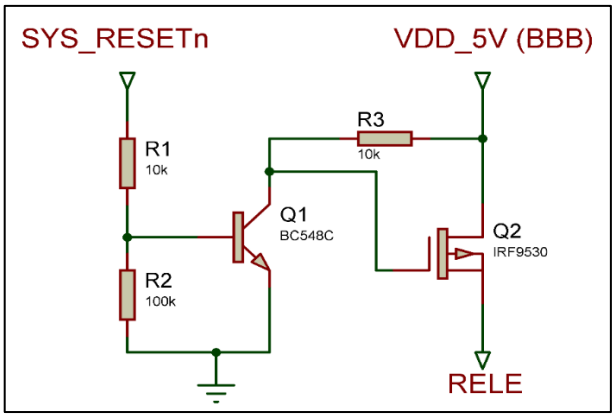


Fig. 17. Circuito de protección.

4.4.2. Circuito de adaptación de rangos de tensión para la entrada analógica

El circuito tiene la finalidad de adaptar el rango de tensión de medida de una entrada analógica de la BBB que va de 0-1,8 Voltios a un rango de 0-5.0 Voltios para lo cual se realizó un divisor de tensión como se indica en la Figura 18.

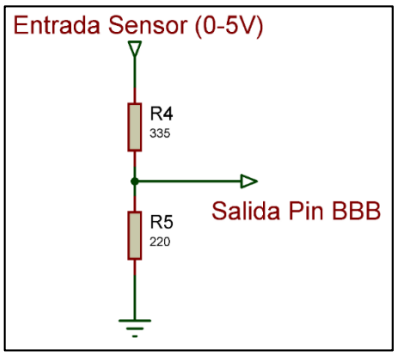


Fig. 18. Circuito de adaptación de rangos de tensión para la entrada analógica.

- Cálculos y consideraciones

Para el cálculo de la resistencias R4 se consideró una R5=220 Ω, un Vsal=1,8 V y Vent=5 V.

$$V_{Sal} = \left(\frac{R_5}{R_4 + R_5}\right) * V_{Ent} \quad (1)$$

$$1{,}8V = \left(\frac{220\Omega}{R_4 + 220\Omega}\right) * 5V$$

$$1{,}8R_4 + 396\Omega = 1100\Omega$$

$$1{,}8R_4 = 604\Omega$$

$$R_4 = 335\Omega$$

La Figura 19 presenta el circuito final de protección y adaptación de rangos de tensión, el relé está alimentado desde el circuito de protección explicado anteriormente. De esta manera el relé está desactivado cuando la salida SYS_RESETn está en estado bajo, protegiendo así el procesador de la BBB, además se implementó un led que indica si la BBB está en marcha.

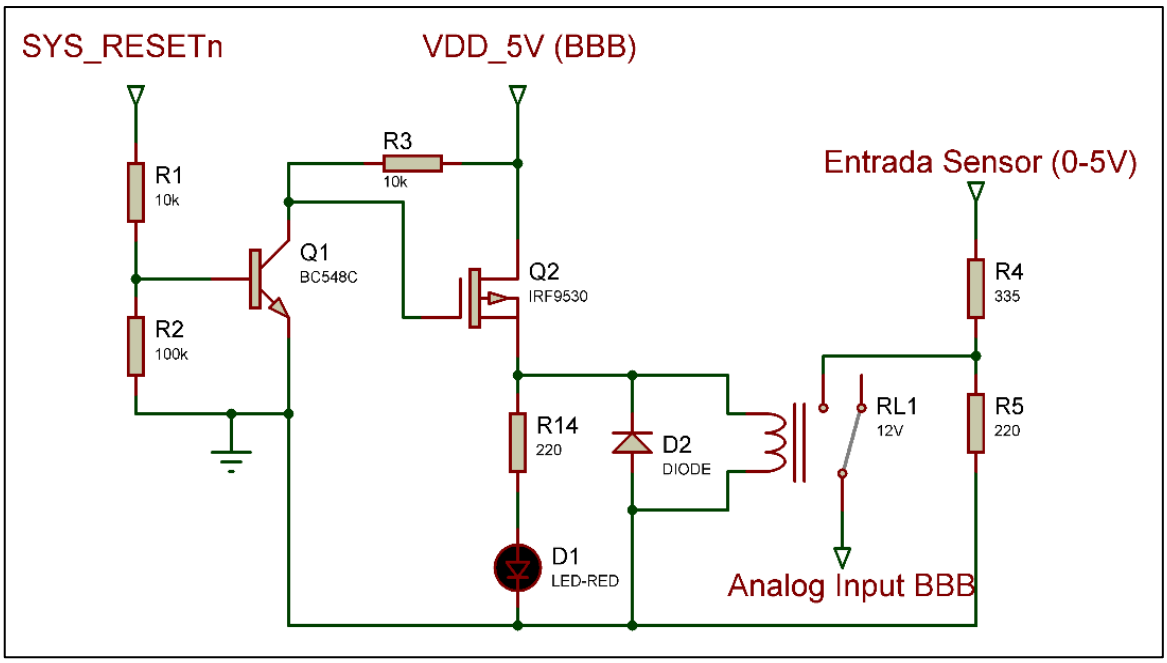


Fig. 19. Circuito de protección y adaptación de rangos de tensión.

4.4.3. Circuito de acondicionamiento para del sensor de nivel

El sensor de nivel analógico ultrasónico empleado proporciona una señal estándar de 4-20 miliamperios (mA) por lo que necesariamente se realizó un circuito de acondicionamiento de señal para una conversión a voltaje de 0-5(V) el cual es un valor estándar común en los sensores industriales.

Para la realización del conversor de corriente (4-20mA) a voltaje (0-5V) se tienen los siguientes circuitos:

- **Circuito seguidor de voltaje**

El seguidor de tensión actuará como amortiguador de aislamiento, aislando el circuito para que la potencia del circuito se altere muy poco. El circuito está representado en la Figura 20.

- **Circuito restador de voltaje**

El objetivo del circuito es obtener un valor de 0V cuando el voltaje en la salida de seguidor de tensión sea el valor mínimo. El circuito está representado en la Figura 20.

- Cálculos y consideraciones

Para el cálculo de la resistencias R9 se consideró una R8=1000 Ω, R7=10000 Ω, Vsalida=1,2V, Ventrada=4.4V y un V(5V)=5V.

$$V_{salida} = V_{entrada} * \frac{R_9}{R_8} - V_{(5V)} * \frac{R_9}{R_7} \quad (2)$$

$$1.20V = 4.40V * \frac{R_9}{1000\Omega} - 5.00V * \frac{R_9}{10000\Omega}$$

$$1.20V = \frac{0.004V * R_9}{\Omega} - \frac{0.0005V * R_9}{\Omega}$$

$$1.20V = \frac{0.0035V * R_9}{\Omega}$$

$$R_9 = 342\Omega$$

- **Circuito amplificador no inversor de voltaje**

El circuito tiene por objetivo obtener una señal de voltaje de 0-5V del sensor de nivel analógico ultrasónico. El circuito está representado en la Figura 20.

➢ Cálculos y consideraciones

Para el cálculo de la resistencias R11 se consideró una R12=1000 Ω, Vsalida=5,0V y un Ventrada=1.2V.

$$V_{salida} = V_{entrada} * \left(1 + \frac{R_{12}}{R_{11}}\right) \quad (3)$$

$$5.00V = 1.20V * \left(1 + \frac{1000\Omega}{R_{11}}\right)$$

$$4.16 = \left(1 + \frac{1000\Omega}{R_{11}}\right)$$

$$4.16 = \frac{R_{11} + 1000\Omega}{R_{11}}$$

$$4.16 * R_{11} = R_{11} + 1000\Omega$$

$$3.16 * R_{11} = 1000\Omega$$

$$R_{11} = 316\Omega$$

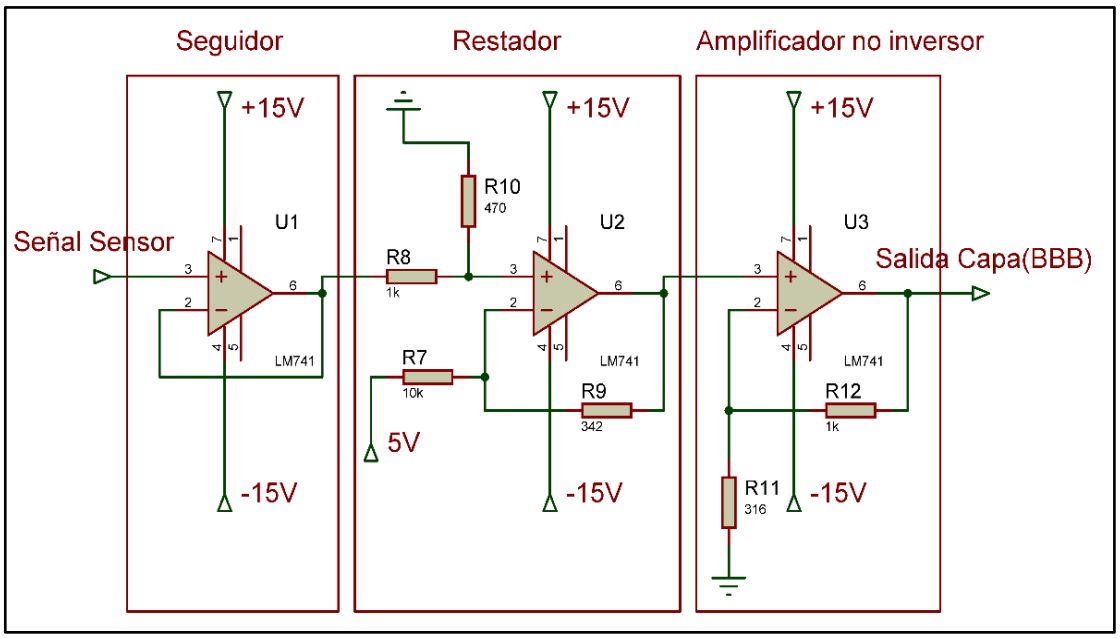


Fig. 20. Circuito conversión de corriente 4-20mA a un voltaje 0-5V

4.4.4. Circuito de acondicionamiento de salida PWM

El objetivo del circuito es manipular la bomba de agua de 24V con la salida EHRPWM de la BBB la cual trabaja con un voltaje de 3.3V insuficiente para la manipulación de la bomba. La Figura 21 representa el circuito diseñado.

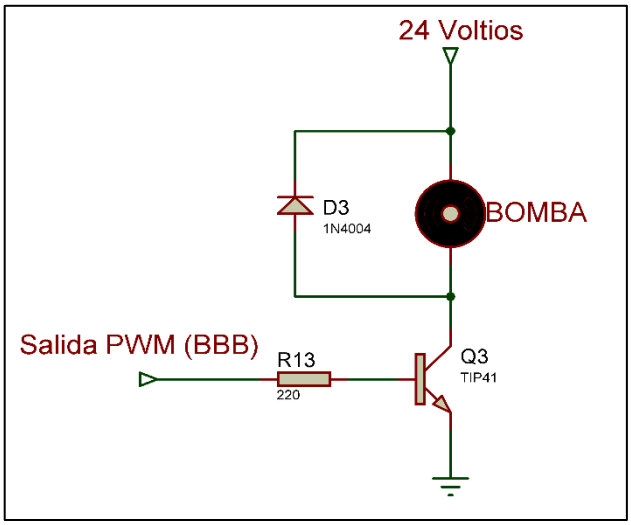


Fig. 21. Circuito de acondicionamiento de salida PWM para bomba de agua

4.5. Procedimiento de Diseño de Bloques de Funciones

Con el objetivo de desarrollar FBs bajo la Norma IEC-61499 basados en C++ para el runtime FORTE en el sistema operativo Debian que utiliza la Beagle Bone Black, se sigue el procedimiento como se presenta en la Figura 22.

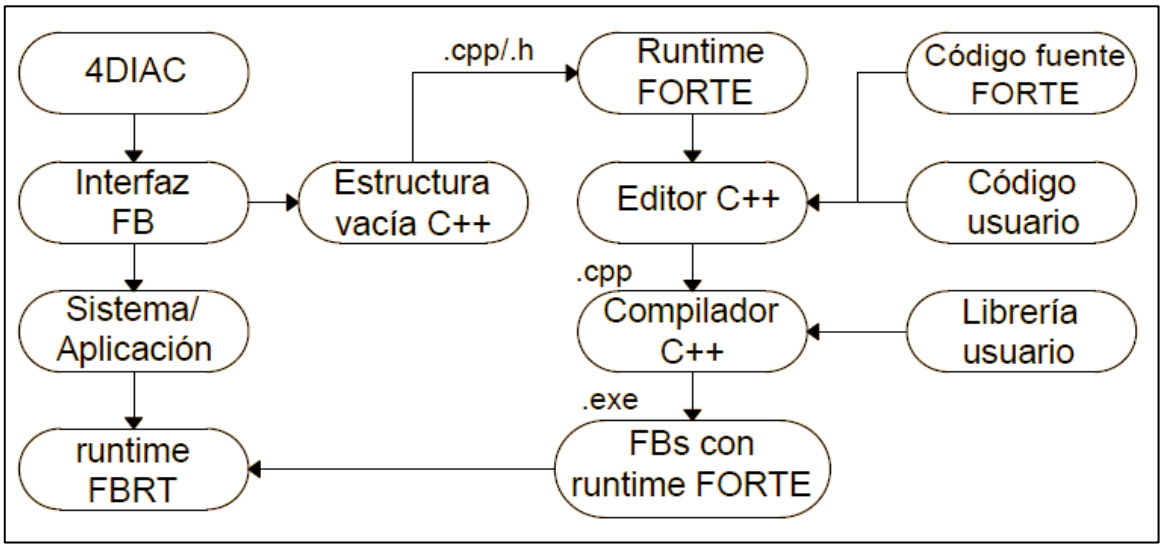


Fig. 22. Escenario de desarrollo del software en 4DIAC[1]

4.6. Bloques de Funciones desarrollados

Se han desarrollado 3 FBs utilizando el entorno de desarrollo 4DIAC los cuales permitirán interactuar a las aplicaciones desarrolladas basadas en la Norma IEC-61499 con las entradas analógicas y salidas PWM de la BBB.

4.6.1. Direccionamiento de pines de la entrada y salida

El direccionamiento de pines de las entradas analógicas y salidas tipo PWM se realizan utilizando un archivo de Lenguaje de Marcas Extensibles (xml), en el cual se encuentra: nombre de la variable de entrada analógica ó salida PWM, pin de la tarjeta seleccionado, el tipo de pin (entrada o salida) y un comentario. En la Figura 23 se representa el archivo xml realizado para la aplicación.

```
<?xml version="1.0"?>
- <esquema>
  - <variable>
      <nombre>SENSOR</nombre>
      <pin>AIN0</pin>
      <tipo>INPUT</tipo>
      <comentario>Lectura Sensor</comentario>
    </variable>
  - <variable>
      <nombre>MOTOR</nombre>
      <pin>P9_14</pin>
      <tipo>OUTPUT</tipo>
      <comentario>Salida PWM</comentario>
    </variable>
</esquema>
```

Fig. 23. Archivo xml de configuración de pines

4.6.2. Bloque de Función ANALOG_INPUT

Este FB fue diseñado para la lectura del valor del sensor analógico empleado en la aplicación.

En la Figura 24 se representa el FB ANALOG_INPUT, el cual tiene una entrada de datos tipo STRING llamada NAME, en donde se encuentra el nombre del sensor de la entrada analógico a manipular, además posee una salida de datos tipo REAL nombrada IN quien indica el valor del sensor analógico de la variable NAME.

Además este FB tiene dos eventos de entrada: el evento INIT quien se ejecuta inicialmente y está vinculado la variable NAME, el evento REQ que se ejecuta periódicamente para leer el valor real de la variable NAME de la tarjeta BBB. Igualmente

existen dos eventos de salida: el evento INITO indica que se completó la inicialización del FB y el evento CNF el cual procede a mostrar el valor de la variable NAME en la salida IN.

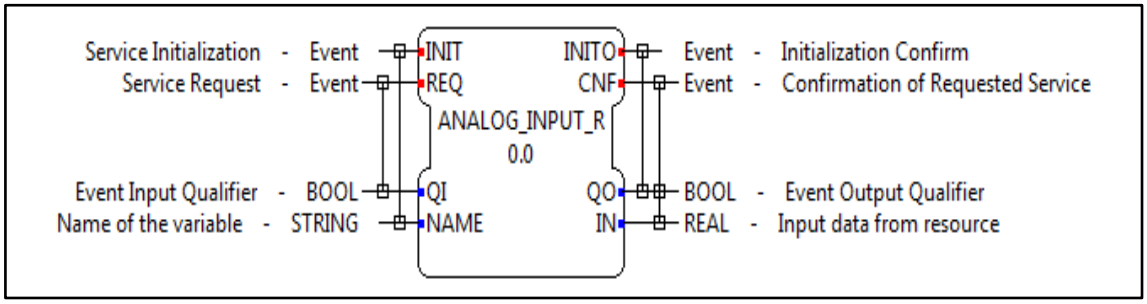


Fig. 24. Bloque de Función ANALOG_INPUT

4.6.3. Bloque de Función ANALOG_OUTPUT

Este FB fue diseñado para la escritura del valor en nuestra salida PWM empleada en la aplicación.

En la Figura 25 se representa el FB ANALOG_OUTPUT, el cual tiene una entrada de datos tipo STRING y una de tipo REAL, las cuales son: NAME donde se encuentra el nombre del actuador de la salida tipo PWM y OUT quien envía un valor a la salida tipo PWM de la BBB.

Además este FB tiene dos eventos de entrada: el evento INIT quien se ejecuta inicialmente y está vinculado la variable NAME, el evento REQ que se ejecuta periódicamente para asignar un valor real en la entrada de datos OUT de la BBB. Igualmente existen dos eventos de salida: el evento INITO indica que se completó la inicialización del FB y el evento CNF el cual indica que se ejecutó correctamente un evento REQ.

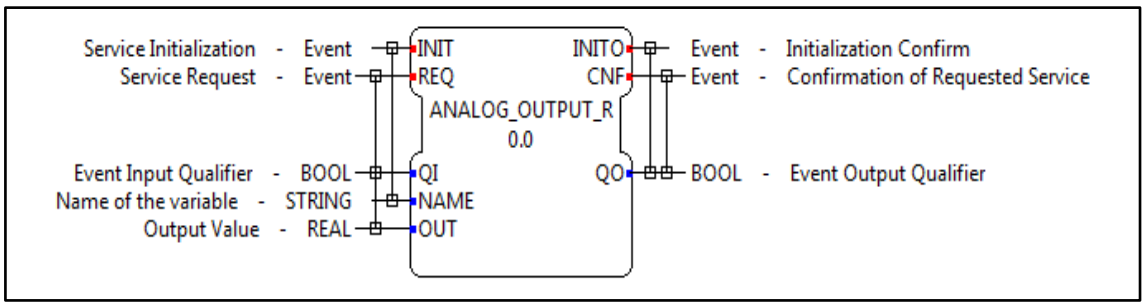


Fig. 25. Bloque de Función ANALOG_OUTPUT

4.6.4. Bloque de Función CONTROLADOR

El FB desarrollado para realizar los algoritmos de control del lazo cerrado para la célula FESTO seleccionada tiene el nombre de CONTROLADOR.

En la Figura 26 se representa el FB CONTROLADOR, este FB tiene cinco entradas de datos tipo REAL las cuales son: SETPOINT: lee el valor deseado, SENSOR: lee el valor del sensor de nivel, G_Kp: lee el valor de la ganancia proporcional, G_Ki: lee el valor de la ganancia integral y G_Ki: lee el valor de la ganancia derivativa además posee una salida de datos tipo REAL llamada CONTROL quien envía el valor de control al FB ANALOG_OUTPUT.

Además este FB tiene dos eventos de entrada: el evento INIT quien se ejecuta inicialmente y el evento REQ que se ejecuta periódicamente para leer el valor real de las variables antes mencionadas. Igualmente existen dos eventos de salida: el evento INITO indica que se completó la inicialización del FB y el evento CNF el cual procede a indicar el valor de la señal de control en la salida CONTROL.

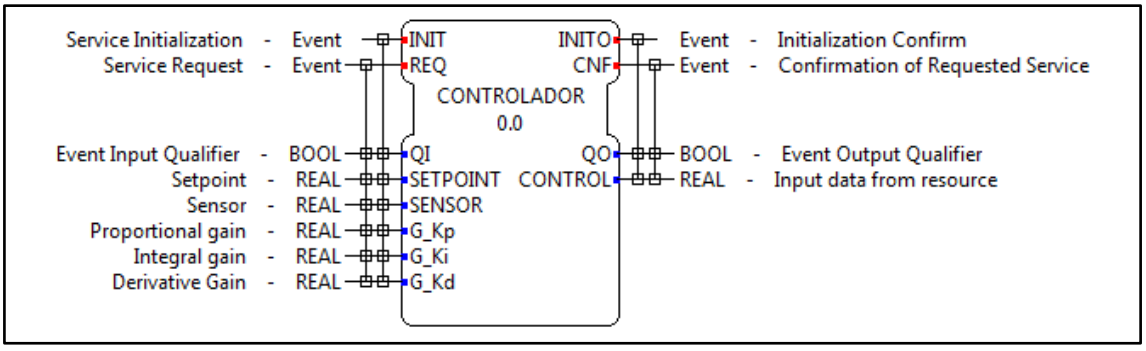


Fig. 26. Bloque de Función CONTROLADOR.

- **Implementación del algoritmo de control PID.**

Para implementar el controlador PID, se necesita encontrar algoritmos que se aproximen a la función de transferencia del controlador analógico, pero en tiempo discreto. El diagrama de este tipo de controlador es mostrado en la Figura 27.

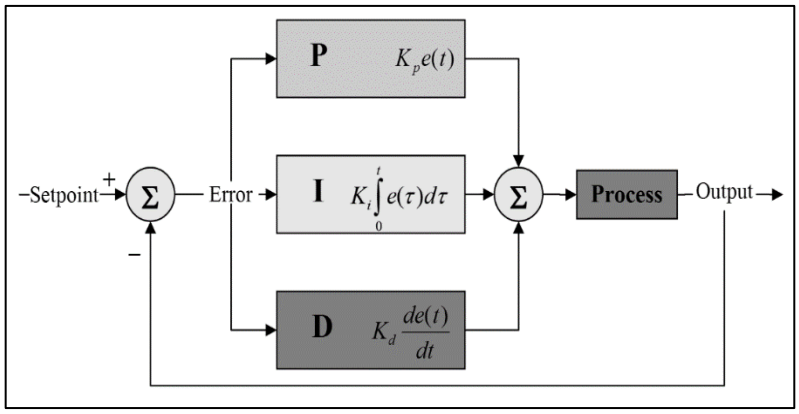


Fig. 27. Diagrama del controlador PID

El error es determinado por la diferencia entre el setpoint o salida deseada del proceso y la salida medida del sistema en determinado tiempo t. El error está determinado por la ecuación.

$$e(t) = G_d - G_k(t) \tag{4}$$

Donde G_d representa la salida deseada y $G_k(t)$ indica la salida medida.

La salida de corrección $x(t)$ se evalúa con la siguiente ecuación:

$$x(t) = K_p e(t) + K_i \int e(t) + K_d \frac{de(t)}{dt}\bigg|_{t=T} \tag{5}$$

Donde K_p, K_i y K_d son las constantes proporcional, integral y derivativa, respectivamente.

Reescribiendo la ecuación, $x(t)$ puede expresarse como:

$$x(t) = K_p e(t) + K_i \int_0^t [G_d - G_k(t)]dt + K_d \frac{de(t)}{dt}\bigg|_{t=T} \tag{6}$$

En el tiempo discreto $t = KT$, donde $K = 1,2,3,\ldots$ siento T el periodo de muestreo.

La salida de corrección o salida de control proporcional está representada por:

$$x_p(t) = K_p * e(KT) \tag{7}$$

La salida de corrección o salida de control integral está representada por:

$$x_i(t) = K_i\big(e(K) * T + e(K-1)\big) \tag{8}$$

La salida de corrección o salida de control derivativa está representada por:

$$x_d(t) = K_d\left(\frac{e(K) - e(K-1)}{T}\right) \tag{9}$$

En la Figura 28 se representa un flujograma del pseudocódigo del controlador PID implementado en el proceso de nivel de líquido.

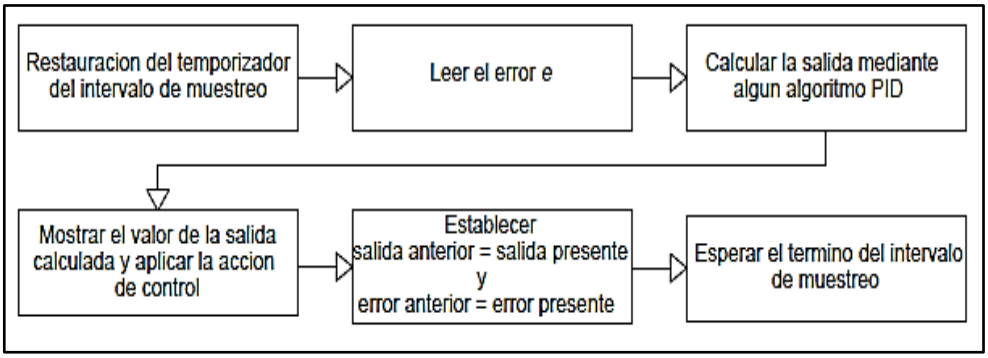


Fig. 28. Flujograma del pseudocódigo del controlador PID

El algoritmo del controlador PID programado se encuentra representado en la Figura 29.

```
Valor_setpoint = SETPOINT()*10;
    Valor_sensor = SENSOR()*10;
    Kp = G_Kp();
    Ki = G_Ki();
    Kd = G_Kd();
    ahora = clock();
    double CambioTiempo = ahora - pasado;
    if(CambioTiempo >= TiempoMuestreo){
            error = Valor_setpoint - Valor_sensor;
            errorPass = error*TiempoMuestreo + errorPass;
            errorD = (error-errorAnt)/TiempoMuestreo;
            P = Kp*error;
            I = Ki*errorPass;
            D = Kd*errorD;
            U = P+I+D;
        pasado = ahora;
        errorAnt = error;
        }
        if(U>100){
                U=100;
        }
        if(U<0){
                U=0;
        }
        CONTROL()= U;
```

Fig. 29. Algoritmo de control PID

4.7. Implementación del sistema de control

Se procederá a programar los FBs que serán posteriormente descargados en nuestro sistema el cual está compuesto por una tarjeta BBB y un PC donde se ejecutará una aplicación HMI sencilla para visualizar las variables principales del proceso e introducir el parámetro de referencia para el control PID. El caso de estudio queda se encuentra representado en la Figura 30.

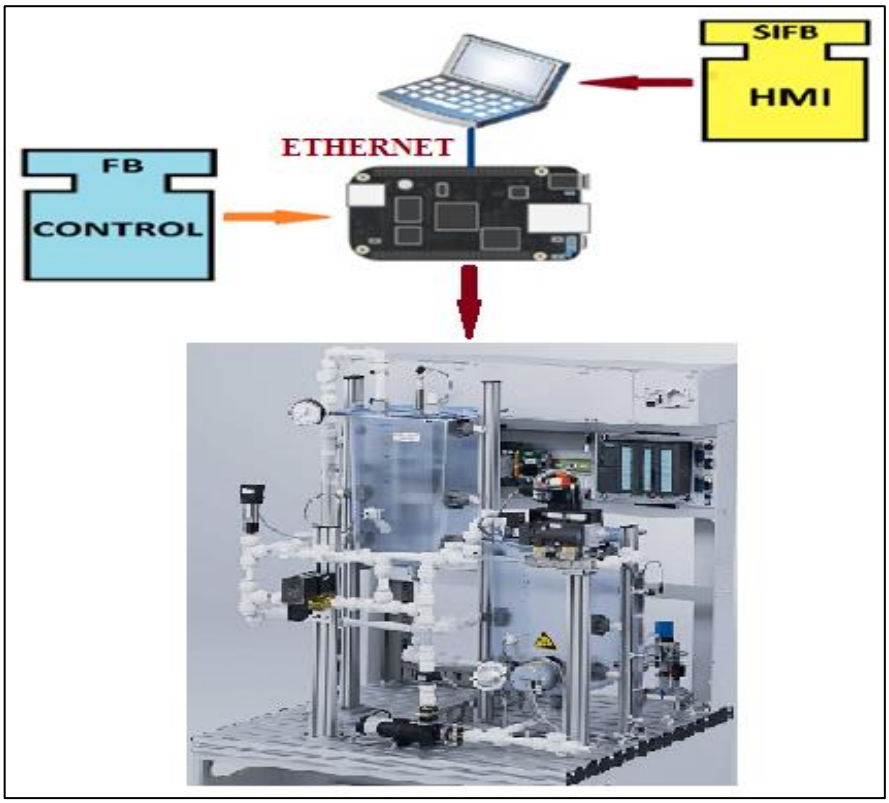


Fig. 30. Caso de Estudio (Control-Nivel)

4.7.1. Diseño de la Aplicación de Control

En la Figura 31 se representa la Aplicación de Control realizada en el entorno de desarrollo 4DIAC-IDE, contiene dos tipos de dispositivos detallados a continuación:

- El primer dispositivo será la estación HMI la cual está representada en color morado, este dispositivo permitirá la manipulación del Setpoint del sistema de control de lazo cerrado, adicionalmente indicará el valor del sensor de nivel y el valor de la señal de control enviada al motor.

- El segundo dispositivo será la tarjeta BBB y se encuentra de color marrón, quien contiene los FBs para la entrada analógica y salida tipo PWM de la tarjeta, además del algoritmo de control de sistema de lazo cerrado.

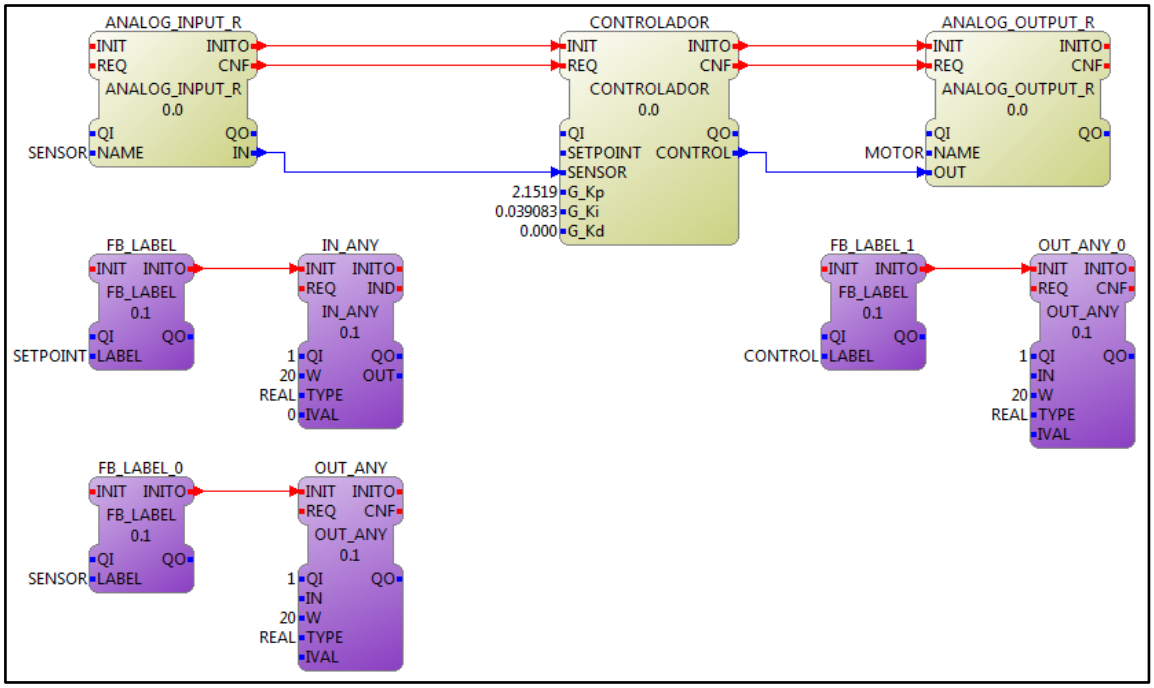


Fig. 31. Aplicación de Control en 4DIAC-IDE

4.7.2. Diseño de la Configuración del Sistema

El sistema de comunicación se configura con dos dispositivos con un único recurso y comunicados a través de una red Ethernet como se indica en la Figura 32.

- El primer dispositivo es un FBRT_WINDOWS con un único recurso PANEL_RESOURCE denominado HMI, en la configuración del dispositivo hay que indicarle los parámetros referentes a la pantalla de visualización que son el BOUNDS y el GRID. La IP en el puerto del manager MGR_ID es 192.168.1.11:61499.
- El segundo dispositivos se mapeará sobre la tarjeta BBB para el control de nivel, son de tipo BeagleBone Black y con un único recurso EMB_RES denominados BBB_NIVEL con una IP 192.168.1.10:61499 en el puerto del manager MGR_ID.

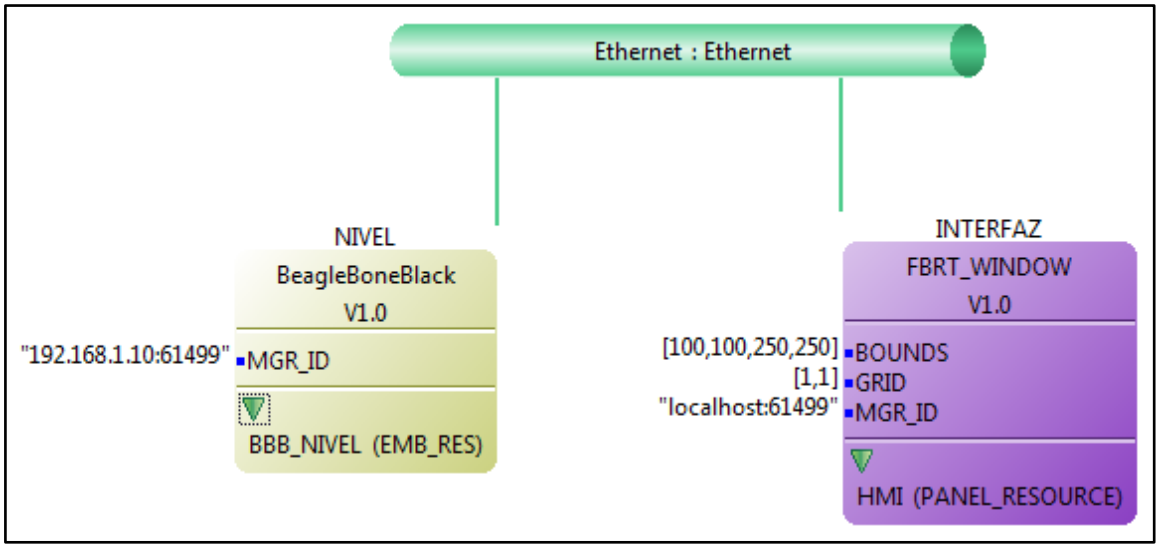


Fig. 32. Configuración del Sistema

4.7.3. Configuración del recurso HMI

Los FBs de color morado mapeados sobre el recurso HMI, permitirá obtener la interfaz gráfica para visualizar el funcionamiento de la aplicación. Este recurso se descargará sobre el runtime FBRT de 4DIAC-IDE, el cual se ejecuta en la PC para visualizar la aplicación HMI. Además contiene FBs de tipo SUBSCRIBE y PUBLISH los que nos permitirán:

- Enviar los valores del SETPOINT del HMI a la BBB.
- Visualizar los valores en el HMI del SENSOR de nivel y la señal de CONTROL enviada al motor.

Las IPs utilizadas son de tipo multicast, se envían los datos en grupos usando diferentes direcciones de IP y puertos. Además se conecta la señal COLD y WARM a cada entrada INIT de los FBs utilizados para inicializar la descarga del recurso al runtime FBRT.

Se agrega un FB llamado E_CYCLE el cual da el tiempo cíclico de ejecución del runtime FBRT, permitiendo que se actualicen los FBs para el HMI. En la Figura 33 se representa la Configuración del Recurso HMI para la aplicación.

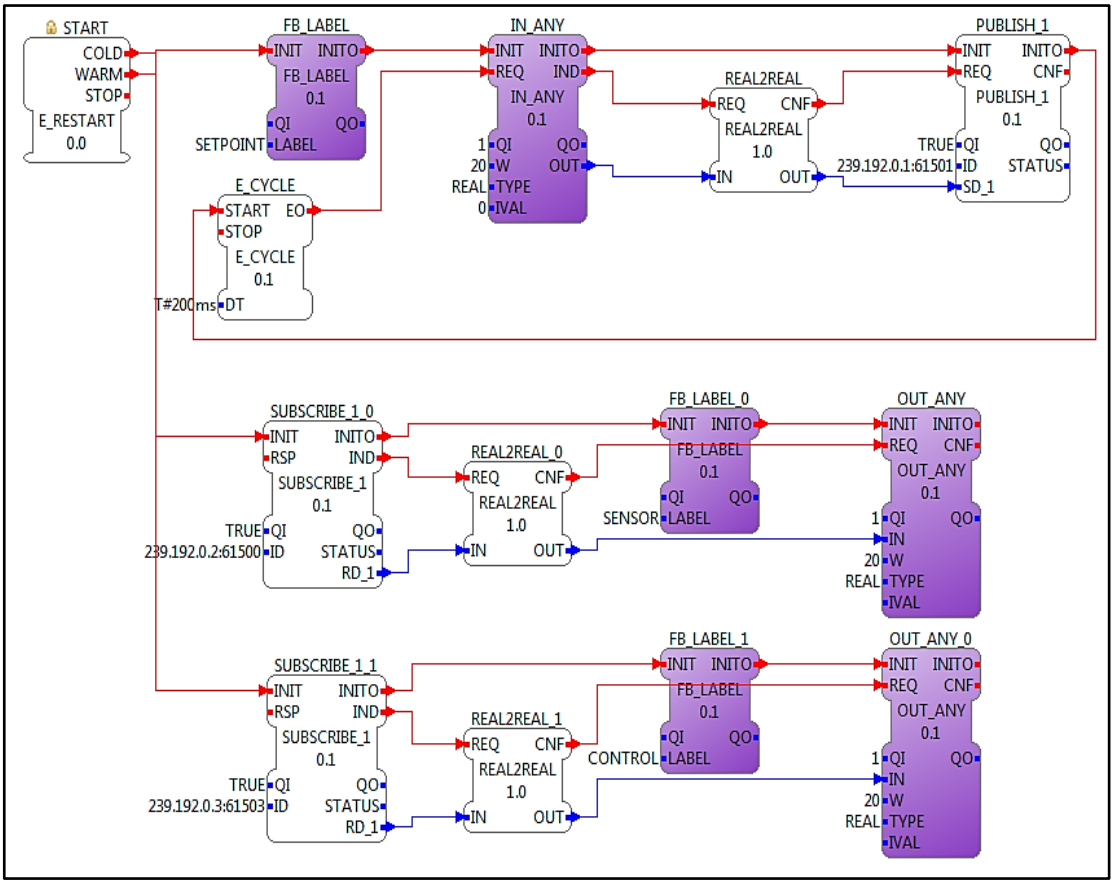


Fig. 33. Configuración del Recurso HMI

Este recurso al ser descargado al runtime FBRT que dispone 4DIAC-IDE nos proporciona una interfaz gráfica muy sencilla la cual se presenta la siguiente Figura 34.

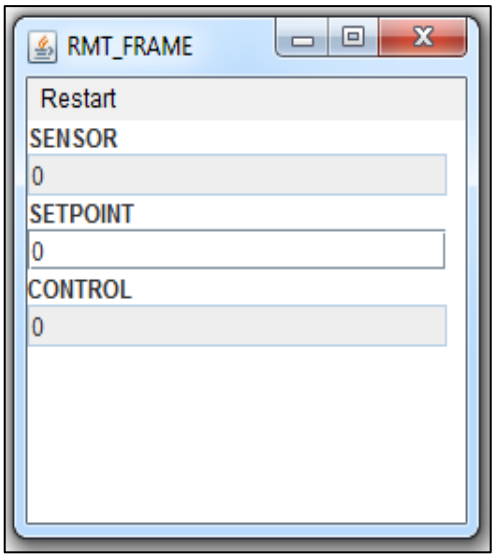


Fig. 34. Pantalla del Recurso HMI

4.7.4. Configuración del recurso BBB_NIVEL

Este recurso posee los FBs desarrollados para el algoritmo de control de la célula FESTO® MPS-PA Compact Workstation y la manipulación de la entrada analógica y salida tipo PWM de la tarjeta BBB. Este recurso se descargará sobre el runtime FORTE a través de 4DIAC-IDE, además contiene FBs de tipo SUBSCRIBE y PUBLISH los que nos permitirán:

- Enviar los valores del SETPOINT del HMI a la BBB.
- Visualizar los valores en el HMI del SENSOR de nivel y la señal de CONTROL enviada al motor.

Las IPs utilizadas son de tipo multicast, se envían los datos en grupos usando diferentes direcciones de IP y puertos. Además se conecta la señal COLD y WARM a cada entrada INIT de los FBs utilizados para inicializar la descargar del recurso al runtime FORTE.

Se agrega un FB llamado E_CYCLE el cual da el tiempo de cíclico de ejecución del runtime FORTE, permitiendo que se actualicen los FBs de la BBB. En la Figura 35 se representa la Configuración del Recurso BBB_NIVEL para la aplicación.

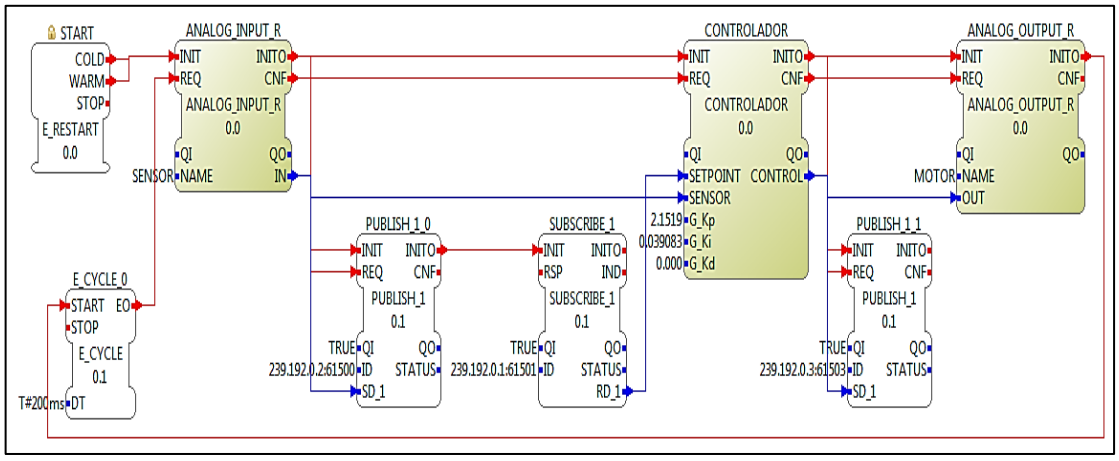


Fig. 35. Configuración de Recurso BBB_NIVEL

4.7.5. Sintonización del controlador PID

Con el fin de encontrar los valores adecuados de las constantes del controlador PID se desarrolló un programa con 4DIAC-IDE para realizar un experimento escalón, el cual trata de una aplicación HMI en el cual se dará una entrada escalón y se visualizará la

medida obtenida por el sensor. Los rangos de las señales del escalón y actuador están representados en la Tabla 5.

Tabla 5. Rangos de las señales del escalón y actuador.

Medición	Proceso	BBB(Voltios)	Porcentaje (%)
Señal Escalón	0 – 10 litros	0 – 3,3	0 – 100
Señal de Sensor	0 – 10 litros	0 – 1,8	0 – 100

Para capturar los datos de la entrada escalón y del sensor de nivel se utilizó un microcontrolador arduino UNO, además se utilizó la herramienta de PID Tuner de Matlab para encontrar los valores deseados, los datos capturados serán almacenado en el WorkSpace de MATLAB para un posterior análisis de los mismos como se indica en la Figura 36.

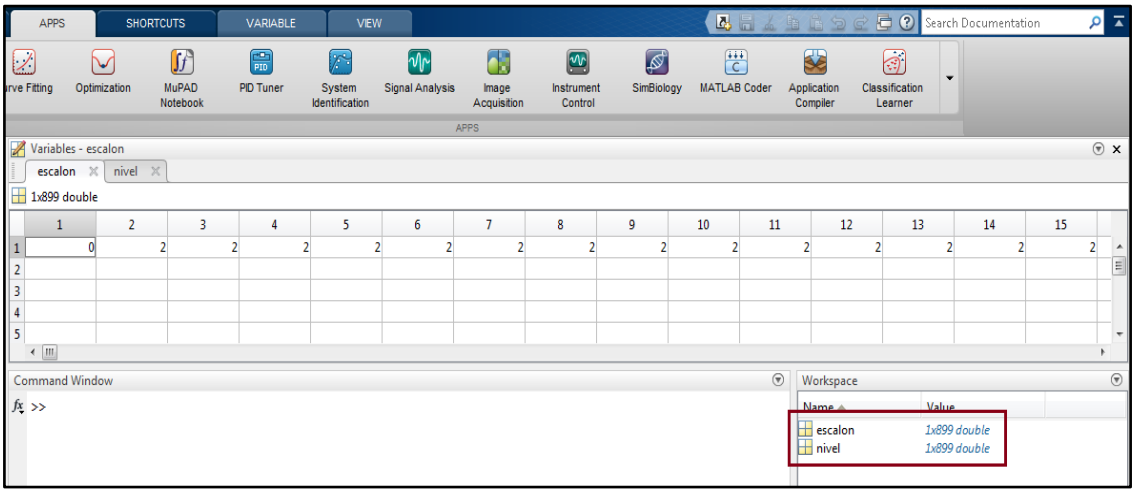


Fig. 36. Datos en el WorkSpace de MATLAB

La medición de las variables necesarias se las realizará del sensor de nivel de 0 a 5V y la salida EHRPWM de 0 a 3.3V, para esta señal es necesario realizar un filtrado para obtener el valor medio, mediante un filtro RC paso bajo que se conecta en paralelo al circuito de acondicionamiento de salida EHRPWM, los valores de los elementos son: R=680Ω y C=100µF como se indica en la Figura 37.

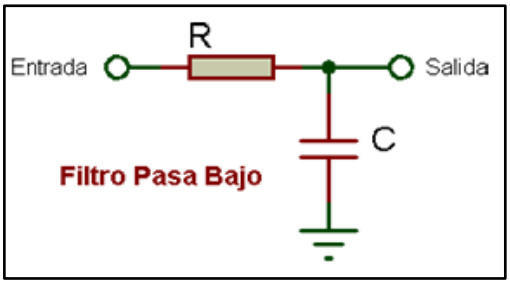


Fig. 37 Filtro pasa bajo

- **Experimento escalón**

Para el experimento, el escalón tiene un valor de 20%. En la Figura 38 se observa la respuesta del proceso con retardo ante el escalón. La línea de color rojo representa el escalón y la de color azul representa la respuesta que tiene el proceso.

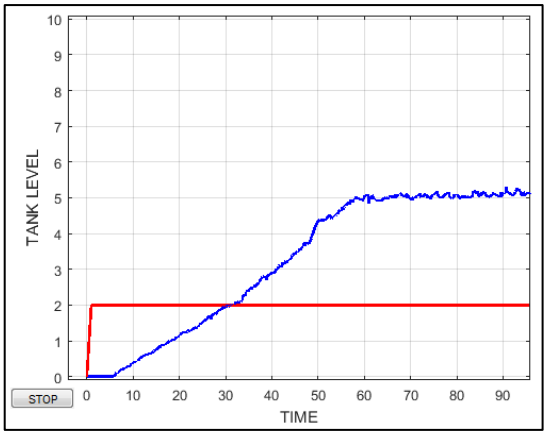


Fig. 38. Respuesta del proceso con retardo ante el escalón

- **Identificación del modelo de la planta**

Para la identificación del modelo de la planta se siguen los siguientes pasos.

1. En la pestaña "APPS" ir a la opción "System Identification" como lo indica la Figura 39, se abrirá la ventana indicada en la Figura 40.

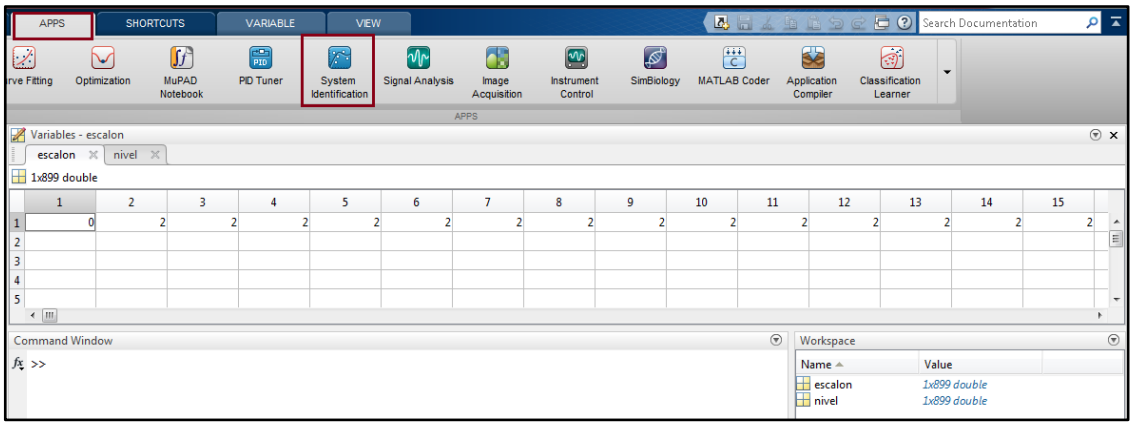


Fig. 39. Apps de MATLAB

2. En el menú de "Import Data" seleccione la opción "Time domain data...", se abrirá la ventana indicada en la Figura 40.

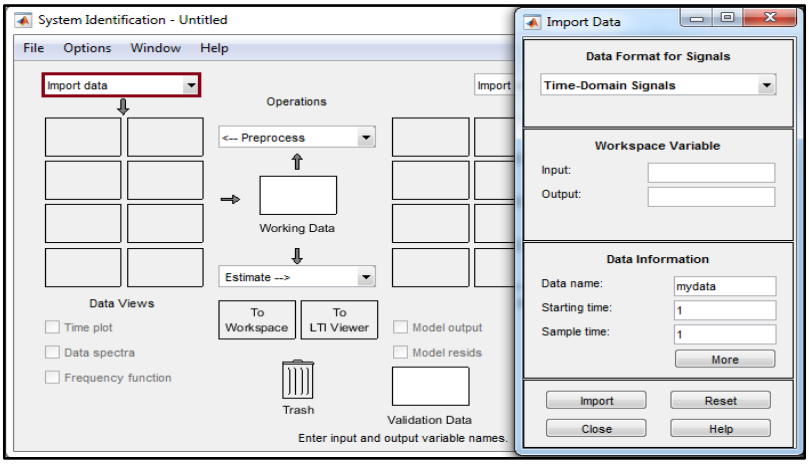


Fig. 40. Ventana "Import Data"

3. En la opción "Input" ingresar el nombre del vector de entrada, en "Output" el de salida, en "Data Name" el nombre para los datos, en "Starting time" colocar el valor de 1 y en "Sampling interval" el tiempo de muestreo, para nuestro caso particular de 0.2.
4. La opción "Import", agrega los datos a la ventana de "System Identification Tool", además se puede visualizar las gráficas con la opción "Time plot", donde aparecerán las gráficas de entrada y salida como se indica en la Figura 41.

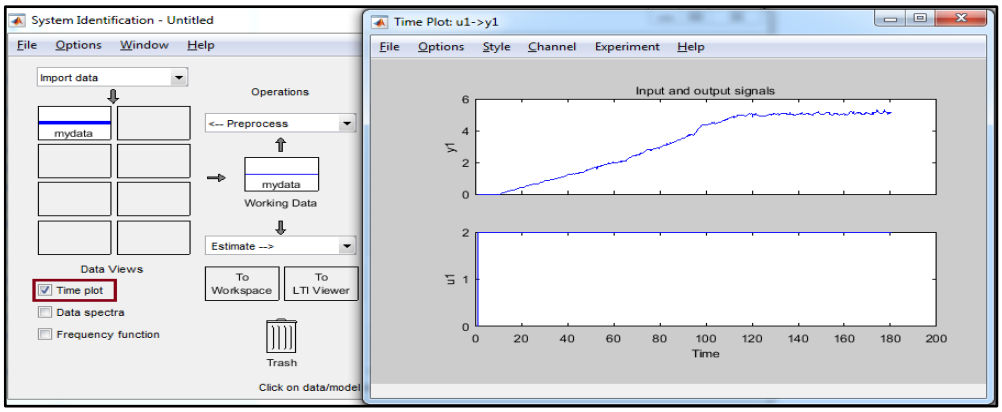


Fig. 41. Gráficas de entrada y salida

5. En la ventana "System Identification Tool", seleccione del menú "Estimate" la opción "Process Model", se abre la ventana de la Figura 42, donde se indica la función de transferencia del modelo, elegir la opción "Estimate" para estimar los valores de K, Tp1 y Td. La función de transferencia calculada para la implementación del controlador PID es:

$$G(s) = \frac{e^{-27*s} * 3{,}116}{71{,}93 * s + 1} e^{-27.5*s} \qquad (10)$$

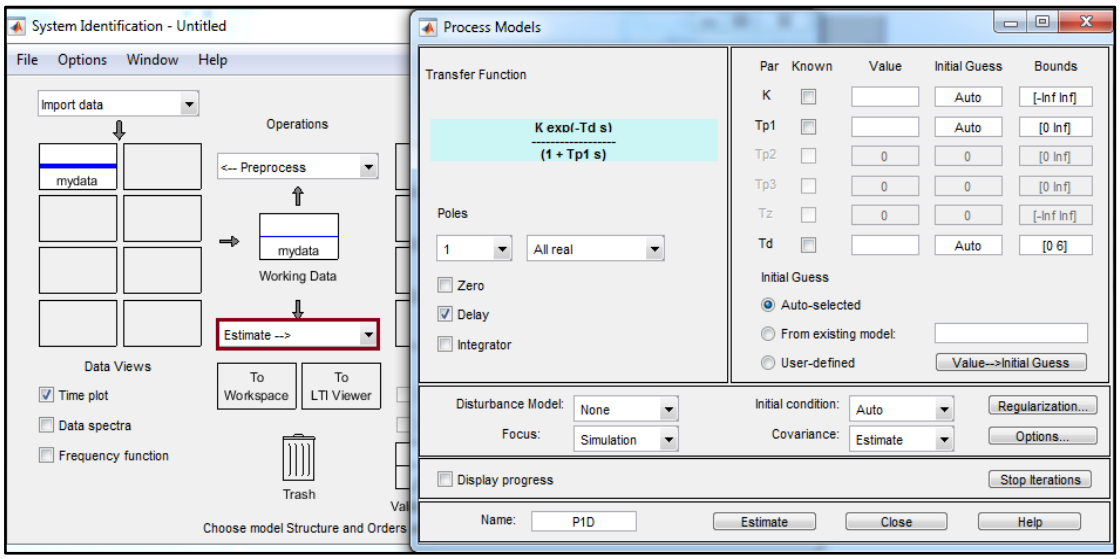


Fig. 42. Ventana Process Models

6. En la ventana "System Identification Tool" mediante la opción "Model output" se visualiza la respuesta de este modelo, la gráfica del modelo estará en color azul y la de planta en color negro la cual está representada en la Figura 43.

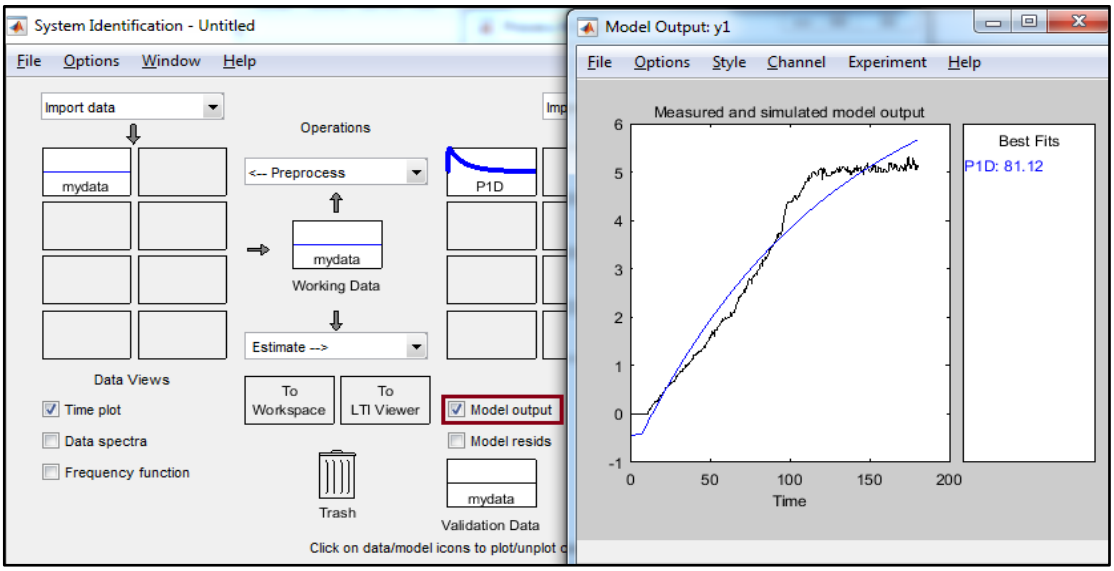


Fig. 43. Salida del modelo

7. En la ventana de "System Identification Tool" desplazar el modelo al bloque "To Workspace" para obtener la función en el espacio de trabajo de MATLAB como se indica en la Figura 44.

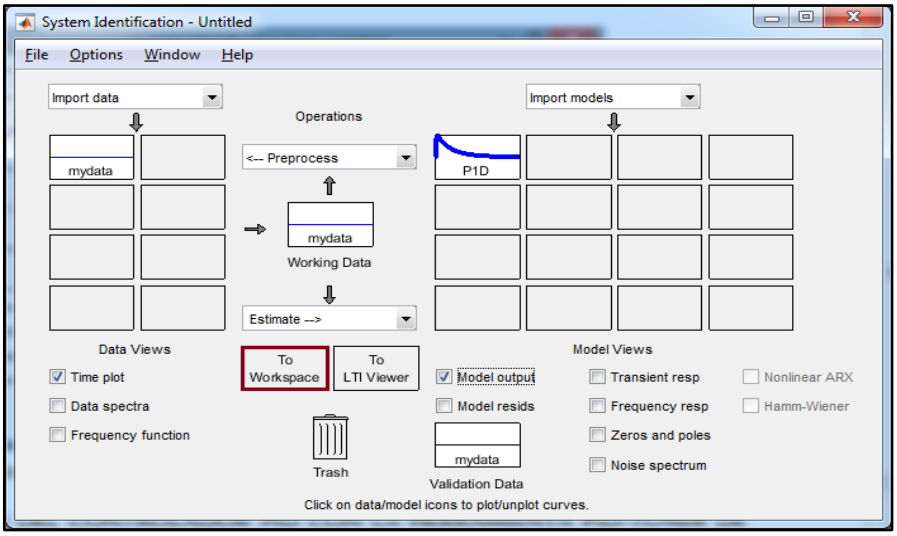


Fig. 44. Exportar función al espacio de trabajo de MATLAB

- **Calibración del controlador PID con la herramienta PID-TUNER de Matlab.**

El modelo exportado al espacio de trabajo es utilizado por la herramienta PID-TUNER para encontrar los parámetros de control del controlador PID a implementarse como lo indica la Figura 45.

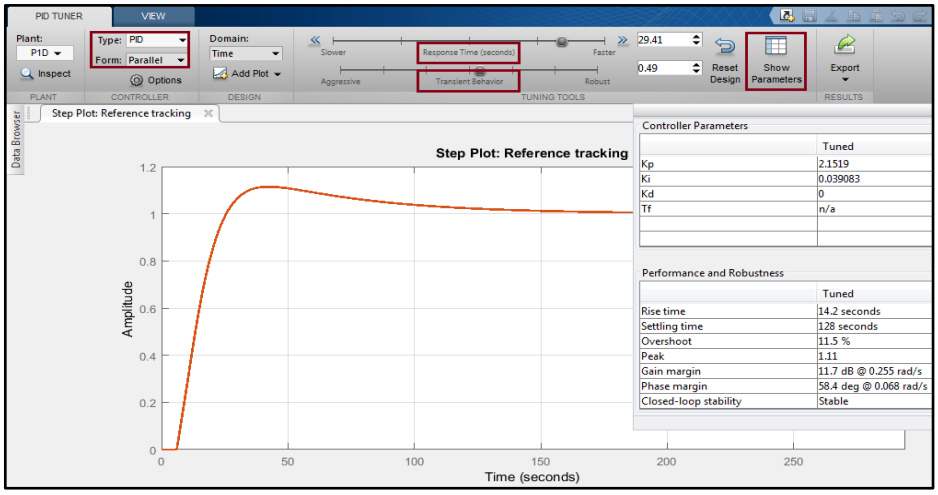


Fig. 45. Ventana PID Tuner

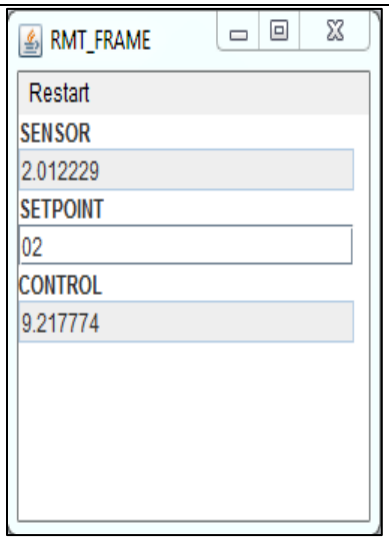
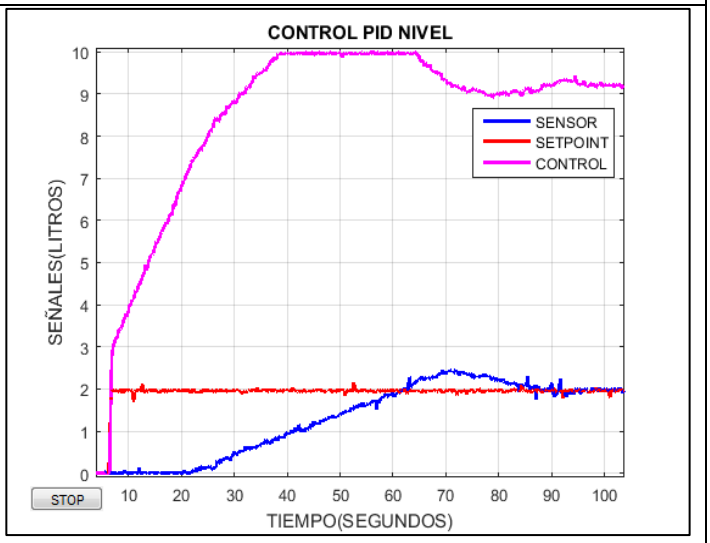
Los parámetros del controlador encontrados son: Kp=2.1519, Ki=0.039083 y Kd=0.0

4.7.6. Prueba de funcionamiento.

Para comprobar el funcionamiento del controlador PID implementado, se asignó un valor inicial de 2.0 litros al SETPOINT, el en Tabla 6 se observa al SENSOR alcanzar el valor

de 2.012229 litros después de aproximadamente 100 segundos, logrando así el valor de la consigna deseada, además se observa la señal CONTROL que es enviada por el controlador hacia la bomba el cual se encuentra en un 92.17774% de su capacidad, para una mejor visualización del funcionamiento se utilizó un ARDUINO UNO para la lectura de las variables, además se empleó MATLAB para la gráfica correspondiente.

Tabla 6. Prueba de funcionamiento con un SETPOINT de 2.0 litros.

Ventana (HMI)	Gráfica de variables
	

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Mediante el Entorno de Desarrollo 4DIAC-IDE se logró el diseñó de Bloques de Funciones basados en el estándar IEC-61499, para la lectura de los valores de sensores analógicos y para la manipulación de salidas de modulación por ancho de pulsos (PWM) de la tarjeta de desarrollo Beaglebone Black. Los Bloques de Funciones diseñados son posteriormente descargados en el runtime 4DIAC-FORTE y programados en lenguaje C++ para ser posteriormente ejecutados. El Bloque de Función diseñado para las entradas analógicas permite la lectura de sensores en un rango de 0.0-1.8 Voltios y el Bloque de Función de salida de modulación por ancho de pulso (PWM) genera un valor entre 0.0-3.3 Voltios.
- Se puede implementar una gran cantidad de algoritmos de control en los Bloques de Funciones descargados en el runtime 4DIAC-FORTE que fueron diseñados en el Entorno de Desarrollo 4DIAC-IDE, los algoritmos de control son programados en C++ el cual es un lenguaje de programación aceptado por la norma IEC-61499. El algoritmo de Control Proporcional, Integral y Derivativo (PID) implementado en el proceso de nivel de líquido se ejecutó correctamente alcanzando los valores deseados en la prueba de funcionamiento del sistema, demostrando así la factibilidad de utilizar Bloques de Funciones basados en la norma IEC-61499 para implementar algoritmos de control de procesos industriales con variables continuas en el tiempo.

- El Entorno de Desarrollo 4DIAC-IDE y el runtime FORTE han demostrado ser herramientas confiables para la implementación de sistema de control de lazo cerrado en tiempo real aplicando la normativa IEC-61499 en plantas de procesos industriales didácticos, igualmente la tarjeta de desarrollo BeagleBone Black ha confirmado ser un sistema embebido de bajo costo ideal para este tipo de aplicaciones. La implementación del sistema de control de nivel en la célula analógica FESTO® MPS-PA Compact Workstation se considera un paso más en el objetivo de demostrar los beneficios que ofrece el estándar IEC-61499. De igual manera, los resultados obtenidos demuestran que es viable continuar investigando con el objetivo de explotar todo el potencial que el estándar IEC-61499 ofrece a la industria.

5.2. Recomendaciones

- El archivo XML de configuración de los pines de entradas analógicas y salidas PWM de la tarjeta de desarrollo BeagleBone Black es sumamente importante, ya que cada Bloque de Función desarrollado como primer paso leerá este archivo para almacenar las variables de cada proceso. Si este archivo está mal configurado no se podría realizar la automatización del proceso, por tal motivo se recomienda configurarlo correctamente para evitar fallos en la implementación del controlador Proporcional Integral y Derivativo (PID).
- El intervalo de muestreo en el algoritmo de control es de mucha importancia, una frecuencia de muestreo demasiado baja puede degradar la estabilidad del sistema, además, se puede perder información debido a que la señales que se utilizan cambian rápidamente, por lo que no se estará trabajando sobre los datos actuales. Por el contrario, si la frecuencia es elevada, los conversores análogos-digitales (CAD) deben ser más rápidos y el volumen de información aumenta, por lo que se debe poner procesadores de mayores prestaciones. Por lo que se recomienda realizar diferentes pruebas de funcionamiento con distintos valores de intervalos de muestreo.
- Utilizar los Bloques de Funciones de Publicador y Suscriptor al momento de realizar la comunicación entre los bloques de función de la Interfaz Hombre Máquina (HMI) y los descargados en la Beaglebone Black (BBB), además tomar

en cuenta los valores que tendrán los “Event Input Qualifier” y los “Connection Identifier” para que el modelado en conjunto de Bloques de Funciones sea el adecuado.

BIBLIOGRAFÍA

[1] M. V. García Sánchez, "Implementación de Sistemas Empotrados de Control Distribuidos bajo el Estándar IEC-61499," Bilbao, 2013.

[2] C. Pang, S. Patil, C. W. Yang, V. Vyatkin, and A. Shalyto, "A portability study of IEC 61499: Semantics and tools," Proc. - 2014 12th IEEE Int. Conf. Ind. Informatics, INDIN 2014, pp. 440–445, 2014.

[3] A. F. R. Obando, "Diseño e implementación de un sistema de control distribuido en el banco de pruebas neumático de la escuela de Ingeniería Mecánica de la Universidad del Valle con base en la Norma IEC 61499," Santiago de Cali, 2016.

[4] L. Lednicki, J. Carlson, and K. Sandström, "Model level worst-case execution time analysis for IEC 61499," Proc. 16th Int. ACM Sigsoft Symp. Component-based Softw. Eng. - CBSE '13, p. 169, 2013.

[5] K. Thramboulidis, "IEC 61499 vs. 61131: A Comparison Based on Misperceptions," J. Softw. Eng. Appl., vol. 6, no. August, pp. 405–415, 2013.

[6] M. V. Gárcia, F. Pérez, I. Calvo, F. López, and G. Morán, "Desarrollo de CPPS sobre IEC-61499 Basado en Dispositivos de Bajo Coste," XXXVI Jornadas de Automática, pp. 230–237, 2015.

[7] A. Zoitl, T. Strasser, and G. Ebenhofer, "Developing modular reusable IEC 61499 control applications with 4DIAC," IEEE Int. Conf. Ind. Informatics, pp. 358–363, 2013.

[8] T. Rosenstatter, R. Wanger, S. Huber, T. Heistracher, and D. Engel, "Applicability of IEC 61499 for Event Based Smart Grid Applications," pp. 278–283, 2015.

[9] E. P. Patiño Ramon y J. . C. Solano Minchalo, "Diseño e implementación de un prototipo de supervisión de un sistema de control industrial utilizando plataformas empotradas de bajo costo y controladores lógicos programables PLCs," Cuenca, 2016.

[10] M. Jovanovic, S. Zupan, and I. Prebil, "Holonic control approach for the 'green'-

tyre manufacturing system using IEC 61499 standard," Journal of Manufacturing Systems, vol. 40, pp. 119–136, 2016.

[11] M. Wenger, A. Zoitl, and J. O. Blech, "Behavioral type-based monitoring for IEC 61499," IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, vol. 2015–Octob, 2015.

[12] F. Esteban Romero, Julio Ariel Estruch, Antonio M Romero, N. I.- Iec, E L Estándar Para, and A. A. L. Q. Control, E L Cnc, "Norma iec-61499 para el control distribuido. aplicación al cnc.," pp. 3–5, 2014.

[13] M. V. Garcia, F. Pérez, I. Calvo, and G. Morán, "Building industrial CPS with the IEC 61499 standard on low-cost hardware platforms," 19th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2014, 2014.

[14] I. Batchkova, G. Popov, H. Karamishev, and G. Stambolov, Dynamic reconfigurability of control systems using IEC 61499 standard, vol. 15, no. PART 1. IFAC, 2013.

[15] D. Li and Z. Zhai, "A Multi-dimensional Integrated Design Framework for CNC System," pp. 996–1001, 2016.

[16] R. Abrishambaf, M. Bal y V. Valeriy , "Distributed Home Automation System Based on IEC61499 Function Blocks and Wireless Sensor Networks," p. 6, 2017.

[17] J. C. Mercé Godoy, "Desarrollo de un sistema de control en red mediante el Estándar IEC-61499," Castellón de la Plana, 2015.

ANEXOS

Anexo1. Hojas de características de los componentes utilizados.

1. Hoja de características del BJT BC548C disponible en:

http://www.philohome.com/sensors/gp2d12/gp2d12-datasheets/bc548.pdf

2. Hoja de características del MOSFET IRF9530 disponible en:

http://www.alldatasheet.com/view.jsp?Searchword=Irf9530

3. Hoja de características del amplificador operacional LM741 disponible en:

http://www.alldatasheet.com/view.jsp?Searchword=Lm741

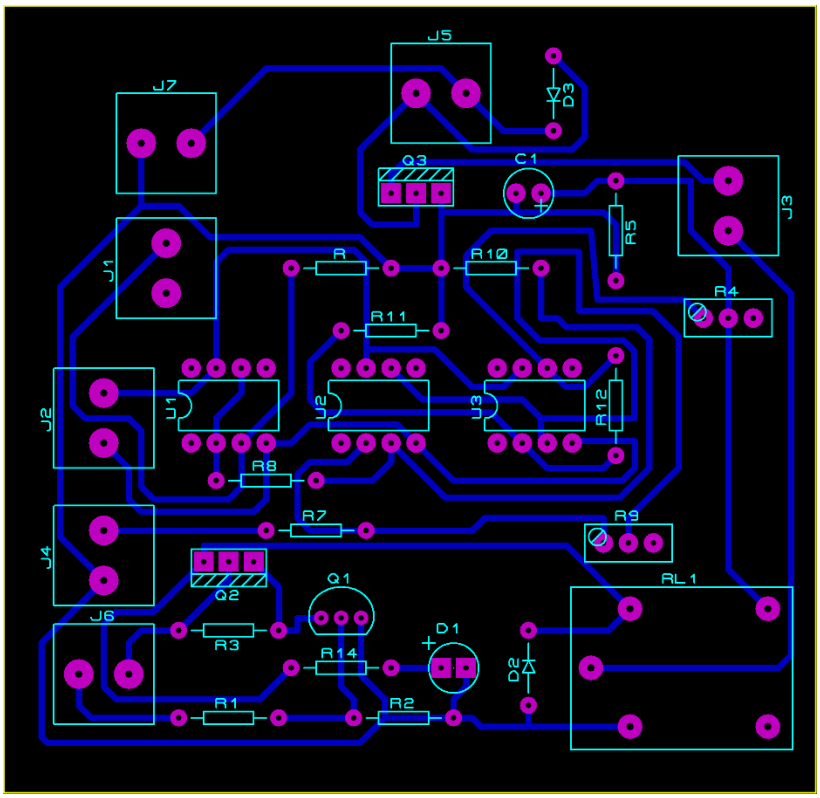
4. Hoja de características del transistor TIP41 disponible en:

http://www.alldatasheet.com/view.jsp?Searchword=Tip41

5. Hoja de características de la tarjeta BeagleBone Black disponible en:

https://cdn.sparkfun.com/datasheets/Dev/Beagle/e14%20BBB_SRM_rev%200.9.pdf

Anexo2. Circuito Impreso realizado en PROTEUS.



Materiales:

N°	Descripción	Cantidad
1	R1=10kΩ	1
2	R2=100kΩ	1
3	R3=10kΩ	1
4	R4=335Ω	1
5	R5=220Ω	1
6	R7=10kΩ	1
7	R8=1kΩ	1
8	R9=342Ω	1
9	R10=470Ω	1
10	R11=316Ω	1
11	R12=1kΩ	1
12	R13=220Ω	1
13	U1=U2=U3=lm741	3
14	Q1=BC548C	1
15	Q2=IRF9530	1
16	Q3=TIP41	1
17	D1=LED	1
18	D2=D3=1N4004	2
19	RL1=relay (5v)	1
20	J1=J2=J3=J4=J5=J6=J7=Borneras de 2 salidas	7