
ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

## DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

### CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

**TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO EN INSTRUMENTACIÓN**

# TEMA: CONTROL DISTRIBUIDO EN PROCESOS INDUSTRIALES UTILIZANDO SISTEMAS EMPOTRADOS DE BAJO COSTO, PARA VERIFICAR LA APLICABILIDAD DE LA NORMA IEC-61499.

**AUTORES:** ESTEBAN XAVIER CASTELLANOS NARVÁEZ

CARLOS ANDRÉS GARCÍA SÁNCHEZ

**DIRECTOR: ING. EDDIE GALARZA**

**LATACUNGA**

**2017**

**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN**

# CERTIFICACIÓN

Certifico que el trabajo de titulación, "**CONTROL DISTRIBUIDO EN PROCESOS INDUSTRIALES UTILIZANDO SISTEMAS EMPOTRADOS DE BAJO COSTO, PARA VERIFICAR LA APLICABILIDAD DE LA NORMA IEC-61499**" realizado por los señores **Esteban Xavier Castellanos Narváez** y **Carlos Andrés García Sánchez**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores **Esteban Xavier Castellanos Narváez** y **Carlos Andrés García Sánchez** para que lo sustenten públicamente.

Latacunga, 11 de diciembre del 2017

Ing. Galarza Zambrano Eddie Egberto.

**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN**

# AUTORÍA DE RESPONSABILIDAD

Nosotros, **Esteban Xavier Castellanos Narváez**, con cédula de ciudadanía N° 180438958-1 y **Carlos Andrés García Sánchez**, con cédula de ciudadanía N°180354304-8, declaramos que este trabajo de titulación "**CONTROL DISTRIBUIDO EN PROCESOS INDUSTRIALES UTILIZANDO SISTEMAS EMPOTRADOS DE BAJO COSTO, PARA VERIFICAR LA APLICABILIDAD DE LA NORMA IEC-61499**" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 11 de diciembre del 2017

Esteban Xavier Castellanos Narváez
**C.C.: 180438958-1**

Carlos Andrés García Sánchez
**C.C.: 180354304-8**

**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN**

# AUTORIZACIÓN

Nosotros, **Esteban Xavier Castellanos Narváez** y **Carlos Andrés García Sánchez**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "**CONTROL DISTRIBUIDO EN PROCESOS INDUSTRIALES UTILIZANDO SISTEMAS EMPOTRADOS DE BAJO COSTO, PARA VERIFICAR LA APLICABILIDAD DE LA NORMA IEC-61499**" cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Latacunga, 11 de diciembre del 2017

Esteban Xavier Castellanos Narváez
**C.C.: 180438958-1**

Carlos Andrés García Sánchez
**C.C.: 180354304-8**

# DEDICATORIAS

A mis padres definitivamente, su diaria preocupación por formar un hijo de bien han dado sus frutos y considero sumamente importante reconocerlo a través de este texto; ahora que pueden darse cuenta de que su objetivo fue cumplido, me honra poder dedicarles este trabajo. Es para ustedes, mentores de vida, porque su apoyo incondicional me demostró que se puede salir adelante si se quiere; y que nada se consigue fácilmente, sino que es necesario perseverar hasta alcanzar. Este es un logro que consigo y que por supuesto no será el único que podrán disfrutar conmigo. A ustedes increíbles GENIOS de vida, MAGNÍFICOS padres.

A ti hermano, porque con cada locura lograbas quitarme el cansancio y estrés, pues indudablemente sabías como cambiar mi panorama de las cosas para tomarlas con agrado y en forma más relajada; ahora entiendo que la vida está llena de altibajos y no solo de estrictas responsabilidades. He comprendido que soy una persona extraordinariamente afortunada por todo lo que mis padres me han inculcado y que a pesar de los años jamás han dejado de demostrar ese amor y apoyo incondicional como en cada una de las etapas de mi vida. En conclusión, gracias familia, sin ustedes el camino hubiera sido definitivamente mucho más difícil.

Esteban Xavier Castellanos Narváez.

*La ignorancia es atrevida* (Lic. Llanganate, por ahí por el 2014 más o menos)

Como esta es la única parte de este proyecto en la que puedo expresarme literariamente a mi gusto, prepárense para sorprenderse. Comprendo y entiendo que en esta parte del presente escrito van los muy famosos *este trabajo se lo dedico a…..* y aquí vienen los nombres de familiares, amigos, vecinos, conocidos, mascotas, etc, etc, etc; pero la brutal honestidad que ha hecho que mis amigos me comparen con una caricatura, me impide hacerlo.

La verdadera sorpresa que mencionaba es que básicamente esta no es mi dedicatoria, sino mi texto de disculpas por no poder ingresar una dedicatoria aquí. Verdaderamente no espero que la persona que se encuentra leyendo esto y que no sea parte de mi familia o amigos comprenda la razón de porque me estoy yendo contra el contenido tradicional que se debe poner en este apartado; pero para ser franco, con lo único que verdaderamente puedo comparar el sentimiento de esta etapa de mi vida, es con haber ganado un combate en contra de 5 cintas blancas (otra vez, creo que solo mi familia y amigos entenderán), es decir que básicamente tuve mis dificultades y todo pero gane porque mis capacidades me lo permitieron al final…. Más o menos en resumen creo que es una victoria vacía.

Con esto no quiero herir susceptibilidades ni dármelas de genio, pero a lo que me refiero es que el problema de este sentimiento yace en mí. La verdad es que envidio a gran parte de mis compañeros, porque sus malas noches las causaba ese sentimiento de que su meta, su sueño se les iba de las manos y debían quemarse las pestañas entendiendo un tema complejo o preparando los métodos de copia para el siguiente día. ¿Mis malas noches? Sesiones

de horas jugando lol, viendo ánime, leyendo libros (no con contenidos universitarios), practicando trucos de magia, conversando con ardillas y tigres; básicamente no dormía por pasar en lo que la mayoría de personas consideran *pendejadas.* No voy a decir que la tuve regalada, pero si por ahí tuve que esforzarme un 10% más, lo hacía porque la sensación que me invadía era la de cumplir con mi obligación y graduarme rápido.

Tampoco estoy diciendo que *mi familia me obligó* a ser ingeniero electrónico. Si me obligaron en algunas cosas fue en las necesarias para criar lo que ellos y yo (a estas alturas de mi vida) consideramos un hombre de bien, pero de ahí en decisiones como profesiones, novias, con qué gente llevarme, cositas así; ellos siempre me han apoyado. Es por esto que a veces no comprendo porque mi Mami dice que le he torturado durante 5 años. Mami, *tu nunca me obligaste a nada* el que tomó la decisión de con qué veneno matarse fui yo (figuradamente hablando), y la tome a sabiendas de cómo me sentiría al final. ¡Es más! Si yo decidí seguir ingeniería fue porque reconocía los beneficios de serlo, y sabía que tarde o temprano los resultados de esta decisión me ayudarían a reencaminarme en la dirección que quería, o quien quita que ¿a la final termine siendo otra cosa la que me apasione? e igual lo que saque de ser ingeniero me pueda ayudar a encaminarme en esa dirección.

En fin, creo que ya me alargué demasiado… Así que bueno, de nuevo me disculpo con el lector por haberle cruzado los cables al darle un texto de disculpas/explicación/cantinfladas en lugar de la clásica dedicatoria; ¡PERO TRANQUILOS! Algún día escribiré una historia increíble, o compondré una canción que no pueda salir de sus cabezas, o haré una ilusión que reanime su fe

en la magia permanentemente, o ganaré un torneo de TKD o LOL en donde se encuentren los adversarios más fuertes… Básicamente el día en que le ponga todo de mí para completar/crear/ganar algo que me apasiona, ese día tendrán su dedicatoria toda cursilona; de ahí… si quieren ponerse un poco sentimentalones lean mis agradecimientos.

Carlos Andrés García Sánchez

# AGRADECIMIENTOS

A Dios, mis padres, mi hermano y mi familia en general. Son ellos la fuente de inspiración diaria que tengo para ser mejor. Tengo excelentes ejemplos a seguir y espero convertirme en uno de ellos para las siguientes generaciones de mi familia.

Le agradezco infinitamente mamita por no dejarme desmayar, por el apoyo incondicional en todo sentido y por ser siempre supremamente sabia al momento de dar un consejo; considero que eso me ha permitido tomar todas las experiencias ajenas como propias para aprender de ellas.

Gracias padre mío, porque tu forma de ser me enseñó a ser responsable en mis obligaciones y porque con tu figura paterna he aprendido a ser un hombre que ama indescriptiblemente a su familia y por ella lo entrega todo. Porque al igual que mi mami me enseñaste a aprovechar y valorar todo lo que tenemos.

A ti ñaño, porque me sigues enseñando nuevas cosas y no te cansas de quererme a pesar de mi genio. Porque he sentido tu inmensa admiración hacia mí, pero aun no comprendes la magnitud de la mía hacia ti.

A mi novia Stefy, porque a lo largo de toda mi carrera universitaria me dio ánimo para vencer los obstáculos y continuar en el camino correcto. Por formar parte de mi vida y complementarla definitivamente.

Finalmente, un agradecimiento especial al Ing. MsC. Marcelo García PhD y al Ing. Eddie Galarza por todas sus enseñanzas que permitieron la correcta elaboración de este proyecto. No está por demás agradecer a mis amigos y compañeros de clase porque con ustedes todo este camino siempre fue mucho más ameno y llevadero.

Esteban Xavier Castellanos Narváez

*Para que puedan entenderme, déjenme primero contarles una historia*:

Había una vez una chica que se enamoró de un chico y su sentimiento fue tan grande que al cabo de un tiempo un bebé estaría por venir al mundo. Lastimosamente, en el corazón del chico las dudas y el miedo crecieron, hasta que un día se manifestaron en la decisión de no querer ser parte de las vidas de la chica y de su bebé; ése día, la chica comprendió que lo que en algún momento sintió por aquel cobarde no era el tan bello sentimiento conocido como “amor” que todas las personas buscan en su vida, pero sabía que la personita que estaba creciendo en su interior podría ayudarle a encontrarlo.

A la final la chica no estuvo sola, siempre tuvo el apoyo del resto de su familia, y día a día la chica se iba enamorando un poco más de esa personita. Todo mundo en la familia de la chica esperaba con ansias el día en que pudieran al fin conocer al bebé, pero el momento que debía ser pura alegría no duró mucho. El niño al fin salió a conocer el mundo exterior del vientre de su mamá, pero su salud fue decayendo como la estela que deja una estrella fugaz a su paso. Uno tras otro, cada doctor que revisaba al bebé no podía darles noticias alentadoras a la chica y a su familia, pero ella nunca quiso darse por vencida, sabía que debía haber algún modo.

Su corazonada era cierta, y de entre tantos doctores hubo uno que llego a dar una luz de esperanza a la familia. Su cura necesitaba de la sangre de la abuelita del bebé, y ella sin siquiera un ápice de duda, dio toda la que el bebé necesitó. Los días pasaban y tras la intervención del doctor, el niño poco a poco fue recuperando la salud. La alegría de la chica no podía ser medida y a pesar de saber el largo camino que debía afrontar ahora que era mamá, ella decidió nunca darse por vencida y tener su final feliz junto a su hijo. Los años pasaban y al niño nunca le hizo falta su progenitor, porque toda su familia: su mami, sus abuelitos, y sus tíos; le dieron con creces todo el amor que él necesitaba.

La mamá del niño sabía que debía trabajar para darle un buen presente a su hijo, y acabar sus estudios para poder darle aun un mejor futuro. Es por

esto que, desde lo más profundo de su corazón ella sacaba la fuerza para poder continuar con su lucha por alcanzar su final feliz junto a su hijo. Con más de un esfuerzo sobrehumano de la chica, poco a poco parecía que la historia que ella escribía para su hijo estaría llena de alegría; pero la vida tenía otros planes. Luego de muchos años de la primera pesadilla, una nueva estaría por comenzar. Un día el niño empezó a decaer de nuevo, y sus tranquilas noches comenzaron a volverse batallas contra delirios y dolores. La madre asustada, pidió ayuda al doctor que le dio esperanza años atrás, el cual, tras una larga búsqueda encontró a una amiga que podría saber cuál era el mal del niño.

Después de meses de incertidumbre y con la ayuda de la doctora, la madre tenía la oportunidad de al fin saber qué le sucedía a su hijo, pero eso no significó un alivio para su corazón. De entre los posibles males que afectaban a su hijo, ambos lo iban a acompañar el resto de su vida, pero con uno de ellos ese período podía llegar a su fin en un doloroso corto tiempo. El día de la última prueba llegó y las plegarias de la madre fueron escuchadas, para la fortuna de su hijo, el mal que debía acompañar a su hijo era el menor de los dos.

Con aún más esfuerzo que el que un humano daría, la madre se aferró a su sueño de escribir un final feliz para ella y su hijo y poco a poco los años volvieron a pasar. El niño se volvió un joven, aprendió a convivir con su enfermedad, y su vida estaba cada día más llena de felicidad. Poco a poco fue conociendo que hay personas que, a pesar de no compartir su sangre formarían parte de su familia; y con la llegada de su hermanita en parte pudo empezar a aprender sobre aquel amor por alguien a quien ayudas a crecer, que su madre pudo comprender con él. Los años siguieron pasando y aquel joven se volvió un hombre, el cual estaba a punto de acabar sus estudios como su madre lo hizo algún tiempo atrás, para así su historia poder continuar…

No es una de mis mejores historias, pero cuando quieres plasmar algo que sale de tu corazón, es mejor quedarte con lo primero que sale de él. ¿Y qué creen? Pues sí, si no han adivinado hasta este momento, aquel niño de

la historia era yo, y ahora sí, pueden comprender de mejor manera el que yo esté agradecido con la vida por las personas que puso en mi camino.

Por aquellos doctores que no perdieron la esperanza de que podría vivir más allá de lo que todos esperaban. Por aquellos maestros que sin importar que hayan sido malos, buenos o excelentes; me terminaron enseñando algo. Por aquellos amigos y hermanos de otras madres, con los cuales he compartido tantas locuras y tantos buenos momentos que ya he perdido la cuenta. Por mis primos que fueron mis primeros hermanos, cuando yo tanto deseaba tenerlos para ir en aventuras y protegerlos. Por mis tíos, que me quisieron lo suficiente como para llenarme de consejos esperando que sean una guía en mi vida. Por una tía abuela que me enseño la felicidad que puede traer a las personas una simple comida. Por mi hermanita, que me hace querer llegar a ser ese superhombre que piensa que soy ante sus ojos. Por mi ñaño, que más que un tío, ha sido esa figura de hermano mayor que tanto admiro, y que me encamina con las experiencias que él ya ha vivido. Por mi Fabi, que por su amor hacia mí, aparte de mi madre es la única mujer que puede decir que tengo su sangre en mis venas. Por mi papi, que me enseñó que padre no es quien te engendra, sino quien te cría. Y por aquella chica que dio un esfuerzo más allá de lo humano para poder cumplir su sueño de poder ver a esa personita que trajo al mundo convertida en un hombre con una vida feliz; porque el hombre que soy ahora, es gracias a todos ustedes.

Carlos Andrés García Sánchez

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# RESUMEN

Este proyecto de investigación propuso la aplicación de la norma IEC-61499 en módulos con bajas prestaciones que cuentan con sistemas empotrados con entornos de desarrollo de libre distribución. La ventaja de extender dicha norma consiste en la posibilidad de integrar dispositivos de diferentes casas fabricantes, dentro de un control distribuido con el 100% de compatibilidad entre ellos. Además, permitió realizar la escalabilidad hacia dispositivos más robustos tales como los controladores lógicos programables (PLCs), sensores y actuadores que se encuentran en los procesos industriales. Todas las pruebas de funcionalidad, eficiencia y eficacia de la norma en dichos módulos, quedaron demostradas mediante un sistema de control distribuido en dos procesos discretos. En primer lugar, se realizó el diseño de bloques de función para los sistemas empotrados de las tarjetas de bajo costo (Raspberry PI y BeagleBone Black) utilizando el estándar IEC-61499. Seguidamente, se procedió a realizar el diseño e implementación del hardware de acople de voltajes para las tarjetas mencionadas; como antecedente, las tarjetas tienen un nivel de voltaje de trabajo (3.3V o 1.5V), por lo que fue necesario realizar una tarjeta de acople a 24V por ser el valor de voltaje nominal al que se trabaja a nivel industrial. Una vez alcanzados los puntos anteriores, se procedió al diseño del esquema del control distribuido en donde se asienta el funcionamiento del estándar en las tarjetas mediante el control de un proceso discreto y, finalmente, se implementó dicho control en la maqueta FESTO FMS-200 en forma distribuida y con las tarjetas de diferente fabricante en donde se puede verificar la compatibilidad y funcionalidad de todo el sistema.

**PALABRAS CLAVE:**

- NORMA IEC-61499
- BLOQUES DE FUNCIÓN IEC-61499
- CONTROL DISTRIBUIDO
- TARJETA RASPBERRY PI
- TARJETA BEAGLEBONE BLACK

# ABSTRACT

This research project proposes the application of the IEC-61499 standard in modules with low benefits that have built-in systems with development environments of free distribution. The possibility of integrating devices from different manufacturers within a distributed control with 100% compatibility between them can be considered as the main advantage of extending this standard. In addition, it will allow scalability to more robust devices such as programmable logic controllers (PLCs), sensors and actuators found in industrial processes. All tests of functionality, efficiency and effectiveness of the standard in these modules were demonstrated by the application of the developed system in a distributed control system in two discrete processes.

First, the design of the function blocks for embedded systems of low cost cards (Raspberry PI and BeagleBone Black) using the IEC-61499 standard have been performed. Next, the design and implementation of the voltage coupling hardware was performed for the mentioned cards. At this point it must be known that cards have a voltage level to which they work (3.3V or 1.5V), so it was necessary to make a 24V coupling card because it is the nominal voltage value that is used in industrial processes. Once the above points have been reached, we proceed to the design of the distributed control scheme where the operation of the standard is performed on the cards through a discrete process control. Finally, this control is implemented in the model FESTO FMS-200 in a distributed form and with the cards of different manufacturers where it is possible to verify the compatibility and functionality of the whole system.

**KEYWORDS**:

- IEC-61499 STANDARD
- FBs in IEC-61499
- DISTRIBUTED CONTROL
- RASPBERRY PI and BEAGLEBONE BLACK

# INTRODUCCIÓN

El presente proyecto se encuentra dividido en seis capítulos, en los cuales se encuentran detalladas las actividades realizadas para la creación de la aplicación.

CAPÍTULO I

Se describe la situación actual del ámbito industrial con el planteamiento y formulación del problema, antecedentes, justificación e importancia, objetivos con los que se proporciona una visión más clara del alcance que posee el trabajo de investigación propuesto y finalmente se presenta la hipótesis y el cuadro de operación de variables.

CAPÍTULO II

En el capítulo II se presenta el marco teórico resultante de la investigación preliminar del presente trabajo de investigación. En el mismo se detalla un resumen de la información correspondiente a los estándares IEC-61131 e IEC-61499.

CAPÍTULO III

Se presenta el procedimiento a seguir para poder utilizar el sistema modular FESTO FMS-200 como planta de pruebas de carácter industrial y verificar la aplicabilidad del sistema a desarrollar. De igual manera se presentan las características de las tarjetas de bajo costo y tipo de elementos IEC-61499 a utilizar como base de la propuesta.

CAPÍTULO IV

En este capítulo se presentan los elementos generados lógica y físicamente como propuesta final del presente trabajo de investigación en conjunto con la metodología utilizada para su creación, necesarios para la implementación de un sistema de control distribuido.

CAPÍTULO V

En el capítulo V se describe la metodología necesaria para poder implementar un sistema de control distribuido basado en IEC-61499, los resultados de su funcionamiento y un análisis comparativo entre el proceso de desarrollo de un sistema de control IEC-61131 en contraste a un sistema de control IEC-61499.

CAPÍTULO VI

Finalmente, en este capítulo se muestran las conclusiones resultantes tras el desarrollo del presente trabajo de investigación.

# CAPÍTULO I

## 1. PROBLEMA

### 1.1. Planteamiento del Problema

Actualmente la tendencia en la automatización de los procesos o plantas industriales a nivel mundial implica el uso de la norma IEC-61131 como estándar general, sin embargo, el desarrollo tecnológico ha ido marcando a lo largo de los últimos 10 años nuevas necesidades de sistemas con requerimientos de interoperabilidad y que posean características de portabilidad y distribución; todo esto con el objetivo de brindar una producción mucho más rápida y eficiente.

Adicionalmente, los procesos requieren un alto costo de implementación que en ocasiones es inaccesible para el avance de las investigaciones, por lo que se requiere de la utilización de alternativas a escala más económicas que estén basadas en una arquitectura similar a la de los dispositivos más robustos.

Está claro que muchos de los dispositivos que actualmente se encuentran en el mercado siguen evolucionando mientras que aquellos que se encuentran en funcionamiento dentro de un proceso se quedan poco a poco desactualizados y con el tiempo llegan a ser obsoletos; esto implica que conforme los dispositivos evolucionan, deben incluir la posibilidad de ser actualizables tanto en software como en hardware con el objetivo de brindar una mayor rentabilidad en el tiempo para la industria.

Varias de las investigaciones realizadas a lo largo de los últimos años determinan que la norma IEC-61499 es un pilar fundamental que rompe los esquemas de centralización al momento de realizar un control, abriendo la puerta a los desarrolladores para iniciar trabajos que complementen la industria y cumplan los objetivos de portabilidad y distribución que hoy en día se necesitan dentro de los procesos.

### Formulación del Problema

Dado que la mayoría de los procesos industriales se encuentran controlados en forma centralizada, la mayor parte de ellos conllevan problemas cuando existe algún cambio por establecer; es por esto que se crea la necesidad de realizar el control en forma distribuida y asimismo que permita realizar la portabilidad e interoperabilidad con el resto de dispositivos existentes en el mercado.

La norma IEC-61499 es la propuesta presentada para solventar las falencias de su predecesora IEC-61131, permitiendo la fácil integración de dispositivos de diversas casas comerciales en sistemas distribuidos; la expectativa reside en su posible implementación en la mayor cantidad de controladores disponibles en el mercado. Sin embargo, al encontrarse ésta norma en expansión, presenta la problemática de estar en desarrollo para un número limitado de dispositivos; por lo tanto, esto permite abrir la posibilidad de superar esta restricción mediante el desarrollo de las funciones requeridas por el estándar para cada terminal.

## 1.2. Antecedentes

Los procesos de fabricación y producción se realizan cada vez más por sistemas y soluciones automatizadas teniendo como consecuencia, el aumento constante del nivel de automatización en las fábricas y plantas (ARTIST2, 2006). Un factor directamente racionado con el crecimiento del nivel absoluto de automatización en las diferentes plantas o sistemas de fabricación, es el aumento del nivel de complejidad al integrar en ellas un mayor y diverso número de: sensores, actuadores y controladores lógicos programables (PLCs) provenientes de diferentes casas comerciales. Incrementando de igual manera la dificultad para alcanzar requisitos de interoperabilidad, capacidad de fabricación y portabilidad entre elementos tan diversos.

Otra tendencia importante en la automatización industrial es la creciente necesidad de plantas personalizadas e individualizadas, lo que significa que

las líneas de producción tendrán que ser construidas y adaptadas a los nuevos procesos lo más rápidamente posible (Christensen, n.d.).

Creada y adoptada en 1992 con el fin de estandarizar los lenguajes de programación en el campo de la automatización industrial, la norma IEC-61131 ha sido durante varios años la principal norma en este ámbito, permitiendo diseñar en implementar sistemas de producción con mayor grado de reconfiguración y flexibilidad (Consortium, 2000). Con el paso del tiempo son varias las nuevas tecnologías que se han desarrollado y revelado dificultades al momento de implementar esta norma, marcando así un punto de inicio en la investigación y desarrollo de una solución para dicha problemática. Es por esto que en el año 2005 la Comisión Electrotécnica Internacional (IEC) lanzó la norma internacional IEC-61499 que actualmente aún se encuentra en proceso de consolidación. Esta norma es desarrollada como una metodología para modelar Sistemas de Control y Medida de Procesos Industriales (IPMCS) (Dai & Vyatkin, 2010) distribuidos y abiertos, presentando más funcionalidades que su predecesora y solventando de igual manera algunas de sus limitaciones. Su objetivo es gestionar la creciente complejidad de los sistemas de automatización de última generación mediante la obtención de una aplicación y configuración de hardware que no dependa de la casa fabricante.

El elemento central de la norma IEC-61499 es el FB (Bloque de Función) que permite la encapsulación de software de control. Los FBs pueden ser posteriormente enviados a los dispositivos de campo inteligentes (Dubinin & Vyatkin, 2006). El FB básico encapsula cierta funcionalidad tal como: control, operaciones matemáticas, comunicación, etc.; por medio de algoritmos. Para crear estos algoritmos de control se pueden utilizar los lenguajes estandarizados bajo la norma IEC-61131 o lenguajes de programación de alto nivel tales como C, C++.

Al trabajar bajo la norma IEC-61499, el software necesario para modelar un sistema de control distribuido se lo conoce como 4DIAC-IDE (Framework for Distributed Industrial Automation and Control), el cual además contribuir para la investigación y desarrollo del estándar, promueve la cooperación

entre los institutos de investigación y la industria. El objetivo de 4DIAC es la obtención de un entorno abierto de automatización y control basado en el estándar IEC-61499 proporcionando las siguientes características (Vyatkin & Hanisch, 2003):

- **Portabilidad**. Soporta e interpreta 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 herramientas de software de múltiples proveedores.
- **Reconfigurabilidad**. Es la habilidad para adaptar el hardware y software de control durante la operación del proceso.
- **Distribución**. La habilidad para distribuir componentes software en diferentes dispositivos hardware sin importar el proveedor, el cual, es un requisito necesario dado por la industria de la automatización.

## 1.3. Justificación e importancia

Mediante la verificación de la aplicabilidad de la norma IEC-61499 en sistemas empotrados en bajo costo se justifica la importancia de este proyecto desde el punto de vista de cuatro aspectos: industrial, tecnológico, educacional, y de la Secretaría Nacional de Planificación y Desarrollo.

Desde el punto de vista tecnológico, el desarrollo de este proyecto implica un gran beneficio al poder extender la norma a nuevos dispositivos. Al tener un mayor grado de interoperabilidad conjuntamente con sistemas de control distribuidos, poco a poco se rompe el paradigma implantado por la norma IEC-61131 al poder implementar dispositivos controladores de diversas casas comerciales en un mismo sistema, adicionalmente con la capacidad de configurarlos desde un mismo entorno de desarrollo.

Consecuentemente incidiendo en el aspecto industrial, al romper la exclusividad de recurrir a una sola casa comercial por equipos y dispositivos al momento de ampliar o modificar el sistema de control, los gerentes y administradores se benefician al adquirir un mayor número opciones de compra con más variedad de costos, permitiendo seleccionar solo lo necesario y reducir costos de implementación.

Desde el punto de vista educacional, al marcar un punto de partida en el estudio de nuevas tendencias en la automatización, este proyecto permitirá tener una guía de posibles temas de investigación que puedan ser implantados y estudiados en la universidad, manteniendo actualizada la malla de estudio permitiendo adicionalmente a los estudiantes poseer conocimientos de vanguardia que cumplan tendencias internacionales.

Finalmente, este proyecto encuentra su importancia al cumplir con las estipulaciones de la Secretaría Nacional de Planificación y Desarrollo al consolidar la transformación productiva de los sectores prioritarios industriales y de manufactura, con procesos de incorporación de valor agregado que maximicen el componente nacional y fortalezcan la capacidad de innovación y de aprendizaje colectivo. (SENPLADES, 2013)

La transformación de la matriz productiva supone una interacción con la frontera científico-técnica, en la que se producen cambios estructurales que direccionan las formas tradicionales del proceso y la estructura productiva actual, hacia nuevas formas de producir que promueven la diversificación productiva en nuevos sectores, con mayor intensidad en conocimientos, bajo consideraciones de asimetrías tecnológicas entre países. (SENPLADES, 2013)

## 1.4. Objetivos

### Objetivo General

Diseñar e implementar un control distribuido en procesos industriales utilizando sistemas empotrados de bajo costo, para verificar la aplicabilidad de la norma IEC-61499 en los procesos discretos de la maqueta FESTO FMS-200.

### Objetivos Específicos

- Investigar las directrices y formas de aplicación de la norma IEC-61499 para la automatización de procesos en sistemas empotrados a bajo costo.
- Diseñar e implementar dos grupos de FBs bajo el estándar IEC-61499 encargados de 1) manipulación de entrada y salida de datos, y 2) control; para los sistemas empotrados desde un entorno 4DIAC-IDE.
- Realizar las pruebas del estándar mediante el diseño e implementación de un control distribuido para los procesos discretos de maquetas FESTO FMS-200 bajo el entorno 4DIAC-IDE y el runtime 4DIAC-FORTE.

## 1.5. Hipótesis

El desarrollo de un sistema de control distribuido en dispositivos de bajo costo para procesos discretos de la maqueta FESTO FMS-200 permite verificar la aplicabilidad de la norma IEC-61499 en procesos industriales.

- **Variable Independiente:** *Desarrollo de un sistema de control en dispositivos de bajo costo para procesos discretos de la maqueta FESTO FM-200*
- **Variable Dependiente:** *Aplicabilidad de la norma IEC-61499 en aplicaciones industriales*

## Operacionalización de variables

*Tabla 1*

*Tabla de operacionalización de variables*

| Variables | Definición Conceptual | Dimensiones | Indicadores |
|---|---|---|---|
| **Desarrollo de un sistema de control en dispositivos de bajo costo para procesos discretos de la maqueta FESTO FM-200.** | Diseño e implementación de un sistema de control distribuido en dispositivos hardware de bajo coste mediante el desarrollo de software para que colaboren mutuamente y puedan controlar objetos físicos de los procesos por lotes de la maqueta industrial FESTO FM-200 | Diseño de FBs bajo la norma IEC-61499 para la manipulación de señales de entrada y salida. | FBs de manipulación control diseñados con sus entradas y salidas usando el software de distribución libre 4DIAC - IDE |
| | | Programación software de FBs en sistemas hardware distribuidos a bajo costo. | FBs programados usando el lenguaje C++ y software 4DIAC-FORTE |
| | | Validación de FBs diseñados para control de sistemas de producción por lotes | Verificación de funcionamientos de FBs desarrollados en el control de tres estaciones del sistema FESTO MPs 200 |
| **Aplicabilidad de la norma IEC-61499 en aplicaciones industriales** | Corroboración de la aplicabilidad de la norma IEC-61499 en sistemas de control a bajo costo de carácter industrial | Hardware | Adaptación de dispositivos a bajo costo como Raspberry PI y BeagleBone Black para control de procesos por lotes de carácter industrial |
| | | Software | Uso de arquitectura IEC-61499 en la programación de los FBs desarrollados |

# CAPÍTULO II

## 2. MARCO TEÓRICO

### 2.1. Estándar IEC-61131

El nacimiento del estándar IEC-61131 se dio con el propósito de normar el lenguaje con el que se deben programar a los controladores en cualquier aplicación industrial de automatización. Fue el primer peldaño en el camino de la estandarización de controladores e inclusive de los periféricos y lenguajes de programación que sufriría la industria. El objetivo general de la estandarización de dichos dispositivos y lenguajes fue mejorar la experiencia de usuario que poseen los ingenieros al realizar la programación de los mismos aun cuando no tienen la suficiente experiencia y conocimiento para el manejo de los controladores. (Otto & Hellmann, 2009)

Si bien es cierto, los Controladores Lógicos Programables (PLCs) vieron la luz aproximadamente desde la década del 60 y con ellos, los lenguajes de programación conjuntamente con la diversidad de controladores fueron apareciendo provenientes de varias casas comerciales a lo largo y ancho del planeta. Todo este crecimiento paulatino de los PLCs se produjo gracias a los estudios constantemente realizados por grupos de investigación especializados, sobre todo de PLCopen quien innovó y creó más y mejores dispositivos en cada lanzamiento.

La norma IEC-61131 pretendía establecer una forma general para la programación de controladores por lo que fue dividida en cinco partes que se explican a continuación (Otto & Hellmann, 2009):

1. Vista general.
2. Hardware.

3. Lenguaje de programación.
4. Guías de usuario.
5. Comunicación

Sin embargo, es de interés especial en este proyecto la tercera parte puesto que existe un amplio trabajo realizado que se explica como sigue: IEC 61131-3 es el resultado del esfuerzo de varias multinacionales especializadas en el campo de la automatización industrial. Este apartado consta de 200 páginas con más de 60 tablas en cuyo contenido se describe el modelo de software con la estructura de un lenguaje de programación; adicionalmente incluye la especificación de la semántica y sintaxis del mismo.

La tercera parte del estándar IEC 61131 estandariza cinco lenguajes de programación dentro del campo de la automatización industrial, permitiendo independizar la programación de los fabricantes de los dispositivos. Son tres lenguajes gráficos y dos textuales estandarizados cuya aplicación se ha realizado hasta los últimos años sobre todo en aquellos lenguajes gráficos (Ramanathan, 2014). Dichos lenguajes son los siguientes:

**Lenguajes Textuales:**

- Lista de Instrucciones (IL, Instruction List).
- Texto Estructurado (ST, Structured Text)

**Lenguajes Gráficos:**

- Diagrama de contactos (LD, Ladder Diagram)
- Diagrama de Bloques Funcionales (FBD, Function Block Diagram)
- Gráfica de Función Secuencial (SFC, Sequential Function Chart)

Sin embargo, la semántica de estos lenguajes no está estrictamente definida, por lo que se presenta el problema de incompatibilidad entre softwares de los fabricantes de los dispositivos.

## Modelo de Software IEC-61131

El modelo software de este estándar está representado en capas con características propias. A continuación, se detallan los elementos necesarios para proporcionar el entorno de software de un PLC (Sunder, Zoitl, Christensen, Steininger, & Fritsche, 2008).

- **Configuración.** Corresponde al sistema del PLC en donde se encuentra el software específico para un cierto problema de control.
- **Recurso.** Proporciona el soporte para la ejecución de los programas; puede declarar variables, tareas y programas asociados a las tareas.
- **Tarea.** Controla la ejecución de programas y bloques funcionales.
- **Unidades de organización de Programa (POU).** Son las funciones, bloques funcionales y programas; las funciones mantienen su objetivo de aceptar entradas y emitir salidas. El cuerpo del bloque funcional es un algoritmo que procesa los datos y está escrito en uno de los lenguajes estandarizados en IEC 61131.
- **Variables Globales y Locales.** Pueden ser declaradas en configuraciones, recursos o programas. Esto permite su uso dentro de programas o FBs

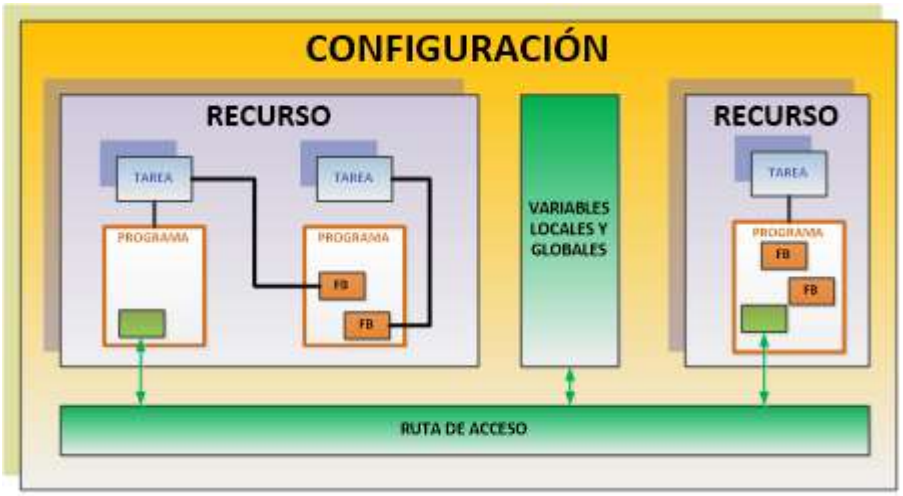



*Figura. 1 Modelo Software IEC-61131-3*

Como se observa en la Figura. 1, en el nivel más alto, el elemento software requerido para solucionar un problema puede estar definido como una configuración. Las configuraciones existentes son únicas para cada sistema de control, en ellas se puede encontrar las características del hardware.

Dentro de una configuración, se pueden definir uno o más recursos. Se puede entender el recurso como un procesador capaz de ejecutar programas IEC 61131 en donde se definan una o más tareas.

Las tareas se encargan de controlar el proceso de ejecución de los programas y/o bloques de función. Generalmente son activadas en forma periódica o a través del cambio de estado de una variable que funciona como un disparo.

Los programas pueden ser escritos en cualquiera de los lenguajes establecidos en IEC 61131-3 y constan de varios elementos software tales como funciones y bloques funcionales en donde se encuentra la declaración de variables y un conjunto de instrucciones para ejecutarse. Un programa es la interacción entre dichos elementos software con el fin de intercambiar datos entre ellos.

Si realizamos una comparación de todo el modelo de software de la norma IEC 61131 y un PLC convencional, claramente se marca la diferencia en la complejidad de las estructuras que soporta el estándar versus el manejo de un solo recurso que ejecuta una sola tarea que contiene un único programa controlado en forma cíclica.

## Justificación de un nuevo Estándar

El estándar IEC 61131 presenta una gran desventaja basada en la semántica de sus elementos del lenguaje, convirtiéndola en un problema cuando se requiere interpretar un mismo código bajo otra herramienta de software. Este problema se debe a falta de claridad en la semántica de los elementos del lenguaje existentes en el apartado tres del estándar IEC 61131. Por esta razón es imposible transferir la configuración de una herramienta de software hacia otra provocando la pérdida total de la capacidad de portabilidad.

Adicionalmente, existe un problema en la reconfiguración por inconsistencias en las herramientas de software mas no del estándar. Los inconvenientes se suscitan por dos razones: (1) La falta de especificación

para la creación dinámica de nuevos recursos dentro de una configuración y, (2) El intercambio de algoritmos sobre la marcha que no se encuentra definido dentro del estándar. No obstante, y ante la necesidad latente de reconfiguración, se han realizado varias soluciones entre las que figura que, para dispositivos de gama media y baja se prefiere eliminar esta característica debido a que no existe demanda en los usuarios, mientras que para los controladores de gama alta la reconfiguración es posible, aunque el costo de los dispositivos sea un tanto elevado.

Los fabricantes de herramientas IEC 61131-3 tienen su manera de implementar los conceptos y FBs para la comunicación entre PLCs definidos en el apartado cinco del estándar, sin embargo, se convierte en un problema dicha implementación debido a que cada fabricante lo realiza de manera diferente provocando que la característica de distribución se vea un tanto afectada.

Con todo lo mencionado anteriormente, la industria actual requiere de la implementación de un nuevo estándar capaz de solucionar todos los problemas mencionados e incluso aquellos que suponen un grado de complejidad más alto y que la norma IEC 61131 no alcanza a resolver.

## 2.2. Estándar IEC-61499

Como consecuencia de la necesidad de la industria, se crea el estándar IEC 61499 orientado hacia el control distribuido de los procesos. El avance de la tecnología suponía una mejor arquitectura y requisitos de herramientas de software por lo que el aparecimiento de un estándar completamente renovado y mejorando las características del IEC 61131 fue creado. La primera versión del diseño del estándar fue aprobada en agosto de 2005 por el comité técnico 65 de medida, control y automatización de procesos industriales (TC, Technical Committe) quienes a la vez pertenecen a la IEC (Kaghazchi, Joyce, & Heffernan, 2007). En tal versión, se define una arquitectura genérica para el control distribuido, así como una guía para el uso del Bloque Funcional (FB) en Sistemas de Control y Medición de Procesos Industriales Distribuidos (IPMCSs).

El propósito general de la norma IEC 61499 se basa en: (1) Romper el esquema de sistemas homogéneos, volviéndolos heterogéneos mediante la integración de dispositivos de control provenientes de diferentes casas comerciales y (2) habilitando la capacidad de reconfiguración dinámica, lo que implica realizar modificaciones en la configuración mientras la aplicación se encuentra sobre la marcha.

El estándar IEC 61499 es considerado (por sus bondades y fortalezas) como la siguiente generación de estándares orientado a sistemas de automatización, en donde se cuenta con la capacidad de interoperabilidad, portabilidad y reconfiguración faltantes en su predecesora IEC 61131-3. Lamentablemente y por el corto tiempo de vida de la norma, aún se encuentra en desarrollo pese a que un gran número de trabajos de investigación ya han demostrado la validez y mejora que se obtiene al usar este estándar.

## Especificaciones IEC -61499

El estándar IEC 61499 se divide en los siguientes 4 apartados (R. W. Lewis, 2001):

- **Arquitectura.** Es el primer apartado de la norma en donde se encuentran los requisitos generales, definiciones y modelos de referencia. Las reglas para la declaración de los tipos de FBs y para su comportamiento.
- **Requisitos de herramienta software.** Es el segundo apartado en donde aparece la definición de los requisitos de herramientas software y la especificación de los tipos de FBs.
- **Manual informativo.** Es el tercer apartado y contiene la información necesaria de la arquitectura IPMCS y de las herramientas software para que cumplan con las especificaciones del estándar.
- **Reglas y Perfiles de Conformidad.** Es el cuarto apartado en donde se encuentra la definición de las reglas que permiten la implementación de los apartados IEC 61499-1 e IEC 61499-2.

## 2.3. Arquitectura

Para mejorar el entendimiento de la organización del sistema o proceso, así como los componentes del mismo, el estándar propone una arquitectura de manera que se define en forma genérica y jerárquica todos los modelos. Dichos modelos son independientes del dominio y extensibles por medio del uso de FBs. A continuación, se detallan los modelos existentes dentro del estándar:

### Modelo de Bloque Funcional (FB)

El bloque funcional (FB) es considerado como el elemento “célula” dentro de un sistema de control. Consta de una cabeza que se encuentra conectada al flujo de eventos cuya función es leer las entradas, analizarlas y posteriormente generar las salidas apropiadas. Todo esto bajo el concepto de eventos, tanto en la entrada como en la salida (R. W. Lewis, 2001). Cada FB basa su comportamiento dinámico en la Gráfica de Control de Ejecución (ECC, Execution Control Chart) que permite la entrada y salida de eventos.

El FB dentro del estándar IEC 61499-1, se mantiene en estado de reposo hasta que reciba una señal de activación en forma digital para generar una entrada, esto permite la activación del FB e inmediatamente se producen los eventos de entrada y salida tal como se muestra en la Figura. 2.

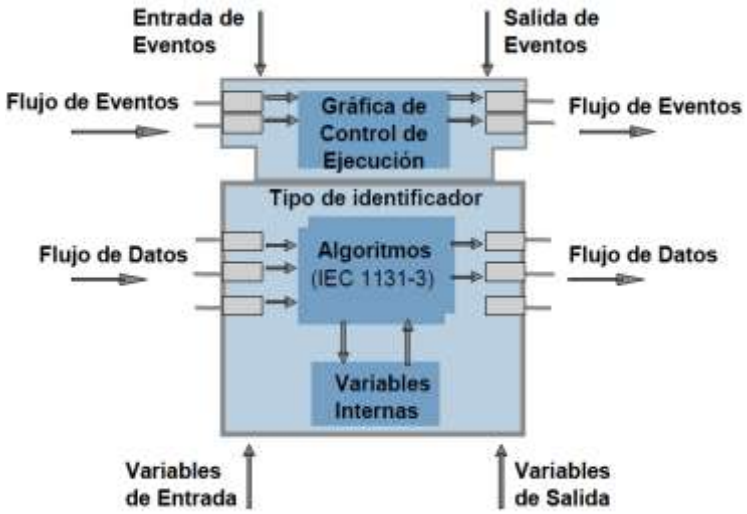



*Figura. 2 Modelo de Bloque de Función IEC-61499*

El comportamiento del sistema está descompuesto en partes pequeñas denominadas estados cuya validez radica en un conjunto de condiciones que se deben cumplir. El ECC es el encargado de ayudar al programador a tratar dichos estados que a su vez están ligados a uno o más algoritmos conjuntamente con eventos de salidas. La activación de un estado significa que se ejecutarán los algoritmos que se encuentren definidos en él.

Existen tres diferentes tipos de FBs que se describen a continuación (Std, 2005):

- **FB Básico:** Es el tipo de FB más sencillo en donde solo se encuentran 2 partes: el ECC y los algoritmos.
- **FB Compuesto:** Este tipo de FB es un poco más completo debido a que conlleva una red de instancias de varios FBs interconectados.
- **FB Interfaz de Servicio:** Es el tipo de FB más completo con el que se puede realizar interacción entre aplicaciones y recursos, además de proporcionar los servicios para cada aplicación.

Un FB funciona por medio de los algoritmos que internamente han sido programados; el lenguaje de programación puede variar entre cualquiera de los cinco definidos en IEC 61131-3: IL, ST, LD, FBD o SFC. Sin embargo y en IEC 61499 se puede programar en lenguajes de alto nivel como: C, C++,

Java o incluso Delphi. La función de cada algoritmo es procesar las entradas con los datos internos para poder emitir salidas.

## Modelo de Recurso

Las funciones dentro del modelo de recurso proporcionan la aceptación, procesamiento y retorno de los eventos y/o datos de las interfaces de comunicación y proceso, así como el control independiente de operación y los servicios a las aplicaciones como se muestra en la Figura. 3 (Std, 2005).

El recurso en esta norma está modelado por tres elementos:

- **Aplicación Local (o parte local de aplicación distribuida)**: Contiene las variables y eventos de entrada y salida de los FBs que ejecutan operaciones.
- **Interfaz de Proceso:** Ejecuta un mapeo, entre las aplicaciones y la interfaz de proceso, de eventos y datos por medio de los bloques de función de interfaz de servicio (SIFB)
- **Interfaz de Comunicación:** Realiza la misma acción que la interfaz de proceso, pero esta vez el mapeo es entre las aplicaciones y la interfaz de comunicación utilizando de igual forma los SIFBs.

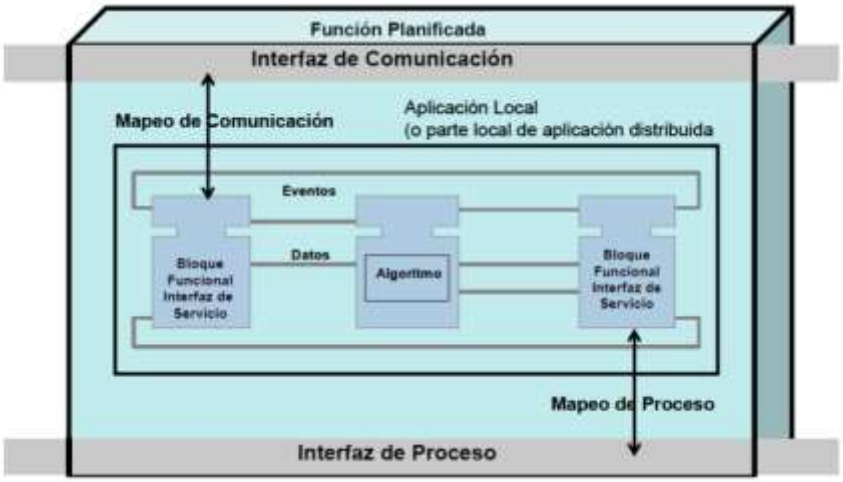



*Figura. 3 Modelo de recurso IEC-61499*

## Modelo de Dispositivo

Es considerado como un contenedor de recursos en donde existe el entorno para la ejecución de aplicaciones. En él existen dos tipos de interfaces: (1) Proceso, en donde se encuentran las entradas y salidas de los dispositivos y (2) Comunicación, en donde se utilizan los diferentes protocolos para la comunicación con otros dispositivos y/o aplicaciones como se muestra en la Figura.  4. Según (Strasser, Sünder, Zoitl, Rooker, & Brunnenkreed, 2007) el modelo físico es una entidad física independiente, capaz de realizar una o más funciones específicas en un contexto particular delimitado por sus interfaces.

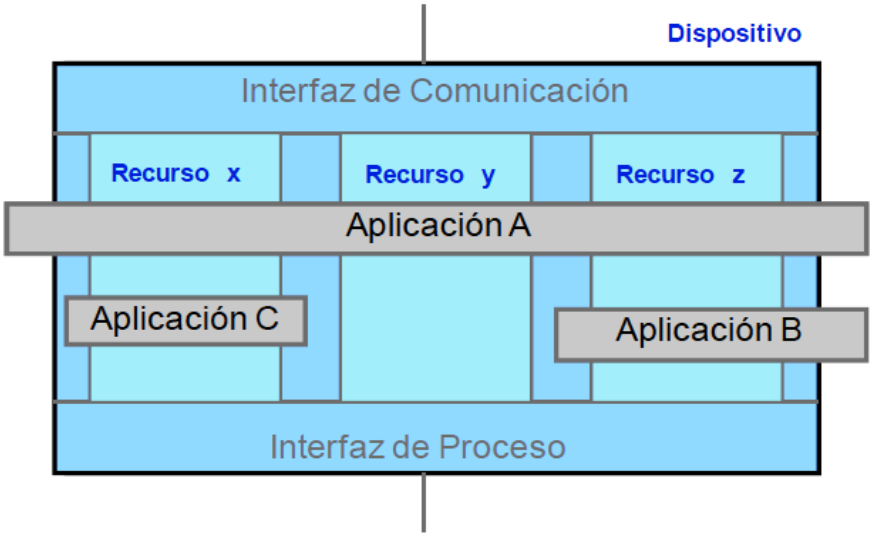



*Figura.  4 Modelo de dispositivo IEC-61499*

## Modelo de Sistema

El modelo de sistema recolecta un número de dispositivos que están interconectados entre sí mediante una red de comunicaciones con segmentos y enlaces cuyo objetivo es formar un conjunto de cooperación entre aplicaciones (Strasser, Sünder, Zoitl, Rooker, & Brunnenkreed, 2007). La Figura.  5 muestra el modelo en forma gráfica para un mejor entendimiento.

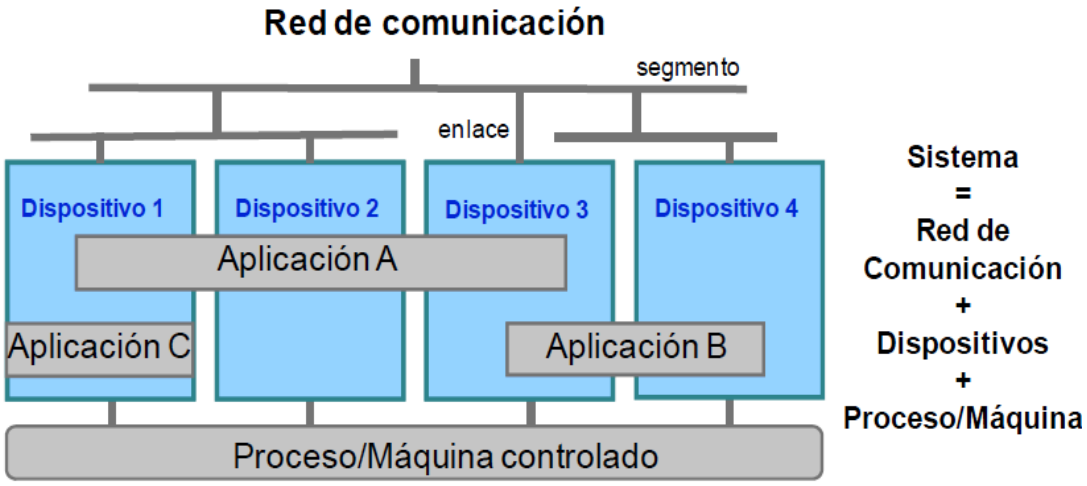



*Figura. 5 Modelo de sistema distribuido IEC-61499*

## Modelo de Aplicación

Es una unidad funcional de software que puede distribuirse entre varios recursos del mismo o diversos dispositivos, así como de la misma o varias aplicaciones. Aprovecha las especificaciones de la aplicación para producir una respuesta correcta de los eventos tanto de la interfaz de proceso como la de comunicación. Adicionalmente permite la modificación de variables, generación de eventos según las necesidades e inclusive la interacción entre la interfaz de procesos y comunicación mediante la programación y ejecución de algoritmos internos.

Una aplicación debe estar definida mediante el flujo de datos de la red de FBs que se compone, en ella debe constar de igual manera el flujo de eventos que inicializan las variables, los que ejecutan los algoritmos internos y aquellos que emiten salidas o entradas como información hacia el siguiente FB tal como se muestra en la Figura. 6.

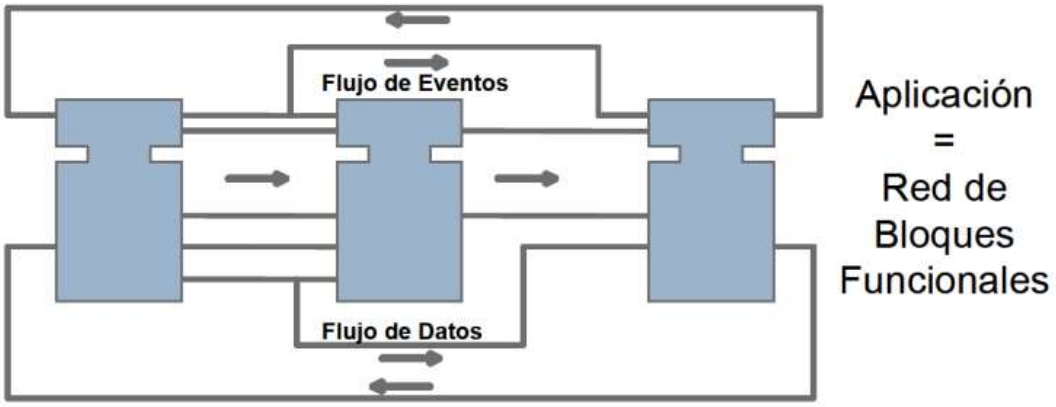



*Figura. 6 Modelo de aplicación IEC-61499*

## Modelo de Distribución

Es la parte final de la aplicación bajo la norma IEC 61499-1 en donde cada FB será mapeado acorde a los dispositivos de control desde donde serán inicializados y ejecutados. Cabe resaltar que por medio de la norma se abre la compatibilidad a muchos más dispositivos permitiendo el control de un proceso en forma distribuida mediante la ejecución de FBs desde varios dispositivos controladores (Strasser T. S.). El modelo se presenta en la Figura. 7.

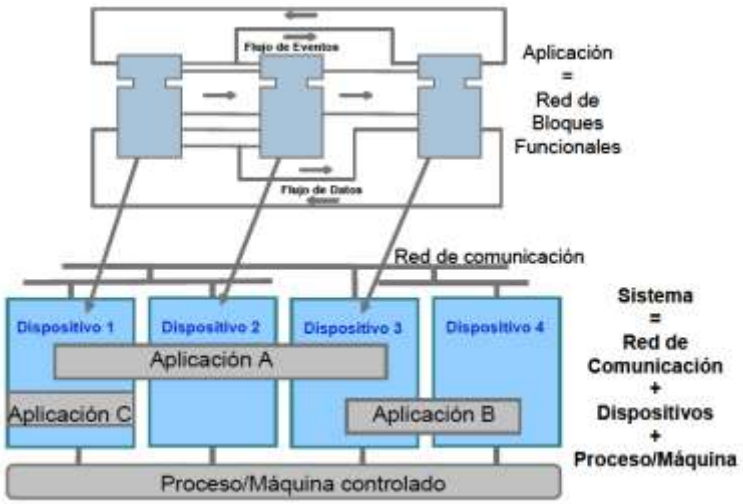



*Figura. 7 Modelo de distribución del estándar IEC-61499*

## Modelo de Gestión

En el modelo de gestión, el estándar IEC 61499 propone un tipo de FB de gestión con lo que se hace posible la configuración de un IPMCS distribuido mediante el uso de las funciones de gestión que adicionalmente pueden ser incluidas en cada dispositivo.

En la Figura.  8, se muestra la representación del conjunto de modelos IEC-61499.

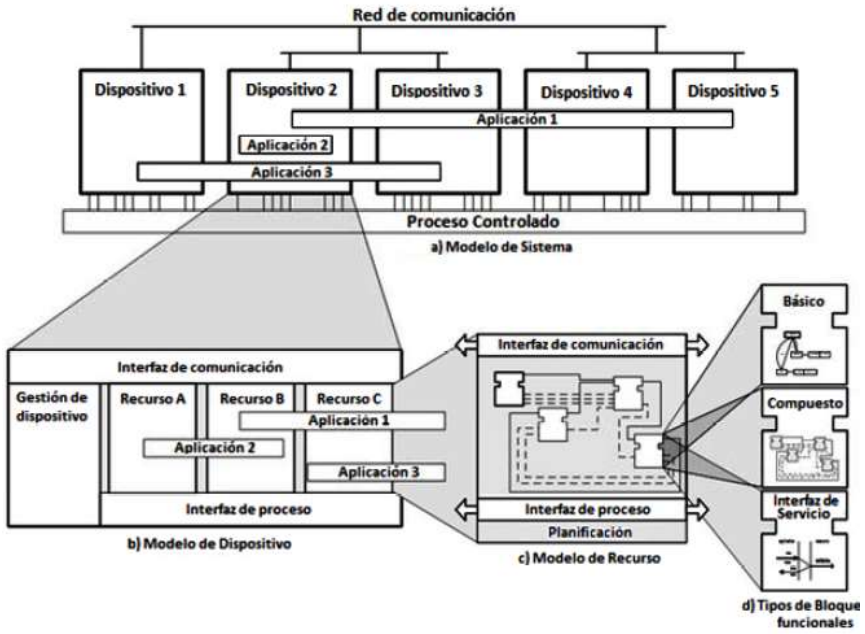



*Figura.  8 Modelo de gestión de un sistema distribuido IEC-61499*

## 2.4. Entornos de Desarrollo y de Ejecución de la Norma IEC-61499

En el apartado 4 de la norma IEC 61499 existen los requisitos para los “Perfiles de Conformidad” cuyo objetivo es garantizar el cumplimiento de las bondades de la norma por dispositivos distribuidos compatibles conjuntamente con herramientas de software; entonces, un sistema distribuido debe cumplir lo siguiente para ser llamado como tal:

- **Portabilidad**. Soporta e interpreta 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 herramientas de software de múltiples proveedores.

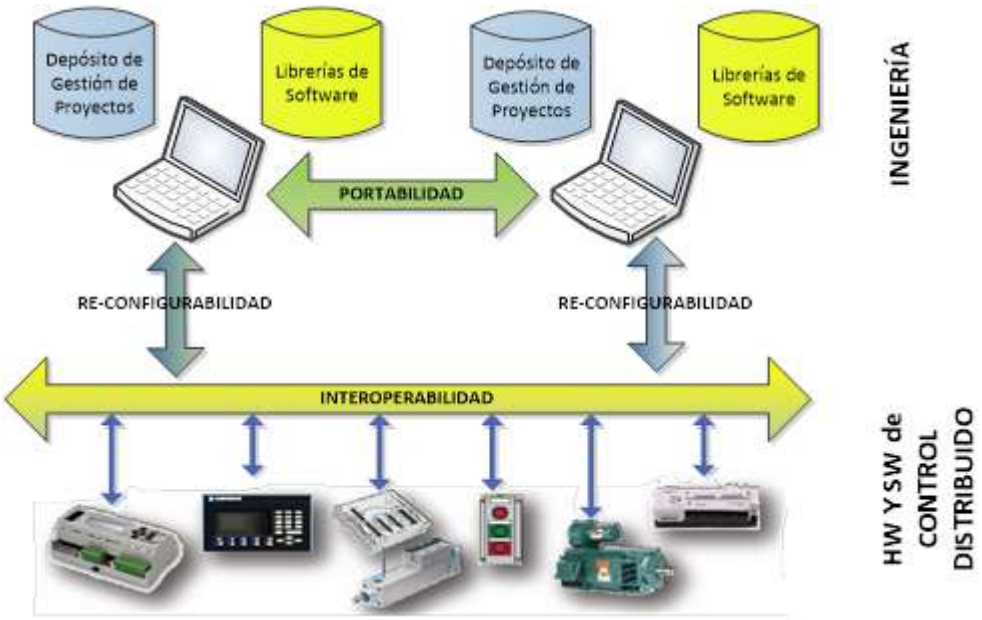



*Figura. 9 La norma IEC-61499 brinda las características de: Interoperabilidad al poder integrar varios dispositivos multimarca en un sistema, Reconfigurabilidad al poder cambiar sobre la marcha la programación de los controladores, y Portabilidad al poder usar la misma programación para configurar varios dispositivos a través del uso de entornos de desarrollo multimarca.*

Cabe mencionar que en los Anexos A y B de la norma se establece como DTDs (Definición del Tipo de Documento) a XML para el intercambio de información entre las herramientas de software que se utilicen. Entonces, los Perfiles de Conformidad deben ser capaces de especificar el nivel en que las herramientas de software compatibles pueden emplear la sintaxis y semántica de los DTDs para establecer la portabilidad de los elementos de software entre las herramientas (Zoitl & Vyatkin, 2009). La Figura. 9 muestra gráficamente todas las características mencionadas.

El entorno de desarrollo para la norma denominado IDE (Integrated Development Environment), consiste de un constructor de interfaz gráfica

(GUI) con un editor de código, un compilador y un depurador que permiten el desarrollo de las aplicaciones de control. Por otro lado, el entorno de ejecución RTE (Runtime Environment) es un estado de máquina virtual que brinda los servicios de software de procesos o programas mientras un ordenador está ejecutándose.

En la Tabla 2 se muestran las principales herramientas de software y sus entornos de ejecución bajo licencia de código abierto (open source) que se utilizan actualmente:

*Tabla 2*

*Sistemas de Desarrollo y Ejecución de IEC-61499*

| ENTORNO DE DESARROLLO (IDE) | ENTORNO DE EJECUCIÓN (RTE) |
|---|---|
| **4DIAC-IDE** | FORTE |
| **FBDK (Function Block Development Kit)** | FBRT (Function Block Run Time) |
| **ISaGRAF Workbench** | ISaGRAF Runtime |
| **NxtSTUDIO** | nxtRT61499F[32] |

A continuación, se describen los entornos de desarrollo más comunes en el ámbito académico y que son, además, código abierto para IEC 61499.

## FBDK / FBRT

### *Entorno de desarrollo: FBDK*

Inicialmente fue desarrollado como una aplicación JAVA con el objetivo de dibujar FBs e incluso redes de FBs. Es considerado como la herramienta original de la norma IEC 61499 que permitió probar el modelo gráfico y el formato de intercambios de archivos XML definidos en el apartado 2 de IEC 61499. Además, se lo creó para guiar y coordinar los esfuerzos de todos los distribuidores e investigadores a nivel mundial para demostrar la “Viabilidad

de un Perfil de Conformidad"; sin embargo, posteriormente se designó dicho Perfil como el apartado 4 de la norma mencionada.

El FBDK, Kit de Desarrollo de Bloques Funcionales es de libre distribución y se ajusta al propósito de educación y software libre (Zoitl, Sunder, & Terzic, 2006). Permite ingeniería centrada en la aplicación, tiene una librería de componentes de software extensa y se puede descargar la aplicación de control a diferentes dispositivos, por estas características es usada para proyectos de investigación.

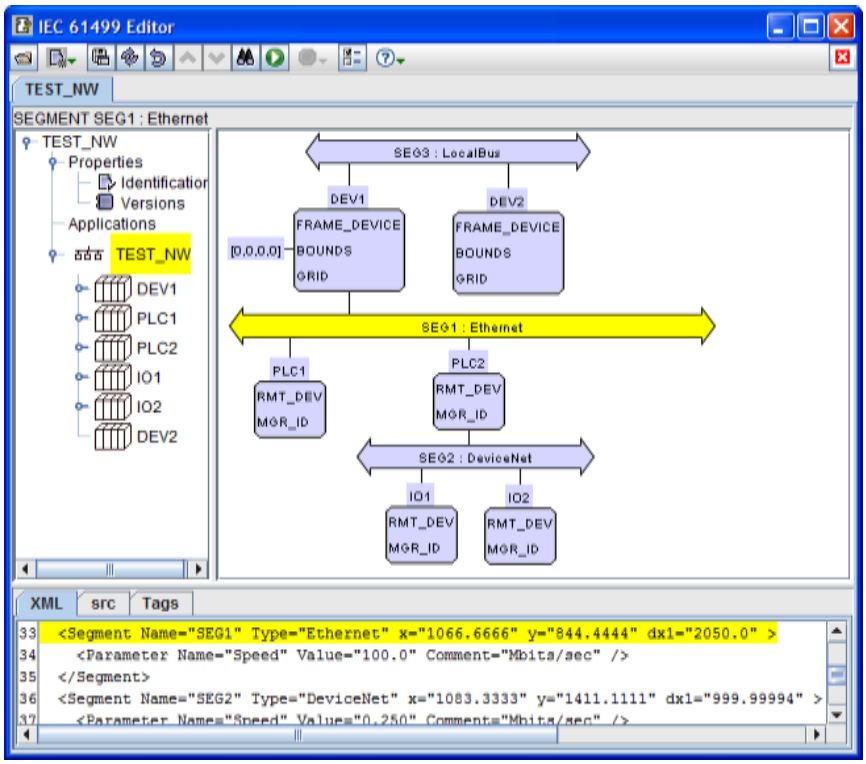



*Figura. 10 Sistema de configuración en FBDK*

El editor es un Entorno de Desarrollo Integrado (IDE) que soporta desarrollo gráfico de Bloques de Función, sistemas y traducción de clases java. Almacena los Bloques de función en el formato basado en lenguaje de marcado extensible XML (eXtensible Markup Language) definido en el apartado 2 del estándar IEC 61499. Se pueden desarrollar y desplegar aplicaciones de sistemas de control distribuido basadas en FBs.

***Entorno de ejecución: FBRT***

Una de las características principales de FBRT es que la implementación de los eventos de salida que disparan la invocación de otros bloques, se la realiza a través de una serie de llamadas a funciones directas dentro de una simple tarea. Este mecanismo detiene la ejecución en la llamada de un bloque a otros FBs a lo largo de la ruta de propagación de eventos hasta que su ejecución haya sido completada.

## 4DIAC-IDE / FORTE

### *Entorno de desarrollo: 4DIAC-IDE*

4DIAC Estructura para la Automatización y Control Industrial Distribuida (Distributed Industrial Automation and Control), es una herramienta de ingeniería de código abierto basada en la plataforma de Eclipse, para automatización distribuida, reconfigurable y software de control. 4DIAC fue creada en el año 2000 por PROFACTOR GmbH & Viena University of Technology (Zoitl S. a., 2011).

El editor es un entorno de desarrollo integrado (IDE) cuyo objetivo es proporcionar herramientas conforme al estándar que permiten establecer un entorno de automatización y control, basado en los objetivos de portabilidad, reconfiguración e interoperabilidad. En esencia, 4DIAC busca:

- Proporcionar una base común para desarrollo e investigación de IEC-61499.
- Suministrar un paquete conteniendo un entorno runtime para diferentes plataformas de control que permitan extender la norma hacia más dispositivos con sistemas embebidos.
- Aportar con ejemplos reales a nivel prototipo para incrementar la aceptación de IEC-61499 y de igual forma el uso de la norma en la industria.

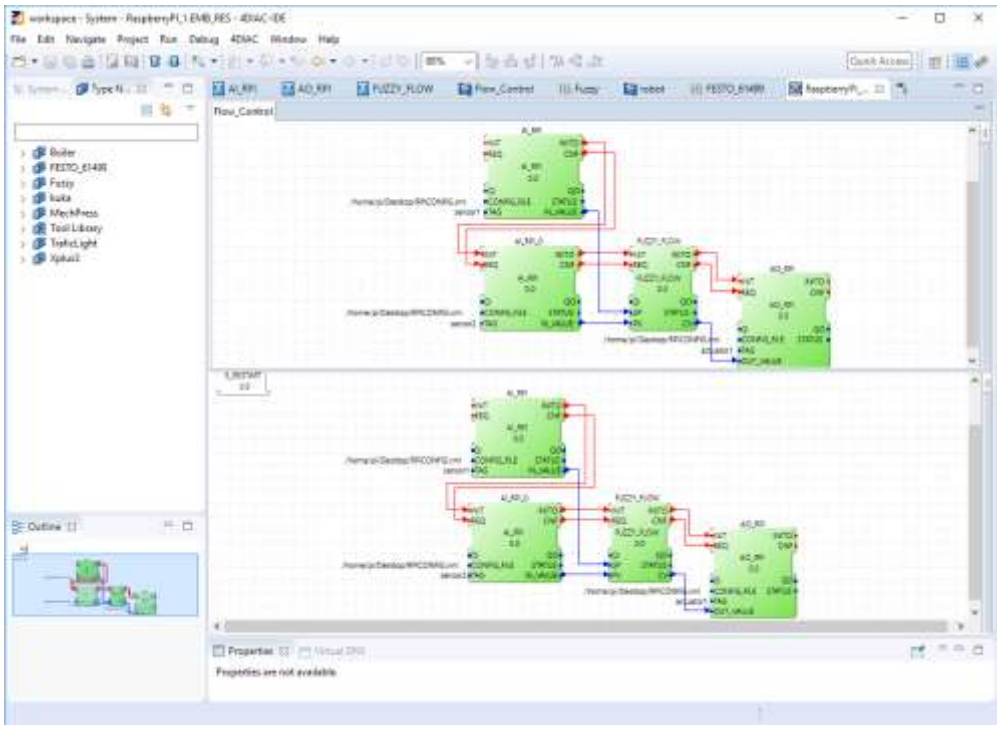


*Figura. 11 Entorno de desarrollo 4DIAC-IDE durante el desarrollo de una aplicación de control*

Las características más relevantes de 4DIAC (Zoitl S. a., 2011) son:

- 4DIAC-IDE es una herramienta IEC-61499 basada en Eclipse.
- Tiene tipos de datos elementales de acuerdo a IEC 61131-3.
- Tiene conexiones de eventos y datos.
- Configuración de comandos (crear, escribir, iniciar de acuerdo con IEC-61499).
- Tiene FBs de comunicación (Client/Server, Publish/Subscribe) para Ethernet.
- Puede ejecutar bloques funcionales de tipo FB básicos, FB compuestos, FB interfaces de servicio SIFBs y adaptadores.
- Se ejecuta bajo plataforma Windows, Linux y Solaris.

El IDE de 4DIAC despunta gracias a sus últimas características que lo vuelven la herramienta de software más utilizada; en su última versión se cuenta con la capacidad de establecer el valor de las variables de manera

remota además de permitir la depuración y prueba de aplicaciones de control distribuido en línea.

### *Entorno de Ejecución: FORTE*

El entorno de ejecución se denomina 4DIAC-RTE (FORTE). FORTE es una pequeña implementación portable de un entorno runtime conforme a IEC-61499 enfocado a pequeños dispositivos de control con sistemas embebidos (16/32 Bit); es portable para múltiples plataformas debido a que está implementado en C++ (Zoitl S. a., 2011). Los mecanismos de ejecución en FORTE permiten la ejecución en tiempo real de configuraciones de control IEC-61499 desencadenadas por eventos externos, donde las diferentes partes de la configuración pueden cumplir diferentes limitaciones en tiempo real y la ejecución de los procesos de baja prioridad no perturban la ejecución de los procesos de mayor prioridad.

En el entorno de ejecución FORTE, se utiliza un disparador de eventos para la planificación de FBs. Es decir, todos los eventos de entrada se los entrega a los FBs destino en orden FIFO (primero entra-primero sale). El disparador de eventos desacopla la ejecución de envío de eventos del bloque receptor, de ese modo crea el periodo de bloqueo de un FB independiente de la topología de red. (Zoitl S. a., 2011).

FORTE se mantiene cambiando según la optimización de ejecución e implementación de interfaz de comunicaciones. Este entorno de ejecución fue desarrollado para funcionar independientemente de la plataforma que se emplea, esto desemboca en una mayor facilidad de uso en diversos tipos de hardware y plataformas con sistemas embebidos. La versión actual de FORTE es soportada por los siguientes sistemas: Windows(Win32), eCos, POSIX, Lego Mindstorms, NXT controller, etc.

## 2.5. Estándar ISA-88

El estándar ISA-88 proporciona una guía para el diseño de sistemas de control por lotes; define la terminología básica necesaria para comprenderla y un grupo de modelos para usar. El objetivo principal de esta norma es

diferenciar toda la información relacionada con el producto, de la información relacionada con el equipo y la maquinaria del proceso. De esta forma, el punto de partida del diseño es el proceso, y los resultados consisten en un grupo de recetas que especifican las acciones requeridas y su correcto orden de ejecución.

Este tipo de recetas se elaboran siguiendo un esquema jerárquico, donde las fases son los elementos básicos en el diseño. Cada una de estas fases describe una función básica de la planta a controlar, y al agrupar un conjunto de fases y ejecutarlas en un orden específico (lineal o paralelo), se pueden originar operaciones. El estándar ISA 88 también define una jerarquía de equipos: un objeto de empresa que contiene uno o más objetos de sitio, cada uno de los cuales contiene uno o más objetos de área. Cada área estará constituida por uno o más centros de trabajo. En ISA 88, todos los centros de trabajo son celdas de proceso (es decir, equipos utilizados en el procesamiento por lotes).

El estándar ISA-88 introduce un lenguaje de diseño llamado Procedure Function Chart (PFC), que hace posible describir gráficamente la organización de las operaciones. Una desventaja de este lenguaje es que al ser una aproximación semiformal, no permite un análisis más grande ni garantiza que, una vez que se active un actuador relacionado con más de una fase, solo se active la fase requerida. Se puede hacer una aproximación en la arquitectura de activación para resolver estos problemas gracias a las grandes similitudes que tienen los PFC con el lenguaje Secuencial Functional Chart (SFC) de IEC 61131.

# CAPÍTULO III

## 3. CASO DE ESTUDIO

Como caso de estudio para el proyecto de investigación planteado se ha seleccionado el sistema de maquetas modulares de procesos FESTO FMS-200, cuyo funcionamiento busca replicar procesos industriales implementados globalmente, permitiendo de esta manera tener una mejor perspectiva del desempeño del sistema de control a desarrollar bajo la norma IEC-61499 en ambientes industriales. El sistema FESTO FMS-200 fue diseñado bajo el concepto de modularidad, poniendo a disposición un amplio grupo de estaciones o módulos individuales enfocados a un proceso distinto, los cuales pueden ser agrupados de varias maneras y generar múltiples escenarios de producción autónoma.

Dentro de todos los posibles procesos que se pueden simular en el sistema, se trabaja con los procesos de distribución, selección, almacenamiento y clasificación. En cada una de las estaciones seleccionadas, lo actuadores y sensores trabajan con señales discretas bajo el gobierno de PLCs y algoritmos de control desarrollados en IEC-61131. Solo en el caso de la estación de almacenamiento el autómata es de la marca FESTO mientras que en el resto se trata de PLCs SIEMENS.

El funcionamiento del sistema se basa en una ejecución serie de cada uno de estos procesos para completar un ciclo del lazo de trabajo global del mismo y simular de esta manera un proceso discreto por lotes. Un ciclo de trabajo comprende: la distribución del material de trabajo, separación de las unidades defectuosas del proceso mediante una selección en base a la altura de cada pieza, transporte de los materiales sin defectos al proceso de almacenamiento, finalizando con la clasificación de los diferentes materiales en base a sus características de color como se muestra representado en Figura. 12.

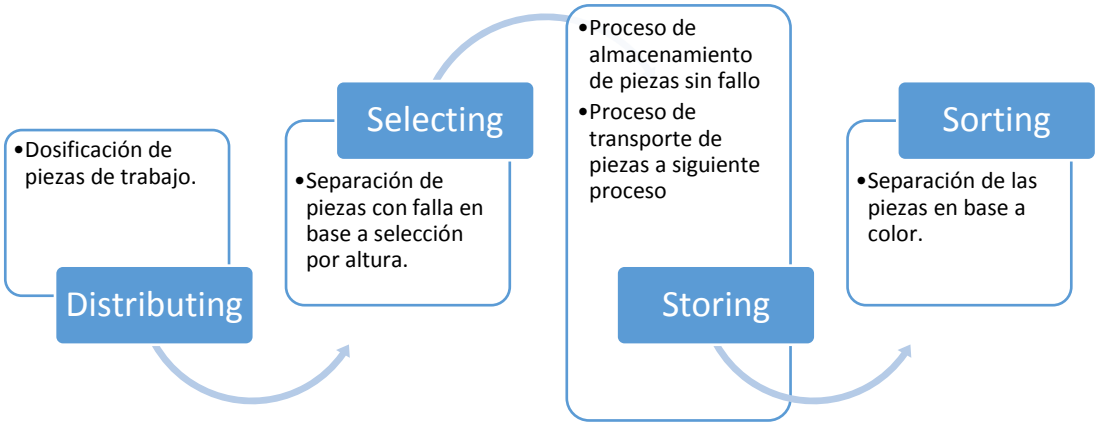



*Figura. 12 Ciclo de trabajo del sistema FESTO seleccionado como caso de estudio*

Para comprobar el desempeño del sistema de control distribuido a desarrollar y aportar al escalamiento de la norma IEC-61499, el siguiente trabajo de investigación se basa en el uso de las tarjetas controladoras Raspberry PI 2, Beaglebone Black y Raspberry PI 3 como propuesta de hardware a bajo costo, para reemplazar el elemento de control principal de las estaciones de selección, almacenamiento y clasificación respectivamente. Estas tarjetas fueron escogidas gracias a que su capacidad inherente de poseer un software embebido, les permite ser dispositivos de control plenamente compatibles con la arquitectura IEC-61499 sin incidir en elevados costos al momento de adquirirlas.

## 3.1. Raspberry Pi

La tarjeta de desarrollo Raspberry Pi es originaria del Reino Unido, más específicamente, sus primeros pasos los dio en el Laboratorio de Computación de la Universidad de Cambridge (Denis, 2016); su objetivo en forma inicial fue proveer de un dispositivo de bajo costo a niños con el propósito de impulsar el estudio de la computación, así como establecer las bases de la programación con que se maneja el mundo de hoy en día.

Varios científicos ingleses entre los que figuran Eben Upton, Rob Mulling, Jack Lang y Alan Mycroft fueron quienes comprendieron, basados en el análisis de los sílabos de estudio de las escuelas y colegios entre los años 2000 y 2005 que, el estudio de la computación iba más allá del entendimiento de sus características, estaba más bien enfocado en las aplicaciones que se obtenían antes que en su análisis de composición para la puesta en marcha.

La primera tarjeta Raspberry Pi que vio la luz fue en febrero del año 2012, tiempo para el cual ya se había pasado seis años bajo constantes estudios y aprobaciones; sin embargo, en el año 2014 se lanzó la segunda versión de la tarjeta Raspberry Pi manteniendo la característica de su lanzamiento al mercado en dos modelos, el Modelo A y Modelo B. El primero de ellos tenía la particularidad de poseer menos potencia que el Modelo B mostrado en la Figura.  13, es decir, no contenía soporte para conectividad Ethernet y las características de memoria RAM y velocidad de procesamiento eran considerablemente menores; por otro lado, el Modelo B de Raspberry Pi tuvo una aceptación amplia que superó la barrera del mercado al cual fue dedicado, abriéndose paso hacia toda la comunidad académica e incluso de investigación por sus bondades y costo accesible.


*Figura.  13 Raspberry PI 2 Modelo B*

## Hardware de Raspberry Pi

En el año 2014 la Raspberry PI Foundation lanza al mercado su tarjeta de desarrollo Raspberry PI 2 Modelo B, en la cual se cambia el procesador utilizado en versiones anteriores por el nuevo modelo BCM2836. Este procesador deja atrás a sus predecesores de un solo núcleo a 700MHz y le brinda al dispositivo la capacidad de trabajar con cuatro núcleos a 900MHz. De igual manera, éste nuevo modelo duplica la cantidad de memoria RAM de sus predecesores pasando de 512MB a 1GB, pero en realidad es un poco menor debido a que ésta memoria se encuentra compartida con la gráfico. Un cambio adicional es la supresión de la conexión RCA pero se incluye un puerto GPIO de 40 pines y se mantienen los cuatros puertos USB.

A partir del año 2016, la Raspberry PI Foundation presenta al público su trajerta Raspberry PI 3 Modelo B en la cual se renueva el procesador, el cual sigue siendo de la casa comercial Broadcom pero pasa de 900MHz a 1.20GHz. En general mantiene las características de su predecesora,como punto novedoso incluye compatibilidad Wi-Fi y Bluetooth 4.1 opia estructura.

*Tabla 3*

*Comparación de características de las tarjeras Raspberry PI 2 Modelo B y Raspberry PI 3 Modelo B*

| | **Raspberry PI 2 Modelo B** | **Raspberry PI 3 Modelo B** |
|---|---|---|
| *CPU* | 900 MHz quad-core ARM Cortex A7 | 1.2GHz 64-bit quad-core ARM Cortex A8 |
| *Juego de instrucciones* | RISC de 32 bits | |
| *GPU* | Broadcom VideoCore IV a 250MHz soporta OpenGL ES 2.0 (24 GFLOPS) Mpeg-2 y VC-1 (con licencia). | |

Continúa

| | | |
|---|---|---|
| | Además contiene un codificador/decodificador a 1080p30 H.264/MPEG-4 AVC. | |
| *Memoria (SDRAM)* | 1GB de memoria SDRAM del tipo LPDDR2 compartida entre la CPU y GPU acorde a las necesidades de cada una. | |
| *Puertos USB 2.0* | 4 | |
| *Entradas de vídeo* | Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF (Raspberry PI Fundation) | |
| *Salidas de vídeo* | Conector RCA (PAL y NTSC), Conector HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD. | |
| *Salidas de audio* | Conector hembra de 3.5 mm, HDMI | |
| *Almacenamiento integrado* | MicroSD | |
| *Conectividad de red* | 10/100 Ethernet (RJ-45) via hub USB | 10/100 Ethernet (RJ-45) vía hub USB, Wifi 802.11n, Bluetooth 4.1 |
| *Periféricos de bajo nivel* | 17 x GPIO y un bus HAT ID | |
| *Consumo energético* | 800 mA, (4.0 W) | |

Continúa

| | | |
|---|---|---|
| *Fuente de alimentación* | 5 V vía Micro USB o a través de las entradas específicas GPIO | |
| *Dimensiones* | 85.60mm × 53.98mm | |
| *Sistemas operativos soportados* | GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, SUSE Linux Enterprise Server for ARM.<br><br>RISC OS. | |
| *Costo* | 35$ - 40$ (Valores tomados de Amazon.com) | 40$ - 45$ (Valores tomados de Amazon.com) |

## Software de Raspberry Pi

Para que la Raspberry Pi pueda entrar en funcionamiento, es esencial instalar un Sistema Operativo (OS = Operating System) sobre la tarjeta microSD. El OS es el director general de todos los recursos que contiene la tarjeta, tanto así que es el encargado de designar las tareas que debe cumplir cada uno de ellos, ya sea por aplicaciones ejecutadas o por servicios propios del sistema (François, 2016). Cuando el software requiere de algún recurso de hardware, es precisamente el OS quien atiende la necesidad e inmediatamente asigna la tarea. De manera implícita se encuentra el manejo del intercambio de información en la red, así como el gobierno de los dispositivos de almacenamiento conectados a la tarjeta.

Raspberry Pi admite varios sistemas operativos entre los que figuran aquellos que son oficiales y otros no oficiales debido a que se encuentran en sus versiones Beta o a su vez contienen errores que hasta el momento de su lanzamiento no pudieron ser corregidos y por lo tanto dejan de ser oficiales.

En la Tabla 4 se presentan las variedades de software que pueden ser instalados en la Raspberry Pi.

*Tabla 4*

*Software disponible para Raspberry Pi*

| Oficiales | No Oficiales |
|---|---|
| Raspbian Debian Jessie, Raspbian Lite Debian Jessie, Fedora, Windows 10 IoT Core, OSMC OpenElec, Pi Music Box, RetroPie, RecalBox, Ubuntu Snappy Core, Ubuntu MATE, Risc OS. | Arch Linux, HypriotOS, Slackware, OpenSUSE, VoidLinux, Kali, Minibian, Minibian-wifi, Pidora, Pignus, PiBang, OpenMediaVault, MotionEyeOS, Kano OS, DietPi, LinuTOP, Q4OS, Occidentalis, Moebius. |

Es importante mencionar que la fundación de Raspberry Pi lanzó su tarjeta con el menor costo posible ($45 dólares de los Estados Unidos aproximadamente) por lo que es natural pensar que los usuarios de dicha tarjeta se involucren en software libre para el desarrollo de sus aplicaciones; a pesar del soporte para OS con licencia como Windows 10 IoT Core, Raspberry Pi propone el uso de software libre, así como proporciona el soporte para el mismo con mucho más énfasis antes que para aquellos que requieren licencia. En la página https://www.raspberrypi.org/downloads/ se puede encontrar con exactitud el software más recomendado por la Raspberry Pi Foundation.

Este es uno de los motivos por los que en el desarrollo del proyecto de investigación se decidió utilizar el software Raspbian Debian Jessie en su versión de septiembre de 2016 la cual es la última lanzada (a octubre de 2017). Sin embargo, existen más razones que se explicarán a continuación, por las que el uso del software más recomendado por Raspberry Pi fue el más apto para la realización de este proyecto.

### *Raspbian Debian Jessie*

El SO Raspbian Debian Jessie es un sistema operativo que está basado en Debian Linux, las diferentes versiones de Debian han sido nombradas con los diferentes personajes presentes en las películas de “Toy Story”. Todas las versiones actuales de Raspbian están basadas en el Debian Wheezy (Raspberry, 2015).

Uno de los puntos amigables que posee esta versión de Debian está basado en el arranque, puesto que ya no inicia el usuario en la pantalla de comandos para arrancar el sistema, esta vez y siguiendo la tendencia de los ordenadores modernos, se arranca directamente desde el GUI (Graphic User Interface = Interfaz Gráfica de usuario) de escritorio, lo que le permite ser mucho más amigable con el usuario final, no obstante y si el programador lo considera necesario, puede cambiar a la pantalla de comandos para el arranque con unos simples pasos desde la ventana de configuración de la tarjeta Raspberry Pi.

La estabilidad y compatibilidad con las librerías lanzadas para soportar la norma IEC 61499 motivaron a los desarrolladores de este proyecto a decidirse por la última versión de Raspbian disponible debido a que luego de realizar varias pruebas con versiones anteriores que suponían una estabilidad mejor para el manejo de puertos GPIO de la tarjeta, colapsaron cuando se ejecutaron aplicaciones que utilizaban librerías como Xerces que maneja la norma mencionada. Adicionalmente, el manejo de Raspbian Jessie, proporcionó un desempeño más limpio al momento de manejar las aplicaciones considerando que éste SO administraba de mejor manera los recursos de hardware de la tarjeta, evitando que la actualización de la pantalla se ejecute en intervalos demasiado extendidos o permanezca congelada durante varios segundos.

### *Instalación del software en la tarjeta Raspberry Pi*

La instalación del software en la tarjeta se realiza mediante el programa Win32 Disk Imager que permite grabar los archivos de la imagen ISO del

software en la tarjeta microSD, para que pueda ser reconocida como el disco de arranque de la Raspberry Pi.

A continuación, se detalla la forma en que se bootea la microSD con el software Raspbian Jessie:

1. Se instala y ejecuta el programa Win32 Disk Imager obteniendo la pantalla como se muestra en la Figura. 14.

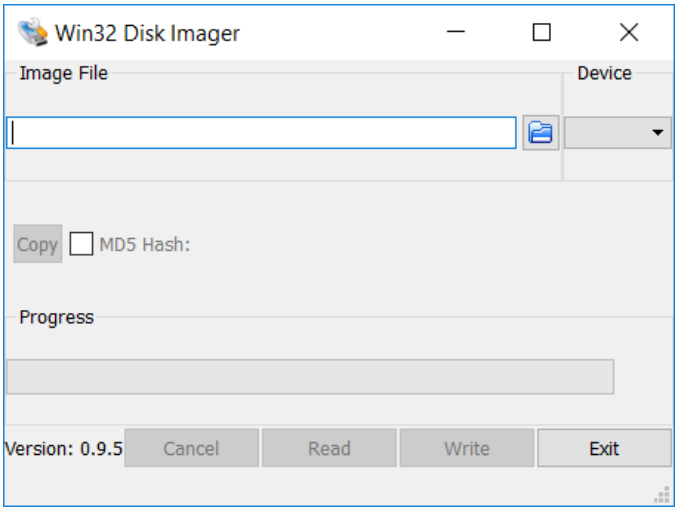



*Figura. 14 Pantalla inicial de Win32 Disk Imager*

2. Se elige el archivo con la imagen ISO de Raspbian Jessie desde la ubicación en donde se la contenga conjuntamente con el dispositivo en el que se va a bootear dicho OS como se muestra en la Figura. 15. Finalmente, se presiona el botón Write para escribir sobre la microSD.

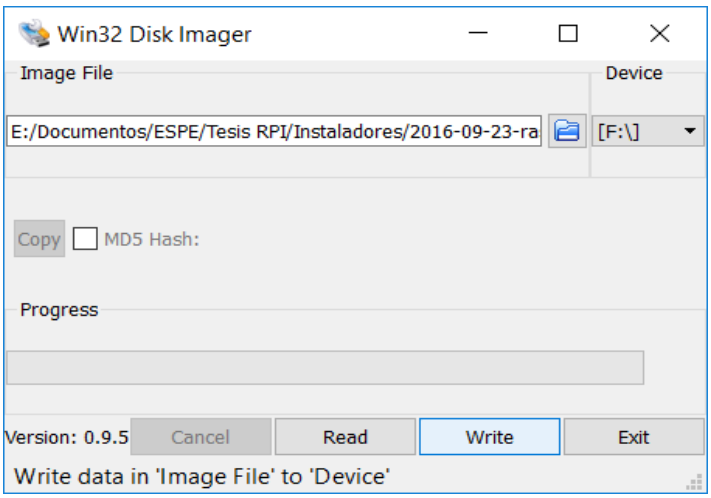



*Figura. 15 Selección de imagen ISO para boot de tarjeta microSD*

### *Configuración de IP estática en la Raspberry Pi*

Para poder iniciar la configuración de la tarjeta para las aplicaciones posteriores, es necesario fijar una IP estática en la tarjeta para poder conseguir la visualización de la GUI de escritorio que proporciona el OS instalado, en los siguientes pasos se detalla el procedimiento para establecer la IP estática sobre la tarjeta.

1. Conectar la tarjeta mediante un cable Ethernet a un módem de internet y encenderla.

   Instalar y ejecutar el programa Wireless Network Watcher que se puede descargar de la página https://wireless-network-watcher.softonic.com/ para obtener una interfaz como la que presenta la Figura. 16.


*Figura. 16 Pantalla principal del programa de detección de IPs de dispositivos conectados a la red Wireless Network Watcher*

Buscar todas las IP disponibles dentro de la Red detectada hasta identificar la IP asignada mediante DHCP a la tarjeta Raspberry Pi como indica la figura Figura. 17.


*Figura. 17 Detección automática de una nueva dirección IP tras la conexión de la tarjeta Raspberry PI en el modem*

2. Una vez identificada la IP nos dirigimos a la aplicación PuTTY que se puede descargar desde su página oficial https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html en donde ingresamos la IP de la Raspberry Pi y seleccionamos la opción *Open* para abrir la ventana de comandos de la tarjeta desde donde es posible realizar la configuración de la IP estática.

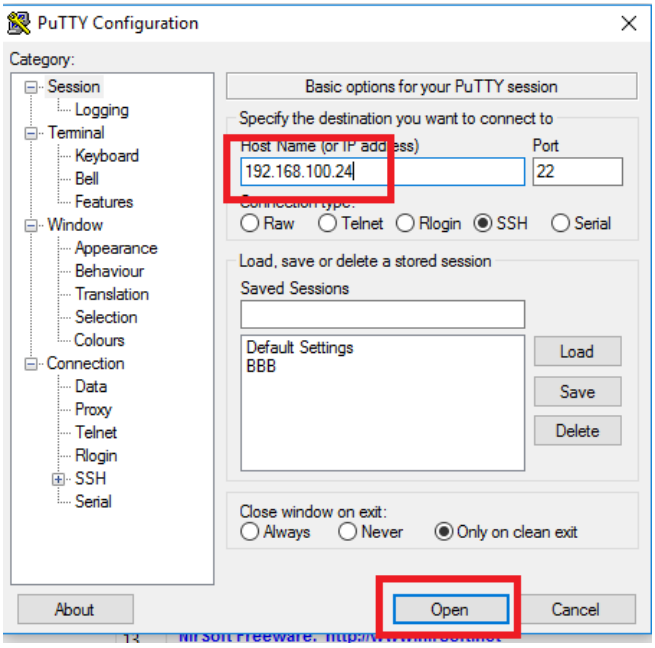



*Figura. 18 Pantalla principal al ejecutar el programa PuTTY, con la dirección IP ingresada para realizar una conexión.*

3. Seguidamente se realiza el login a la tarjeta mediante el usuario *pi* y la contraseña *raspberry* como se muestra en la Figura. 19.


*Figura. 19 Ventana de terminal para conexión SSH con tarjeta Raspberry PI*

4. Ingresar con permisos de superusuario a la dirección en donde se encuentra el archivo de configuración de red mediante el código *sudo nano /etc/dhcpcd.conf,* y realizar necesarios para obtener una configuración parecida a la de la Figura. 20 y de esta manera haber configurado una dirección IP estática en el dispositivo.

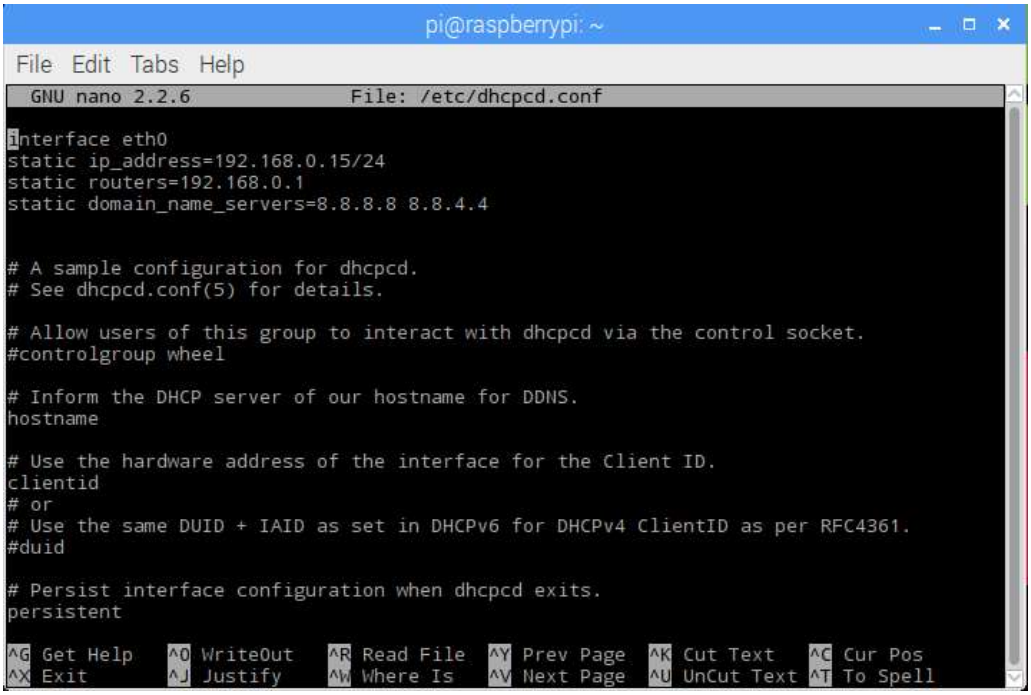



*Figura. 20 Archivo de configuración de red de la tarjeta Raspberry PI tras haber ingresado los datos necesarios para el seteo de una dirección IP estática.*

Los comandos vistos en la Figura. 20, hacen referencia a la IP con la que se fija a la tarjeta, así como la clase de submáscara que posee, estableciendo de tal forma una submáscara tipo C para el caso de este proyecto de investigación. Con los pasos vistos previamente se fija la IP de la tarjeta para dar lugar a la visualización de la GUI de escritorio.

### *Escritorio de Raspberry Pi desde Windows 10*

Para poder visualizar la GUI de escritorio de la tarjeta Raspberry Pi es necesario conectar mediante un cable Ethernet desde la computadora hacia la tarjeta previa la configuración de una IP estática en la tarjeta (explicado en el apartado inmediatamente superior). A continuación, se explica detalladamente cómo se accede al escritorio remoto de la Raspberry Pi desde Windows 10:

1. Abrir la aplicación de escritorio remoto de Windows como se muestra en la Figura. 21 e ingresar la dirección IP de la Raspberry Pi.

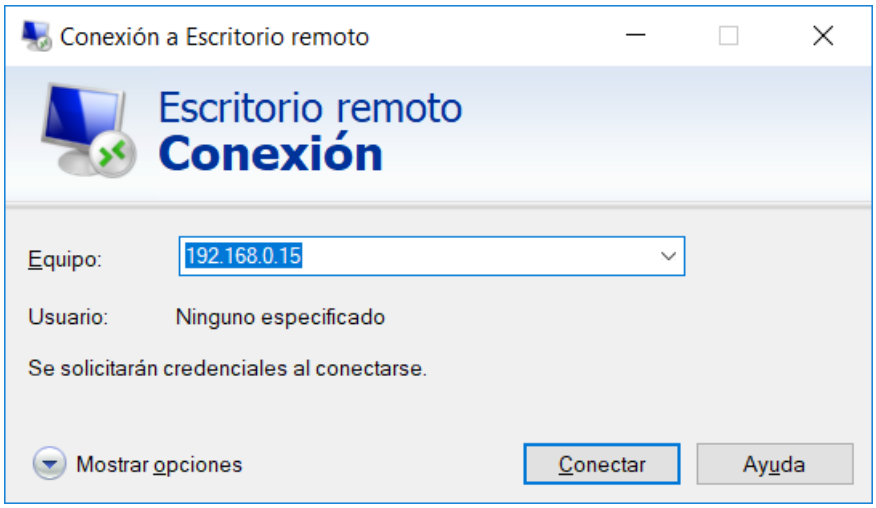



*Figura. 21 Ventana de acceso a escritorio remoto en Windows 10*

2. Una vez que se presiona Conectar, aparece la pantalla de login idéntica a la pantalla de la Figura. 22; en este punto se debe ingresar el mismo usuario y contraseña explicados anteriormente.

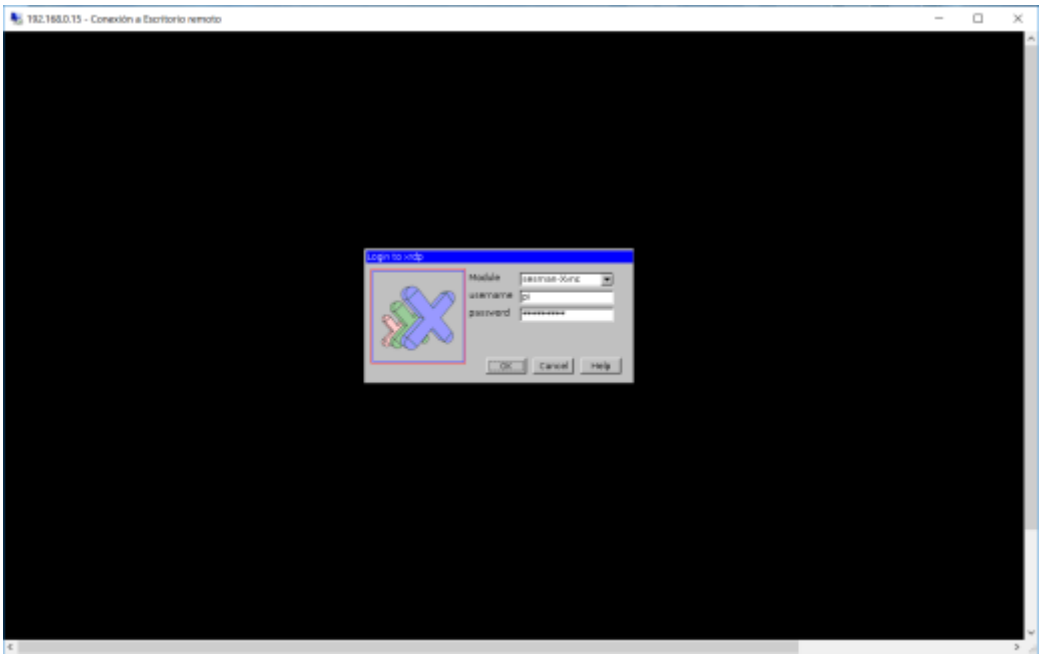


*Figura. 22 Ventana de ingreso de usuario para conexión remota desde Windows 10*

Finalmente, obtenemos la GUI de escritorio de la Raspberry Pi mediante el uso del escritorio remoto de Windows 10 con lo que se facilita el manejo de archivos dentro de la tarjeta.

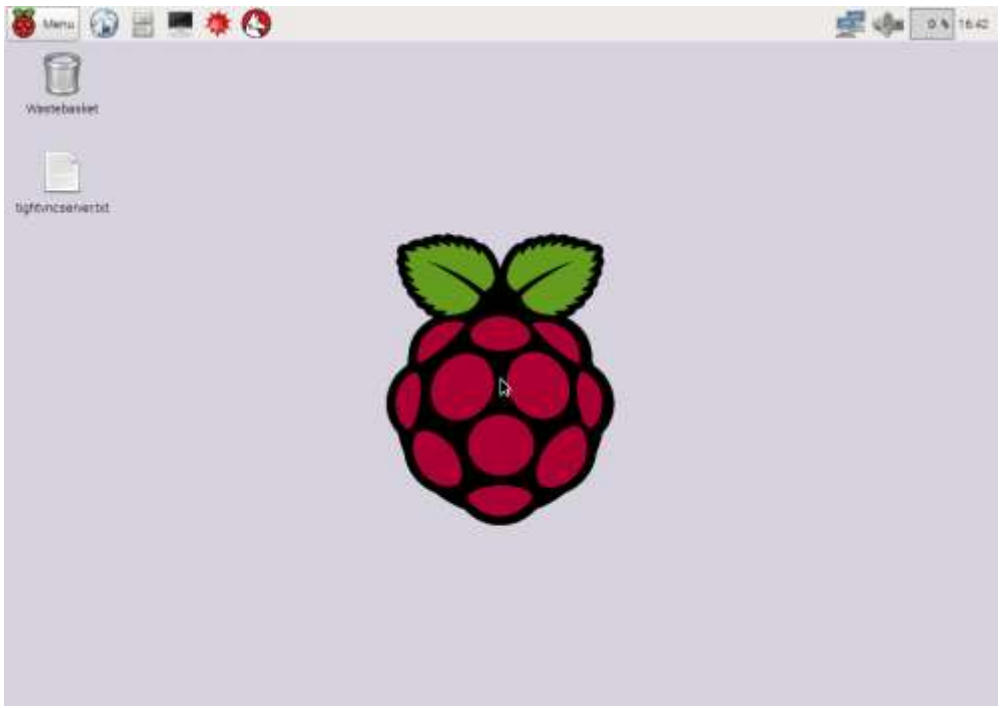


*Figura. 23 Escritorio de la tarjeta Raspberry PI vista a través del escritorio remoto de Windows 10*

### *Transferencia de archivos a la tarjeta Raspberry Pi*

La transferencia de archivos desde la PC con Windows 10 hacia la tarjeta Raspberry Pi se la puede realizar mediante el protocolo SSH gracias al programa WinSCP, el cual se puede descargar en dos versiones de distribución libre: portable o de escritorio desde su página oficial https://winscp.net/eng/download.php#download2.

Generalmente la opción más utilizada es la versión portable en donde una vez abierto se obtiene una pantalla como la que se presenta en la Figura. 24 en donde se deben ingresar los datos de dirección IP, nombre de usuario y la contraseña (pi & *raspberry en el caso de la tarjeta Raspberry)* para poder logear y empezar la transferencia de datos directa entre el ordenador y la tarjeta.

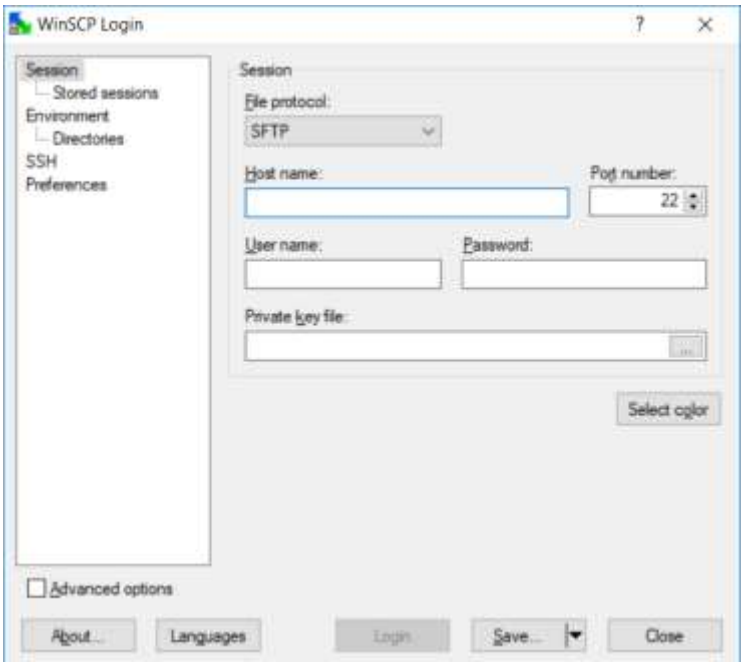



*Figura. 24 Ventana de inicio del programa WinSCP*

Una vez que se ha generado un correcto login se obtiene una pantalla como la que se muestra en la Figura. 25 en donde en el lado izquierdo se puede navegar entre los archivos de la PC y en el lado derecho se encuentran todos los archivos de la tarjeta.

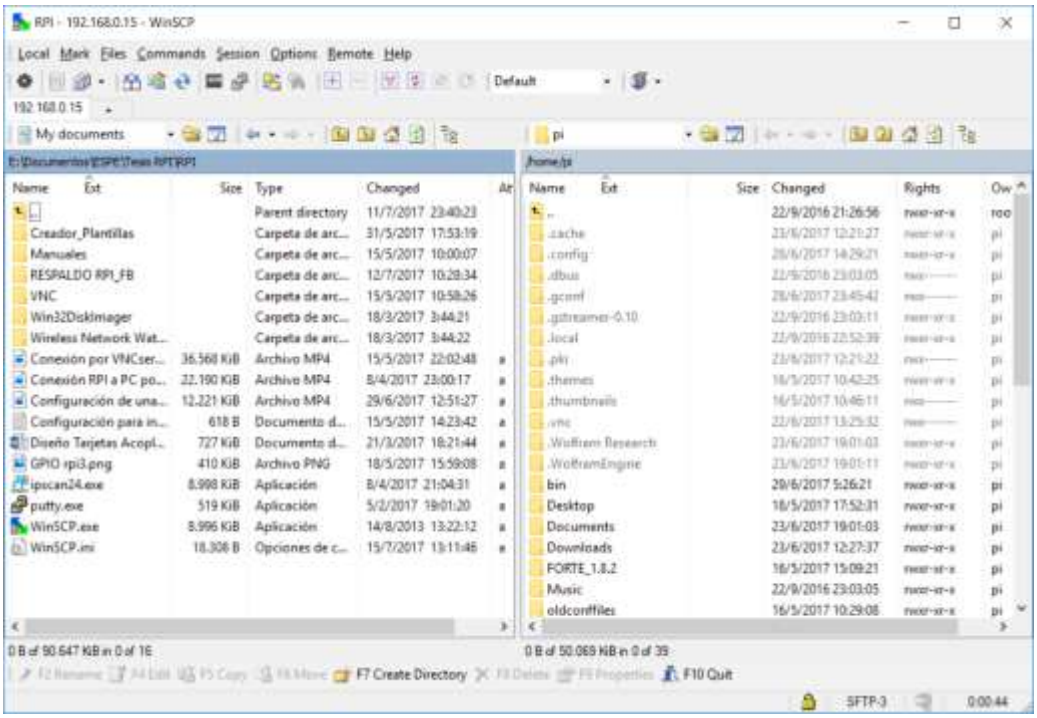


*Figura. 25 Sesión de transferencia correctamente iniciada entre la computadora y la tarjeta Raspberry PI*

Para copiar o mover un archivo desde la PC a la tarjeta, se requiere buscar y seleccionar el archivo en el lado de la PC (izquierdo); y, en el lado de la tarjeta (derecho) dirigirse a la ubicación en donde se desea pegar. Una vez hecho esto, seleccionar la opción requerida (F5 Copy o F6 Move) y automáticamente aparecerá una pantalla como la Figura. 26 en donde se debe confirmar la ubicación en donde va a ser transferido el archivo seleccionado para posteriormente reconfirmar la acción.

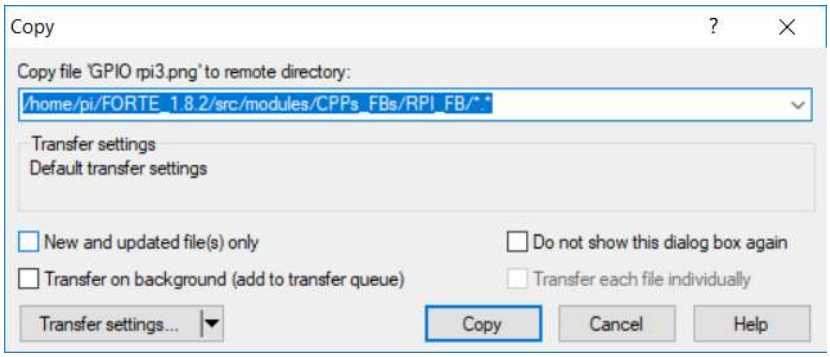



*Figura. 26 Ventana de asistente de copia de archivos de WinSPC*

El proceso para la transferencia de archivos desde la Raspberry hacia la PC es idéntico pero esta vez se transfiere desde el lado derecho (tarjeta) hacia el izquierdo (PC).

## 3.2. BeagleBone Black

La tarjeta BeagleBone Black mostrada en la Figura. 27 al igual que sus antecesoras BeagleBoard y BeagleBoard-xM, fueron diseñadas por el Ingeniero Gerald Coley perteneciente al grupo de ingenieros desarrolladores de Texas Instruments (Barret & Kridner, 2016). Su primer lanzamiento se produjo en abril de 2013 por la organización de BeagleBone.org compuesta por varios desarrolladores establecidos en el año 2008. La idea de lanzar al mercado una tarjeta de este tipo fue principalmente la necesidad de brindar a los usuarios una computadora totalmente libre para desarrollar cuantas aplicaciones se vengan en mente, dicho de otra forma, se crea la BeagleBone con el objetivo de aumentar el desarrollo de tecnología mediante la simplificación del ordenador sin poner límites de uso a los desarrolladores.

El ingeniero Gerald Coley bautiza a su creación con el nombre de Beagle debido a su afinidad muy particular con los caninos de la raza Beagle sin mencionar que su mascota “Jake” pertenecía a dicha raza. Sin embargo, años después se desarrolló un acrónimo para Beagle dando un realce más fuerte para el entendimiento del propósito general que posee la tarjeta. El acrónimo es el siguiente:

- **B**ring your own peripherals
- **E**ntry-level costs
- **A**RM Cortex-A8 superscalar processor
- **G**raphics accelerated
- **L**inux and open source community
- **E**nvironment for innovators

*Figura. 27 Tarjeta Beaglebone Black REV C*

## Hardware de la BeagleBone Black

Con dimensiones de 76.2 x 76.2 x 16 [mm] y un peso de 36.68 g la BeagleBone Black pertenece al grupo de ordenadores sumamente pequeños y muy potentes (ElectroniLab, 2016). Contiene la misma tecnología de procesamiento que la Raspberry, es decir, el procesador es de tipo SoC. Además, posee periféricos y conectividad Ethernet para la comunicación entre dispositivos o a su vez para el acceso a la web. Contiene LEDs indicadores que son precisos para el conocimiento básico del trabajo de la tarjeta; la alimentación es por medio de un cable mini-USB o también por el Jack DC. En la Tabla 5 se presentan más a fondo las características de hardware de la tarjeta BeagleBone Black.

*Tabla 5*

*Características de la tarjeta Beaglebone Black Rev C*

| ***Característica*** | **Descripción** |
|---|---|
| *CPU* | Sitara AM3359AZCZ100 a 1 GHz |
| *GPU* | SGX530 3D, 20M Polygons/S |

| | |
|---|---|
| *SDRAM* | 512 MB DDR3L 400Mhz |
| *Flash* | 2GB,8bit Embedded MMC |
| *Almacenamiento integrado* | MicroSD |
| *Conectividad* | Puerto Ethernet 10/100 Mbps RJ45 |
| *GPIO pins* | 2 conectores de 46 pines a 3.3V TTL |
| *Audio* | HDMI, Stereo |
| *Video* | 16b HDMI, 1280×1024 (MAX)1024×768, 1280×720, 1440x900w/Soporte EDID |
| *Costo* | 62,99$ – 64,75$ (*Valores obtenidos de Amazon.com*) |

## Software de BeagleBone Black

Típicamente el software que se utiliza dentro de la BeagleBone Black es Linux. Dependiendo de la aplicación en la que se va a utilizar, se puede instalar una distribución que contenga mayor afinidad a la aplicación para la que se desea utilizar la tarjeta. Lo importante es que todas y cada una de las distribuciones de Linux están basadas en la misma kernel, por lo que las diferencias que brindan están basadas en ciertas configuraciones, así como la experiencia de usuario que proporciona la distribución usada.

Debian, Ångström, Ubuntu y Arch Linux son las versiones de Linux más utilizadas dentro de esta tarjeta. La estabilidad, experiencia de usuario y código abierto son los pilares que incentivaron a la comunidad investigadora para decidir utilizarlas con más frecuencia. No obstante, Debian es la

distribución de Linux con mayor soporte técnico a través de foros e información en la web por lo que fue elegida por los desarrolladores de este proyecto para arrancar con las configuraciones necesarias en favor del cumplimiento de los objetivos del trabajo de investigación.

BeagleBone Black es una tarjeta que viene pre instalado el software Debian por lo que es una tarjeta considerada Plug&Play. El único requisito que se tiene es instalar los drivers en la computadora desde la que se va a administrar la tarjeta para lo que es suficiente conectarla y Windows buscará automáticamente los drivers que permiten el manejo total de la tarjeta. Una vez instalados los drivers se tendrán nuevos dispositivos disponibles en la PC como por ejemplo los siguientes:

- Acceso a la partición FAT de la BeagleBone Black como si fuera una memoria USB.
- Acceso serial a través del driver *Gadget Serial*.
- Un Gadget para el internet a través de USB denominado *Linux USB Ethernet/RNDIS.* (RNDIS = Remote Network Driver Interface Specification)

Conforme la BeagleBone Black se encuentra lista para ser operada desde un ordenador, se deben realizar las configuraciones para tener acceso a internet a través de la computadora mediante el cable USB con el que se alimenta la BeagleBone Black. En los siguientes párrafos se describe cómo lograr lo previamente dicho.

### *Configuración para obtener acceso a internet por USB en la Beaglebone Black*

Cuando se tienen infraestructuras complicadas para la integración de un nuevo dispositivo a una red con acceso a internet, se recomienda utilizar este método para acceder a la web desde la BeagleBone Black. El requisito principal es una PC con conexión a internet ya sea de forma inalámbrica o alámbrica para realizar la configuración y poder compartir el internet. Basta con ingresar al Centro de redes y recursos compartidos para visualizar que

se ha creado una nueva conexión de red perteneciente a la BeagleBone Black.

Para compartir la conexión a internet, se debe ingresar en las propiedades de la red actual con acceso a internet, dirigirse a la pestaña “Uso compartido” y finalmente seleccionar ambas opciones que se encuentran disponibles tal como muestra la Figura.  28.

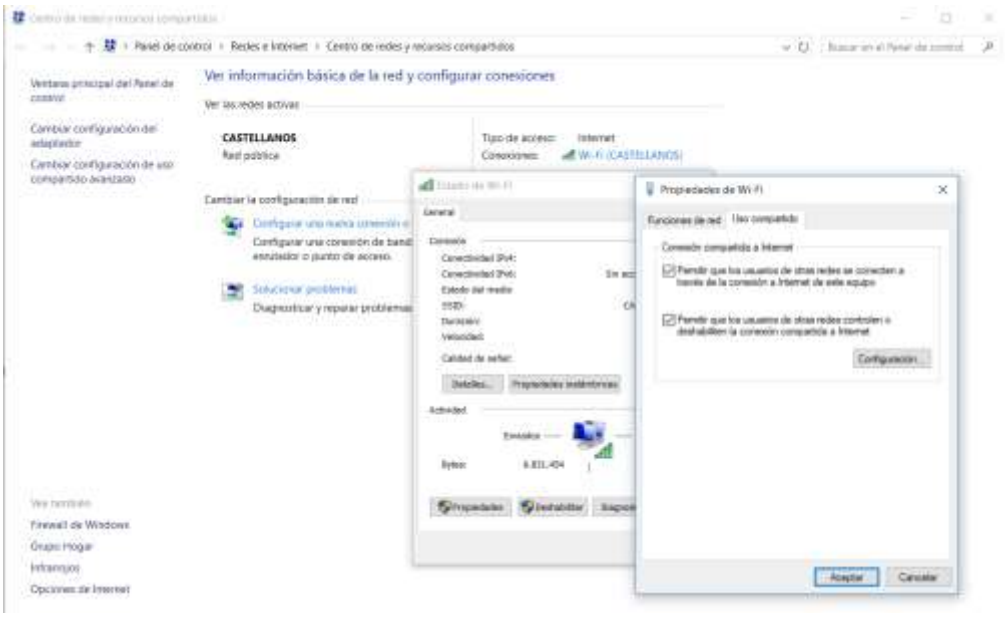



*Figura.  28 Habilitación para compartir internet a través de USB en la BBB.*

Una vez que se tiene la tarjeta conectada al internet se puede empezar a realizar la actualización, instalación de paquetes y librerías en la misma a través de la herramienta PuTTY para manejar la consola de comandos de la BeagleBone Black; el usuario y contraseña por defecto en una instalación limpia del sistema operativo de la Beaglebone Black para el ingreso son *debían* y *temppwd* respectivamente

Si se requiere la presencia de un escritorio remoto para la organización y facilidad de manejo de los archivos que se tienen que modificar y crear, se puede seguir los pasos descritos en el apartado de Escritorio de Raspberry Pi 2 desde Windows 10 en donde se detalla la manera en que el escritorio remoto de Windows accede a la tarjeta sin problema alguno.

### *Configuración de IP estática en la Beaglebone Black*

La Beaglebone Black está configurada de manera predeterminada para usar el protocolo de configuración de host dinámico (DHCP) para la asignación de su dirección IP alámbrica e inalámbrica. Los enrutadores de red normalmente ejecutan un servidor DHCP que asigna un conjunto de direcciones a dispositivos conectados a la red. Si bien el DHCP funciona bien para la mayoría de los dispositivos en una red local, esto puede causar dificultades si desea hacer que la Beaglebone Black sea visible fuera de un firewall doméstico mediante el reenvío de puertos. Esto se debe a que los dispositivos DHCP pueden recibir una dirección IP diferente cada vez que se inician (dependiendo del tiempo de arrendamiento del enrutador).

Para poder dar una dirección IP estática a la tarjeta se deben seguir los siguientes pasos:

1. Conectar directamente la tarjeta a la computadora a través del cable mini-USB, y con la ayuda del navegador de internet de preferencia ingresar a la dirección 192.168.7.2:3000, la cual es la dirección bajo la cual se accede a la distribución de Cloud9 instalada por defecto en la tarjeta. Tras lo cual se obtiene una interfaz como la que muestra la Figura. 29.

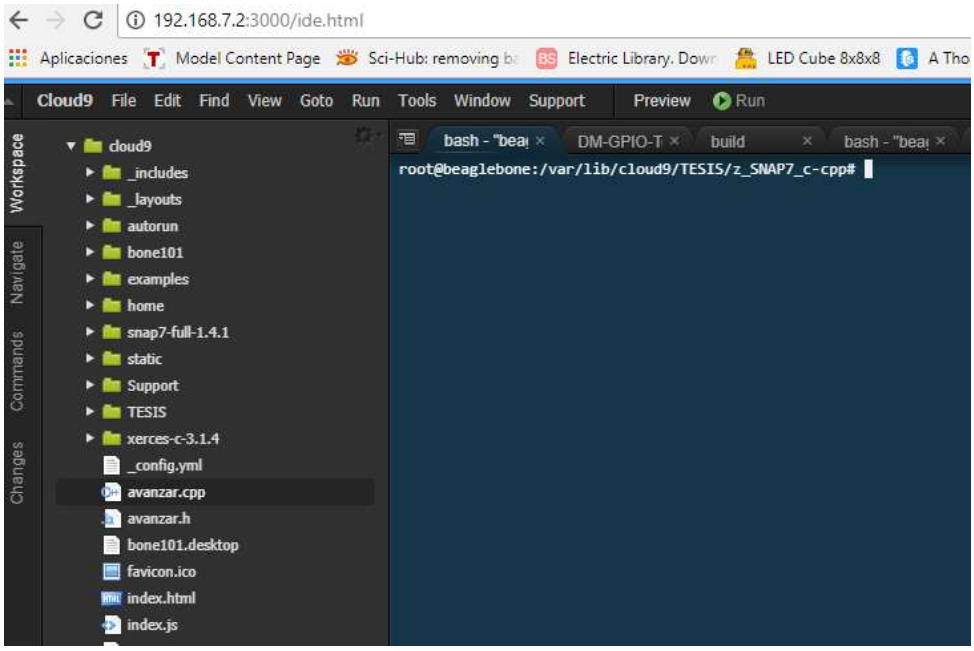



*Figura. 29 Interfaz de la distribución de Cloud9 instalada en la tarjeta Beaglebone Black*

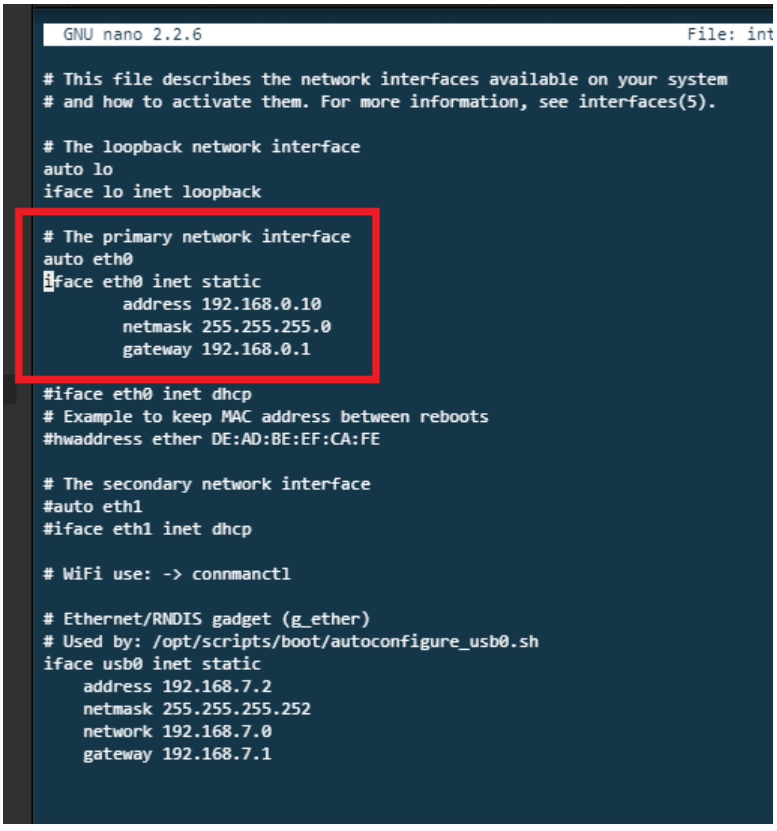



*Figura. 30 Archivo de configuración de red de la Beaglebone Black, resaltando en rojo la configuración realizada para dar la IP estática 192.168.0.10*

2. Abrir un nuevo terminal de comandos e ingresar las líneas de comando **cd/etc/network** y **sudo nano interfaces**. Con lo cual, el editor de texto

nano abrirá el archivo de configuración de red de la tarjeta, en donde se deberán hacer los cambios necesarios para que el archivo termine con una información parecida a lo mostrado en la Figura. 30 dependiendo de la dirección IP que se le quiera asignar.

### *Escritorio remoto de Beaglebone en Windows y Transferencia de archivos*

Para poder trabajar con la tarjeta Beaglebone Black desde una conexión de escritorio remoto en Windows 10, los pasos son los mismos que se siguieron con la tarjeta Raspberry PI en la sección ***Escritorio de Raspberry Pi desde Windows 10*** siendo necesario solo el conocimiento de: la dirección IP bajo la cual se conectó la tarjeta a la misma red de la computadora y el usuario y contraseña para ingresar (*debían y temppwd respectivamente*).

De igual manera para poder transferir archivos entre el ordenador y la tarjeta con la ayuda del programa WinSPC, se sigue la temática mostrada en la sección ***Transferencia de archivos a la tarjeta Raspberry Pi,*** necesitando nuevamente la dirección IP con la que la tarjeta se conectó a la misma red que la computadora, pero esta vez usando el usuario *root* sin contraseña alguna para llenar los datos de Log in.

## 3.3. Bloques de Función de Interfaz Servicio (SIFB)

La creación de una aplicación distribuida solo puede ser posible con la implementación adicional de Bloques de Función de Interfaz de Servicio (SIFB: Service Interface Function Blocks), los cuales comunican datos y eventos entre recursos. Teniendo de esta manera que en donde sea que se requiera alguna forma de interacción entre los FBs dentro de un recurso con el mundo exterior, es necesaria la intervención de los SIFBs. La norma IEC-61499 no contempla estándares para tipos particulares de SIFBs pero si estipula que deben ser definidos con un set específico de variables de entrada/salida y eventos de entrada/salida. Adicionalmente, existe una

notación especial que se utiliza para describir la secuencia de interacciones tal como el envío de una petición y la espera de una respuesta, las cuales suceden externamente durante la ejecución de un SIFB (R. Lewis, 2008).

En un controlador industrial, es necesario leer los valores de las entradas físicas de los sensores involucrados, y también escribir valores de salida en actuadores. También hay requisitos para transmitir valores a través de enlaces de comunicaciones serie, para enviar copias de datos a controladores externos, visualizadores de unidades y entradas de lectura desde paneles de visualización y otros equipos HMI. Todas estas acciones son ejemplos de interacciones externas con un recurso que puede ser modelado usando diferentes tipos de SIFB. Algunos ejemplos de SIFBs que pueden ser utilizados para modelar un sistema de control industrial se muestran en la Tabla 6.

*Tabla 6*

*Ejemplos de SIFBs más utilizados en modelos IEC-61499*

| Nombre de FB | Servicio Proporcionado | Ejemplo de Aplicación |
|---|---|---|
| IO_Writer | Permite que un recurso transmita uno o más valores a dispositivos de E/S conectados localmente.<br><br>Nota: IEC 61499 supone que el subsistema de E/S del controlador se encuentra fuera del recurso, pero se puede acceder mediante SIFB | Utilizado para actualizar el valor de actuadores manejando dispositivos físicos como: válvulas, calentadores, bombas. |
| IO_Reader | Permite que un recurso reciba el último valor o conjunto de | Leer el último conjunto de valores de la posición de |

| | | |
|---|---|---|
| | valores provenientes de uno o más dispositivos. | dispositivos de entrada, por ejemplo: los micro-switches de un mecanismo de posicionamiento robótico. |
| **Publisher** | Transmite el valor o conjunto de valores de una o más variables a uno o múltiples recursos externos que den soporte al servicio de “”Suscripción” | Transmisión continua e valores de salida a un cierto número de controladores remotos encargados de manejar actuadores. |
| **Subscriber** | Recibe uno o más valores de un recurso externo que ofrece el servicio de “Publicación”.<br><br>Nota: El publicador y el suscritor identifican un set particular de valores mediante una dirección única y específica de red o un servicio de identificador. Puede haber múltiples suscritores recibiendo los valores de un solo publicador. | Para recibir regularmente el flujo de valores de un recurso externo dado, por ejemplo: para actualizar los valores de una pantalla HMI. |

## Definición de SIFBs

En la norma IEC-61499 se especifica en términos generales la manera en la que un SIFB es definido. Cada uno de ellos provee un servicio de alguna clase, por ejemplo: lectura de entradas/salidas, publicación de valores en la red, recepción de valores, entre otros. Es por esto que se propone que el nombre que se le asigne a cada SIFB refleje el nombre del servicio que el bloque provee, por ejemplo “ValvePositioner” o “HMIWriter”.

## Elementos de un SIFB

Al igual que un FB genérico, los SIFBs poseen dos componentes básicos en su estructura, una cabecera en donde se genera el flujo de eventos y un cuerpo en donde se produce el flujo de datos. Es necesario acotar que, un SIFB puede inicializar un servicio en respuesta a un estímulo proveniente de una aplicación, aunque alternativamente el SIFB puede responder a una petición externa como puede ser la señal de un panel HMI para pedir los valores de una aplicación. Los SIFBs pueden ser modelados para ser compatibles con ambos tipos de comportamientos.

### *Entradas y salidas estándar para un SIFB*

Cada SIFB a generar debe usar las siguientes entradas y salidas estándar, aunque no es mandatorio el uso de todas ellas y se puede omitir su implementación dependiendo de la aplicación que se le dé:

- **Eventos de Entrada**
  - **INIT:** Este evento de entrada se utiliza para inicializar un servicio en particular brindado por el bloque. Por ejemplo, puede inicializar un servicio para proveer la transmisión de datos sobre un enlace serial. Este evento es típicamente enviado con un número de parámetros de entrada para caracterizar el tipo de servicio, tal como dirección de red y baud rate.
  - **REQ:** Este evento inicializa una petición para obtener los datos de un agente externo. Por ejemplo, este evento puede ser utilizado para inicializar la transmisión en base a una petición por datos.
  - **RSP:** Este evento inicializa la transmisión de una respuesta a un agente externo. Por ejemplo. Puede enviar los datos a un dispositivo HMI en respuesta de una petición por datos.

- **Eventos de Salida**
  - **INITO:** Este evento de salida señala que el SIFB ha completado su inicialización; por ejemplo, en el caso de haber recibido un evento

INIT no necesariamente significa que el servicio ha sido inicializado correctamente, y es por esto que una salida de “STATUS” es implementada para este propósito.

- **CNF:** El evento de “confirmación” se genera cuando el bloque ha completado la transmisión de una petición a un agente externo. Por ejemplo, debe ser utilizado para indicar que una petición de lectura de un punto de entrada/salida particular ha sido procesada por el subsistema del controlador de entrada/salida.
- **IND:** El evento “indicador” es generado cuando el SIFB ha recibido una respuesta de un agente externo. Por ejemplo, un evento IND se debe producir cuando una lectura de entrada/salida del controlador del subsistema ha obtenido el valor del sensor seleccionado.

- **Datos de Entrada**
  - **QI (BOOL):** La entrada de datos QI es utilizada con el evento de entrada INIT como un calificador simple. Cuando tiene el valor “true” indica que el servicio dado por el bloque debe ser inicializado, mientras que cuando tiene el valor de “false” indica que el servicio dado debe ser finalizado.
  - **PARAMS (ANY):** Esta entrada representa una estructura de datos que posee una serie de valores asociados al SIFB y define sus características particulares. El número y tipo de datos dentro de la estructura, al igual que sus valores, son específicos para el tipo de servicio dado por el bloque. Esta entrada es únicamente utilizada con el evento INIT debido a que es utilizada para la inicialización del servicio. Por ejemplo, la entrada PARAMS para un SIFB de comunicación contendrá los valores de: dirección de red, velocidad de transmisión, tipo de paridad, etc.
  - **SD_1, … SD_N (ANY):** Estas entradas de datos son utilizadas para enviar datos con respuestas y peticiones. El número de entradas y sus datos serán específicos al tipo de servicio dado por el bloque (es por esto que se muestra la notación SD_1, … SD_N). Por ejemplo, al leer los datos de las diferentes salidas de un dispositivo que pueden ser datos: booleanos, enteros, cadenas, etc.
- **Datos de Salida**

- **QO (BOOL):** Éste evento de salida se usa para indicar cuando se ha completado el servicio para cualquier evento de entrada. Por ejemplo, al seguir la inicialización de un evento INIT, un valor de “true” indicará una correcta inicialización, mientras que un valor de “false” indicará que el evento ha fallado en su inicialización.
- **STATUS (ANY):** Esta salida puede ser seteada con cualquiera de los eventos de entrada y es utilizada para proveer el estado de haber procesado el evento seleccionado. Por ejemplo, STATUS va a ser seteado cuando un servicio es inicializado usando el evento INIT, y se tiene una ejecución exitosa. De igual manera, cuando un evento REQ se usa para transmitir valores a un dispositivo remoto y subsecuentemente falla, STATUS es usado para almacenar la razón de fallo.
- **RD_1, … RD_N (ANY):** Estas salidas son usadas para transmitir los datos recibidos de confirmaciones e indicaciones. De igual manera que con las entradas SD_1 … SD_N, el número de salidas y sus tipos de datos serán específicas al servicio dado por el bloque. Por ejemplo, cuando se ha generado una petición de lectura de un conjunto de 10 señales, sus valores se verán reflejados en las salidas RD_1 … RD_10.

En la Figura. 31 se muestran dos SIFBs dados como ejemplos en el estándar IEC-61499. Estos bloques aún son “genéricos” debido a que no se les ha dado un número específico de entradas y salidas. Los tipos de datos de las entradas y salidas son definidos usando la palabra ANY, dando a entender que estos bloques son plantillas para estructurar nuevos SIFBs. Es por esto que un mayor detalle de números de entradas/salidas y sus tipos es necesario para utilizarlos en aplicaciones.


*Figura. 31 Plantillas de SIFBs genéricos: a) REQUESTER, b) RESPONDER*

# CAPÍTULO IV

## 4. Propuesta

En el capítulo 3 se analizó el caso de estudio en donde se pudo presentar como base de la propuesta el uso de las tarjetas Raspberry Pi 2 Modelo B, Raspberry Pi 3 Modelo B y BeagleBone Black; debido a su capacidad para operar bajo un sistema operativo embebido lo que transforma a las tarjetas de desarrollo en dispositivos totalmente compatibles con la norma IEC-61499 a un costo sumamente accesible para los investigadores de cualquier campo de estudio.

La propuesta a presentar en el siguiente trabajo de investigación se centra en el desarrollo y generación de los FBs necesarios para poder implementar un sistema de control distribuido en el sistema modular FESTO presentado en el capítulo ***CASO DE ESTUDIO*****.** Es por esto que esencialmente se exhiben dos grupos generales de FBs:

- **FBs de Control:** Estos Fbs encapsulan los algoritmos desarrollados bajo el modelo jerárquico planteado en ISA-88 para el control individual de las estaciones de: selección, almacenamiento y clasificación.
- **FBs de Entrada/Salida:** Estos FBs se encargan de encapsular la codificación necesaria para poder controlar la entrada y salida de datos física y lógicamente. Es por esto que dentro de este grupo se encuentran los FBs desarrollados para el control de los puertos de propósito general (GPIO) de las tarjetas propuestas, y los FBs que dentro de su codificación incluyen el protocolo de comunicación S7 de la casa comercial SIEMENS utilizados para poder comunicar a las tarjetas con autómatas de dicha marca.

Adicionalmente, como las tarjetas propuestas fueron creadas originalmente con fines de desarrollo e investigación a bajas tensiones, se presenta como propuesta de hardware una tarjeta de acople, que permite

traducir estas señales de baja tensión (0 a 3.3V) a señales estándar de tensión utilizadas industrialmente (0 a 24V) y de esta manera generar la correcta unión entre los circuitos de control con los de potencia.

## 4.1. Metodología de desarrollo de FBs

### Generación de FBs en 4DIAC-IDE

Uno de los pilares bajo los cuales trabaja la norma IEC-61499 es la reutilización de software, por lo cual se busca que cada FB generado pueda ser utilizado en más de una aplicación, optimizando de esta manera la programación de las aplicaciones. El entorno de programación 4DIAC-IDE permite el desarrollo de los nuevos FBs de una manera específica en la que solo pertenecerán al sistema en el cual se crearon, o de una manera global en la que podrán ser utilizados por cualquier nuevo sistema desarrollado tras su creación.

Para desarrollar FBs que puedan ser utilizados por cualquier sistema a desarrollar por defecto se deben seguir los siguientes pasos:

1. En la ventana de navegación dirigirse a la pestaña *“Type navigator”*, localizar la carpeta *“Tool Library”*, desplegar sus propiedades, seleccionar *“New”* y posteriormente seleccionar *“Folder”* como se muestra en la Figura. 32.
2. En la ventana de ayuda emergente dar un nombre a la carpeta y seleccionar *“Finish”*. Tras esto, se desplegará el apartado de *“Tool Library”* indicando la carpeta generada como se muestra en la Figura. 33.
3. Sobre la carpeta agregada se debe desplegar sus opciones, dirigirse a “*New”* y seleccionar *“Type”* para desplegar el asistente de creación de un nuevo FB como se muestra en la Figura. 34.
4. En la ventana emergente agregar el nombre del FB. Como son FBs encaminados a brindar un servicio, en el apartado “*Type*” seleccionar la opción “*ServiceInterface.fbt”* como se muestra en la Figura. 35. Luego de aceptar, automáticamente el FB generado se desplegará en el área

de trabajo para poder manipularlo acorde a nuestras necesidades como se muestra en la Figura. 36.

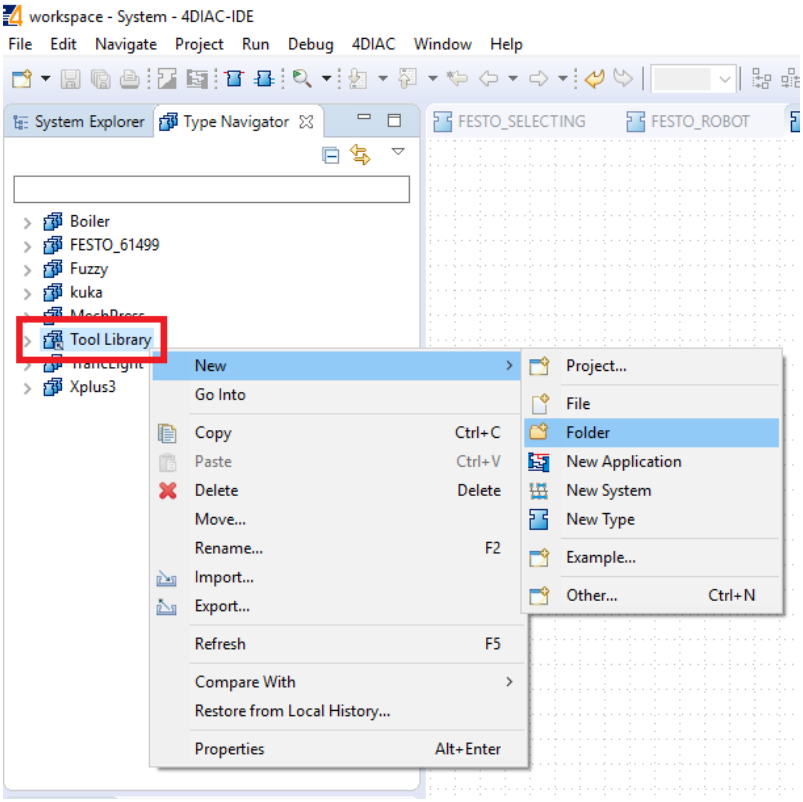



*Figura. 32 Generación de una nueva carpeta dentro de la ubicación global de los FBs de 4DIAC-IDE*

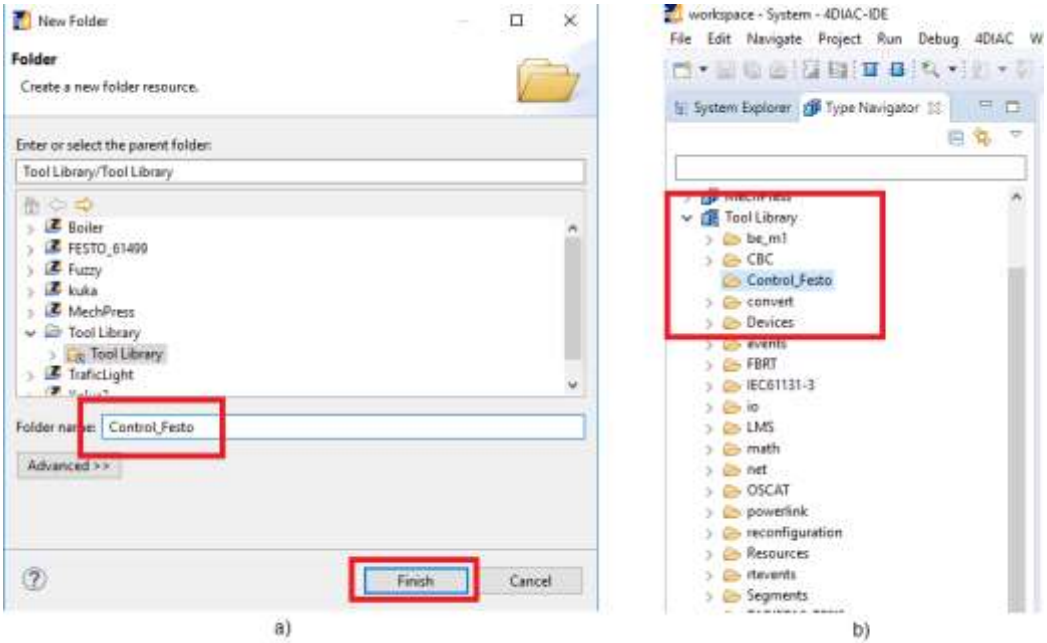



*Figura. 33 a) Nombramiento de una nueva carpeta b) Ubicación dentro del apartado global "Tool Library"*

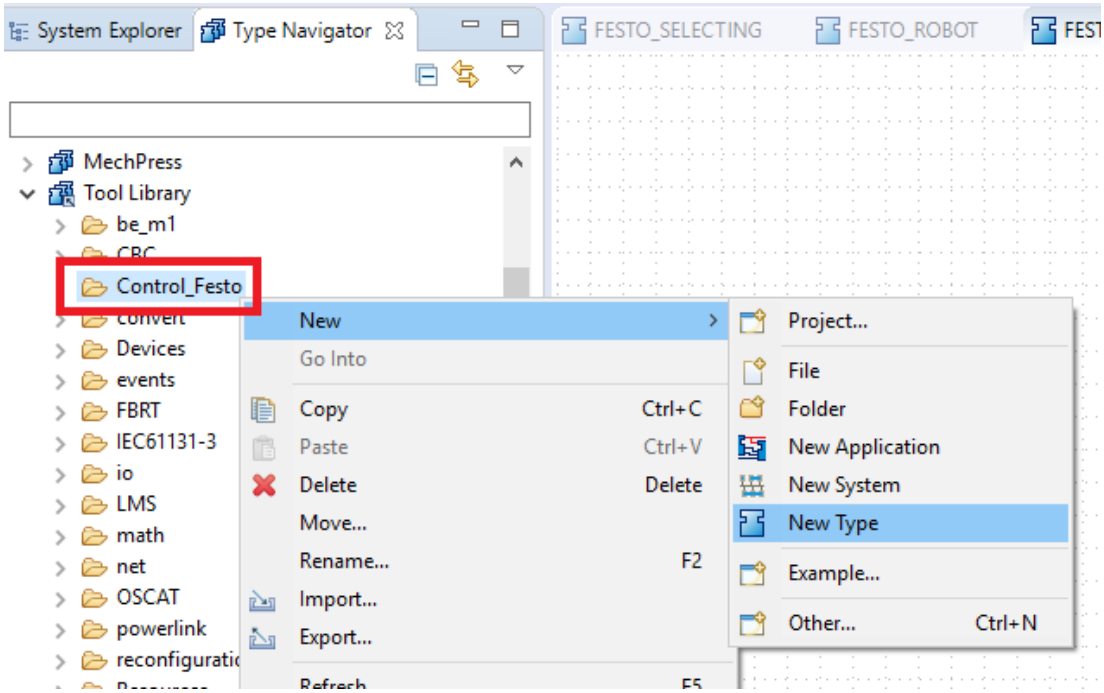



*Figura. 34 Pasos iniciales en la creación de un FB*

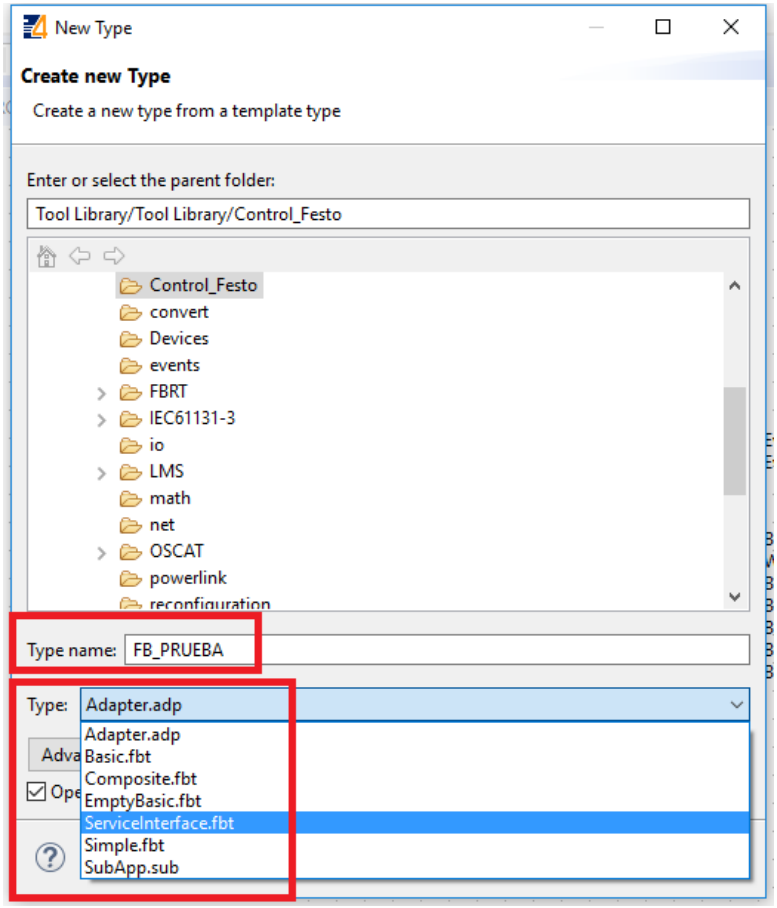



*Figura. 35 Nombramiento y selección de tipo de un nuevo FB*

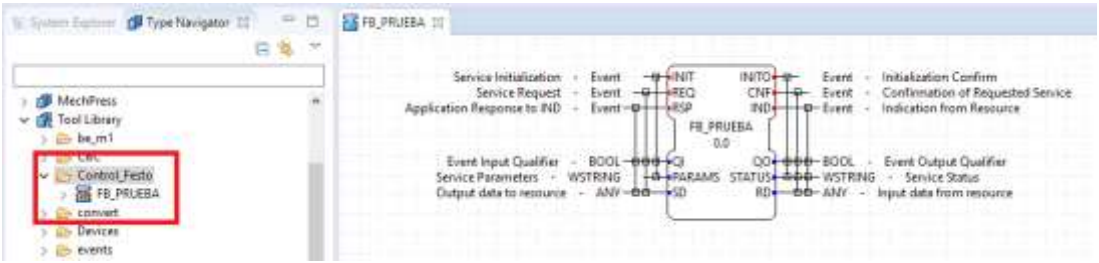



*Figura. 36 FB preparado para ser manipulado en el área de trabajo tras su creación.*

El FB generado se crea con las entradas y salidas genéricas descritas en el apartado ***Entradas y salidas estándar para un SIFB***, pero como se discutió en el mismo no es regla que los FBs trabajen con todas ellas. Dependiendo de las necesidades y la finalidad a la cual se destina el FB, se pueden añadir y eliminar las conexiones que consideremos necesarias (1) dando clic derecho en el FB para desplegar las opciones de creación de eventos de entrada y salida, entradas de datos y salidas de datos como muestra la Figura. 37; o (2) dando clic derecho sobre uno o varios de ellos (previamente marcados presionando la tecla CTRL) para desplegar la lista de acciones con la opción de *"eliminar"* incluida, de la misma manera que muestra la Figura. 38. Cabe recalcar que esta última acción puede ser replicada de igual forma omitiendo los comandos por mouse y presionando la tecla DELETE del teclado.

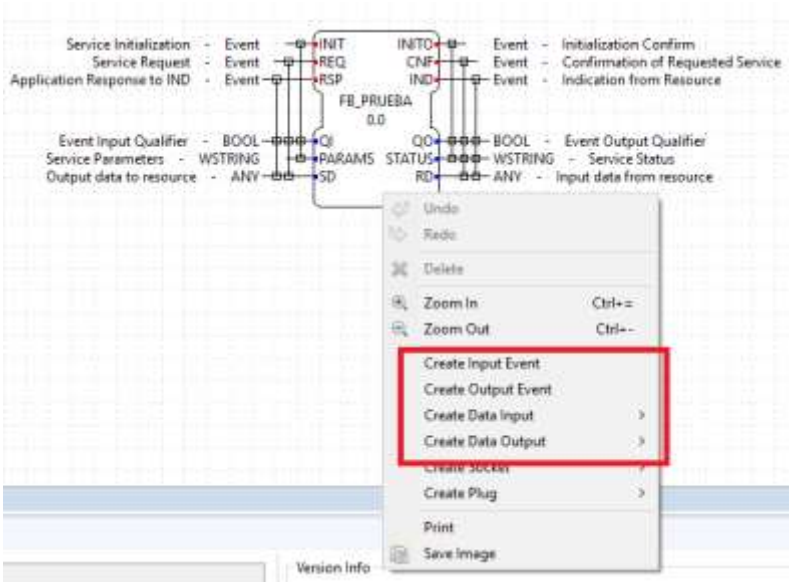



*Figura. 37 Opciones para creación de eventos de entrada y salida, entradas de datos y salidas de datos*

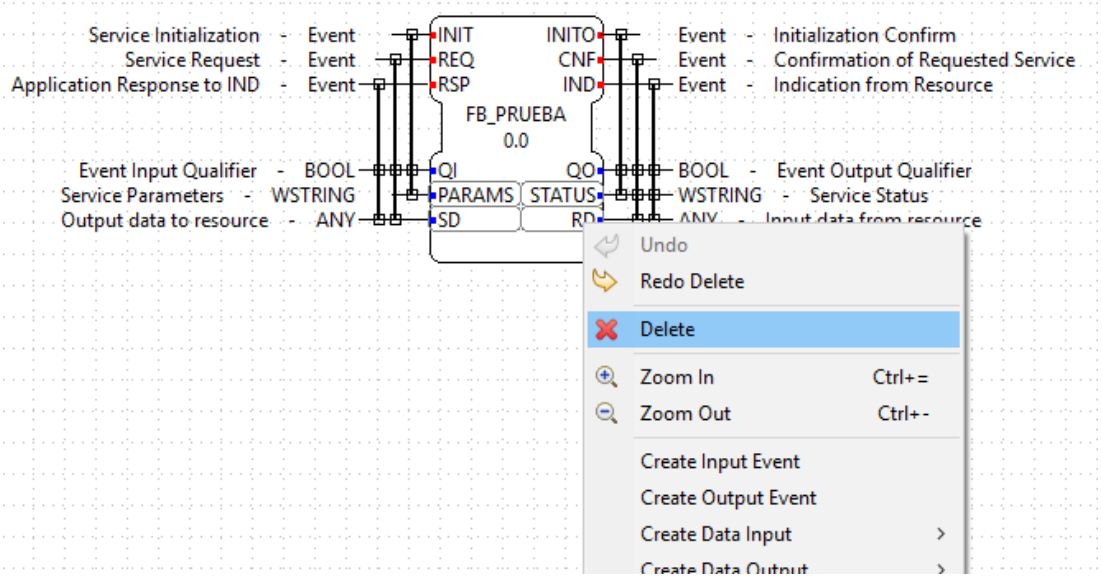



*Figura. 38 Menú de acciones desplegado con la opción de eliminar incluida.*

Tras la metodología detallada, el FB resultante aún no se encuentra preparado para su integración dentro de una aplicación funcional debido a que todavía es un cascarón vacío, porque su programación asociada no va más allá de la declaración de variables relacionadas a los eventos, entradas y salidas de datos definidas en su estructura. Por esto, una vez creado un FB es necesario generar un sistema del cual podamos hacer la exportación de su código fuente relacionado.

## Exportación de FBs

Genéricamente junto a cada FB que sea creado, se generan automáticamente dos archivos compatibles con el lenguaje de programación C++ relacionados a dicho bloque, los cuales son: (1) un archivo .cpp en el cual se deben añadir los algoritmos de programación deseados y (2) un archivo de cabecera .h que contiene a manera de variables declaradas las entradas y salidas del bloque.

Los FBs que se creen con la metodología detallada en el apartado ***Generación de FBs en 4DIAC-IDE*** aun no pueden ser exportados, debido a que esta acción no puede ser realizada con elementos localizados en la carpeta global *“Tool Library”*. Es por esto que, para poder exportar los

archivos fuentes de FBs en general, es necesario que éstos se encuentren localizados en la librería de FBs de un sistema.

Es recomendable hacer un análisis previo de todas las funciones que se necesitarán dentro de un sistema, para de esta manera desarrollar todos los FBs correspondientes a esas funciones y así tener que generar únicamente un nuevo sistema para que automáticamente todas las funciones ubicadas en *"Tool Library*" sean añadidas a su carpeta individual de funciones; para lo cual es necesario:

1. En cualquier parte en blanco del explorador del sistema dar clic derecho, seleccionar *new* y posteriormente seleccionar *new system* como muestra la Figura. 39.
2. En la ventana emergente, asignar un nuevo nombre al sistema y posterior a esto presionar el botón *finish*, como se muestra en la Figura. 40. Tras lo cual, el nuevo sistema automáticamente aparecerá en el explorador de sistemas de 4DIAC-IDE como se muestra en la Figura. 41(a), conteniendo todas las funciones que se encuentren en la carpeta *"Tool Library"* hasta el momento de su creación, como se muestra en la Figura. 41(b).

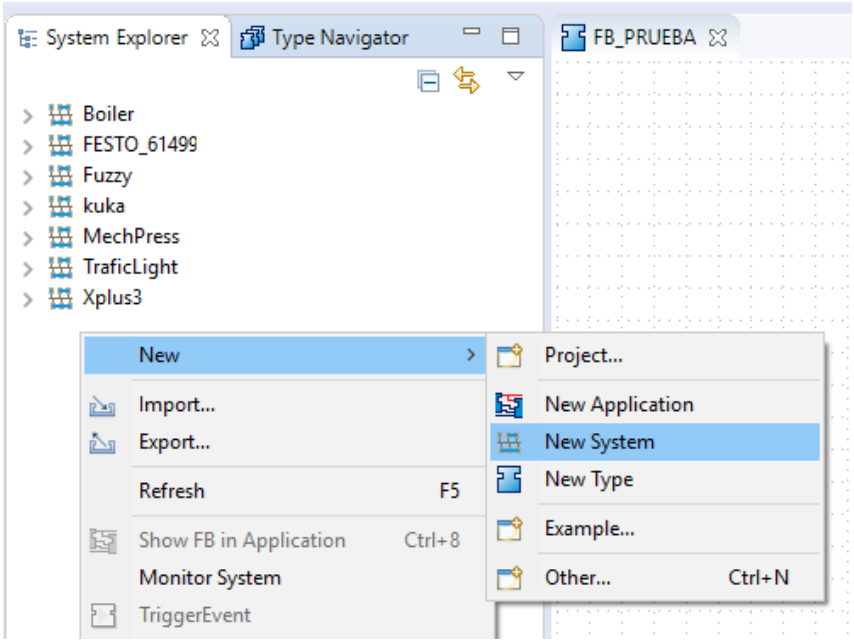



*Figura. 39 Primera etapa de generación de un nuevo sistema en 4DIAC-IDE*

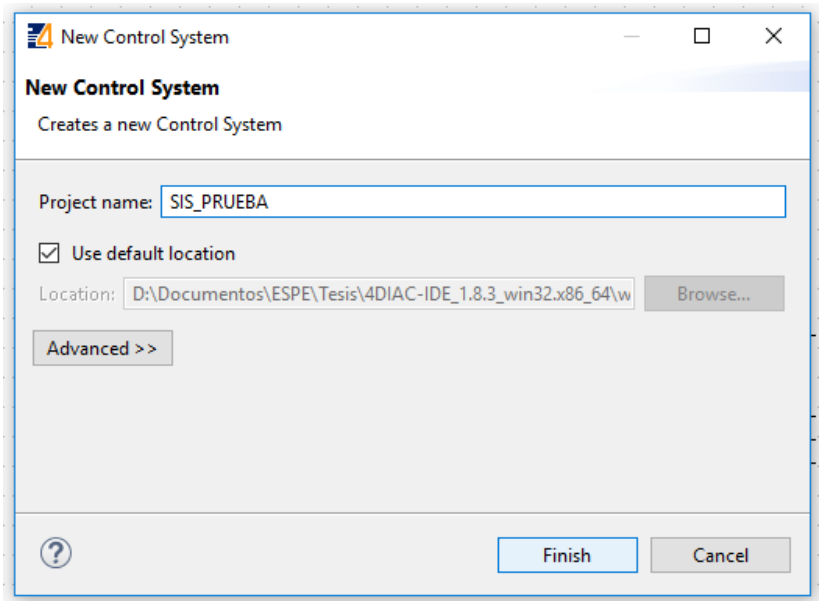



*Figura. 40 Asistente de creación de un nuevo sistema en 4DIAC_IDE*

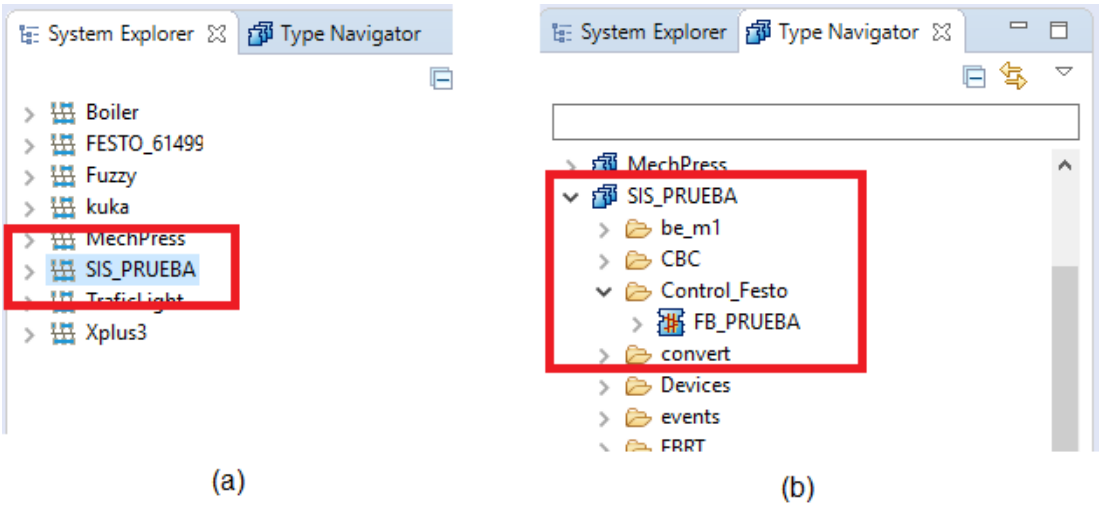



*Figura. 41 (a)Nuevo sistema ubicado en el explorador de sistema de 4DIAC-IDE (b)Carpeta de funciones individual del nuevo sistema, en donde se incluyen todas las funciones ubicadas en la carpeta "Tool Library".*

De igual manera en caso de querer añadir cualquier otra función durante el desarrollo del sistema, se puede generar su FB en la carpeta *"Tool Library"* y posterior a esto, copiar directamente la función y pegarla en el directorio de funciones individuales del sistema como se puede observar en la Figura. 42, en donde se hace el copiado mediante interacciones con mouse. El proceso de copia también se lo puede realizar mediante atajos por

teclado (CTRL+C & CTRL+V) tras la selección de todos los FBs que se deseen copiar.


*Figura. 42 Secuencia de copiado de FBs de la carpeta de funciones globales Tool Library al directorio de funciones individuales de un sistema. (a) Copia de FBs desde la carpeta de funciones Tool Library (b)Pegado de FBs en el directorio individual de un sistema (c) FBs reubicados en un nuevo sistema*

Una vez que se tengan ubicados todos los FBs que se deseen exportar dentro de la carpeta de funciones de un sistema el procedimiento que se debe seguir para la exportación de sus archivos fuentes es el siguiente:

1. Dentro del sistema en uso, marcar todos los FBs que se deseen exportar. Posterior a esto desplegar sus opciones y seleccionar *export* como se muestra en la Figura. 43.
2. En la ventana emergente asegurarse de que la opción *4DIAC Type Export* de la carpeta *4DIAC* se encuentre seleccionada como asistente de exportación como se muestra en la Figura. 44, tras lo cual presionar *Next*.
3. En el asistente de exportación emergente en primera instancia seleccionar el destino en el cual se desean exportar los archivos fuentes. Es muy recomendable generar una nueva carpeta en una ubicación de fácil acceso para tener una fácil localización de los archivos tras su exportación. De igual manera asegurarse de que el exportador tenga seleccionada la opción *FORTE 1.X* como se muestra en la Figura. 45, tras lo cual presionar el botón *Finish.*

4. Tras este último paso, el asistente de exportación desaparecerá y al dirigirse con el explorador de archivos de Windows a la ubicación predeterminada en el asistente, podremos encontrar los archivos fuente exportados de los FBs seleccionados como se muestra en la Figura. 46.

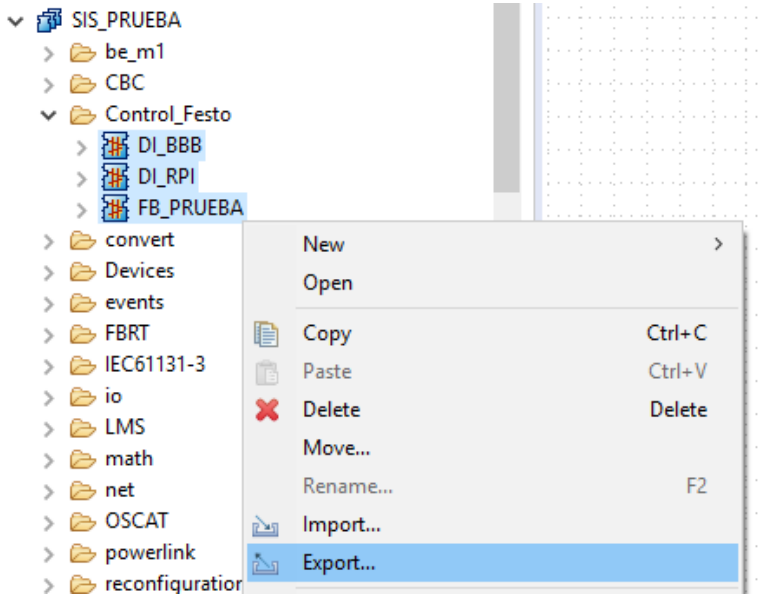



*Figura. 43 Selección de FBs a exportar e inicios de exportación a través de interacciones por mouse*

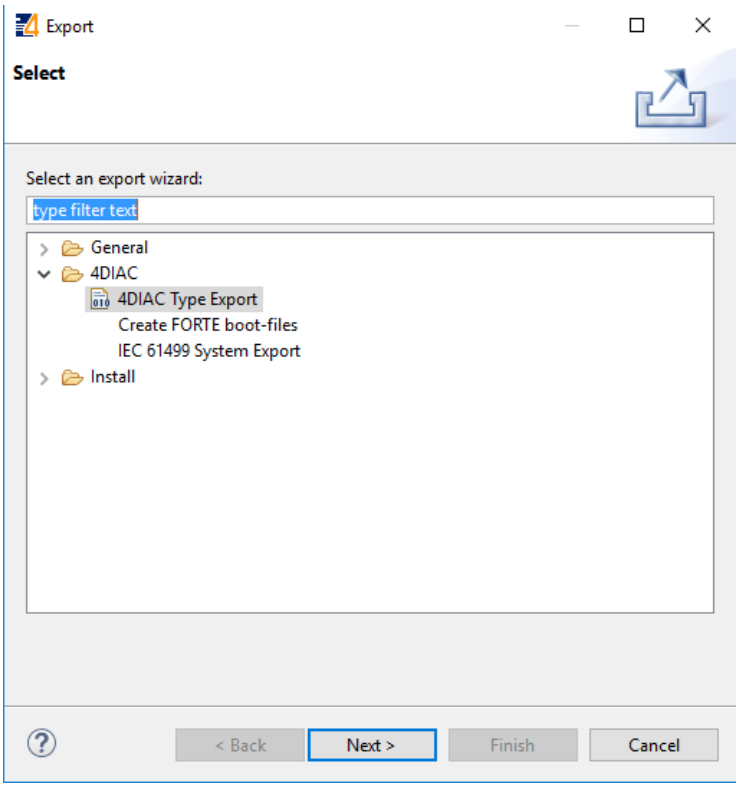



*Figura. 44 Selección de un asistente de exportación*

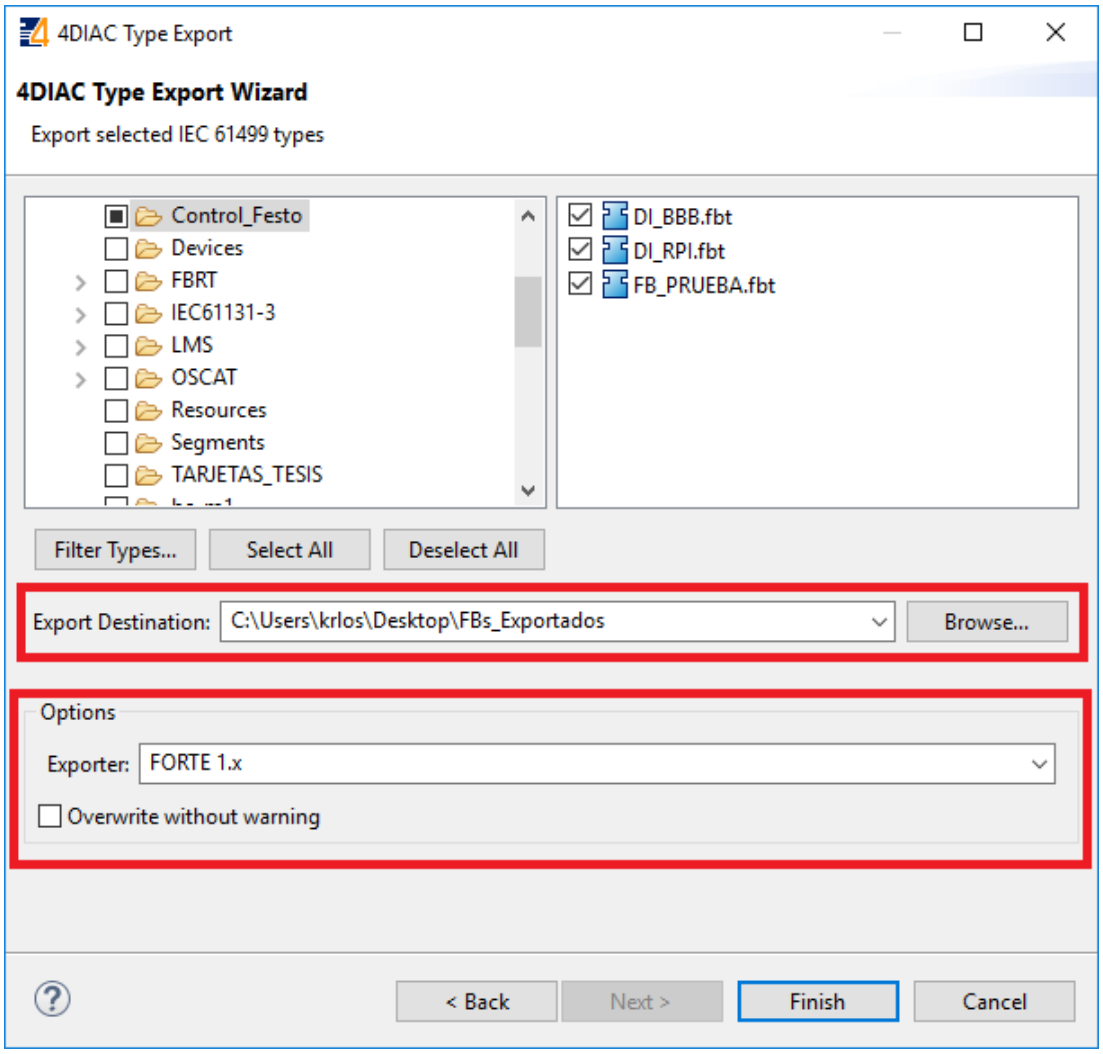



*Figura. 45 Asistente de exportación con la configuración necesaria para exportar los archivos fuente de los FBs marcados*

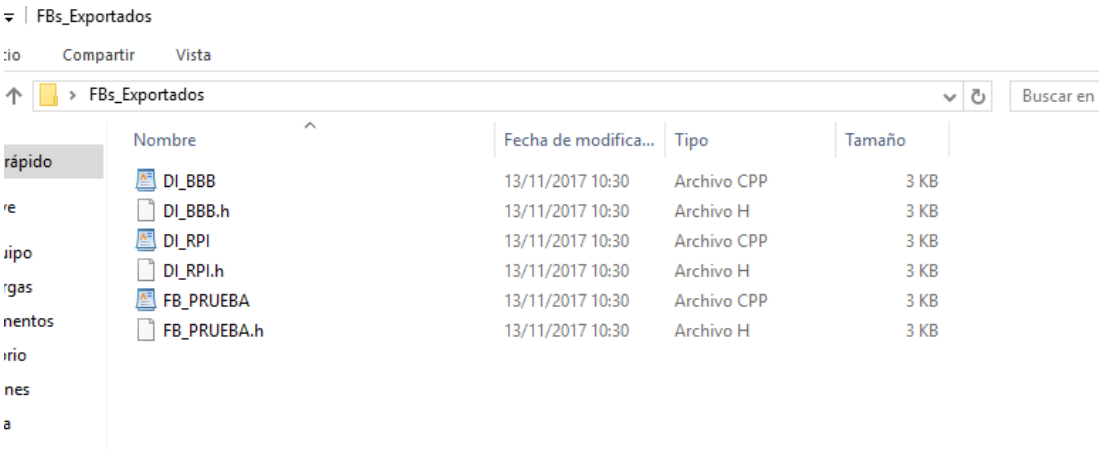



*Figura. 46 Archivos fuente de los FBs exportados en la ubicación preseleccionada en el proceso de exportación*

Como se discutió previamente a inicios de este apartado, los archivos generados aún no se encuentran listos para su implementación dentro de una aplicación funcional debido a la programación básica dentro de los mismos. Es por esto que mediante la ayuda de editores de programación compatibles con el lenguaje C++ se puede modificar el código de los archivos y de esta manera encapsular los algoritmos necesarios para que el FB pueda cumplir la función para la cual fue desarrollado. Adicionalmente, como punto favorable de la exportación de archivos en lenguaje de programación C++ se tiene la posibilidad del uso de librerías y sub-funciones; haciendo posible la reutilización de conocimiento ya generado por varios desarrolladores a lo largo del mundo.

Es mucho más recomendable desarrollar la programación de los archivos fuente dentro de los dispositivos en los cuales se van a implementar, por lo cual para este proyecto de investigación con la ayuda de WinSPC se copió la carpeta con los archivos exportados al escritorio de las Raspberry Pi y la Beaglebone Black. En el caso de las Raspberrys el editor de programación a utilizar es *Genie,* el cual se instala por defecto en las imágenes de aspbian; mientras que en la Beaglebone se utiliza como editor la distribución de Cloud9 instalada por defecto en la imagen de Debian para Beaglebone Black.

## 4.2. Incorporación de FORTE en tarjetas de desarrollo

En el apartado ***4DIAC-IDE / FORTE*** se discutió que FORTE no es otra cosa que el entorno de ejecución instalado en el dispositivo de control, lo que significa que FORTE básicamente es un aplicación que debe ser compilada en base a las funcionalidades que se le quieran dar. Es por esto que la manera en la que se incluyen los FBs desarrollados en FORTE, es mediante la incorporación de sus archivos fuente en un apartado específico de la carpeta que contiene los archivos base de FORTE.

Para tener los archivos base de FORTE en las tarjetas de desarrollo es necesario:

1. Ir a la página oficial de 4DIAC en internet y en el apartado de *Downloads* encontrar las distribuciones de FORTE disponibles. Es recomendable descargar la penúltima versión de las opciones que se muestran en la página (FORTE 1.8.2 a octubre de 2017), debido a que la última versión disponible (FORTE 1.8.3 a octubre de 2017) generalmente se encuentra en desarrollo y puede tener errores sin solucionar.
2. Con la ayuda de WinSPC copiar el comprimido descargado y ubicarlo en una carpeta con privilegios de lectura y escritura en las tarjetas de desarrollo. Generalmente es recomendable dejarla en el escritorio del dispositivo.
3. Finalmente, descomprimir el archivo en la carpeta previamente seleccionada, tras lo cual se tendrán los archivos base de FORTE listos para su manipulación como se muestra en la Figura. 47.


*Figura. 47 Archivos base de FORTE ubicados en escritorio de la tarjeta Beaglebone Black*

## Compilación y generación de archivo ejecutable FORTE

Como ya se mencionó en el apartado anterior para poder usar los FBs desarrollados junto a FORTE es necesario crear un nuevo módulo en los archivos base de FORTE, en donde se adicionarán sus archivos exportados. Para esto es necesario ingresar en la carpeta contenedora de FORTE y ubicarse en la dirección */src/modules*, en donde se debe crear una carpeta (en el caso de este proyecto *CPPs_FBs*) como se ve en la Figura. 48.

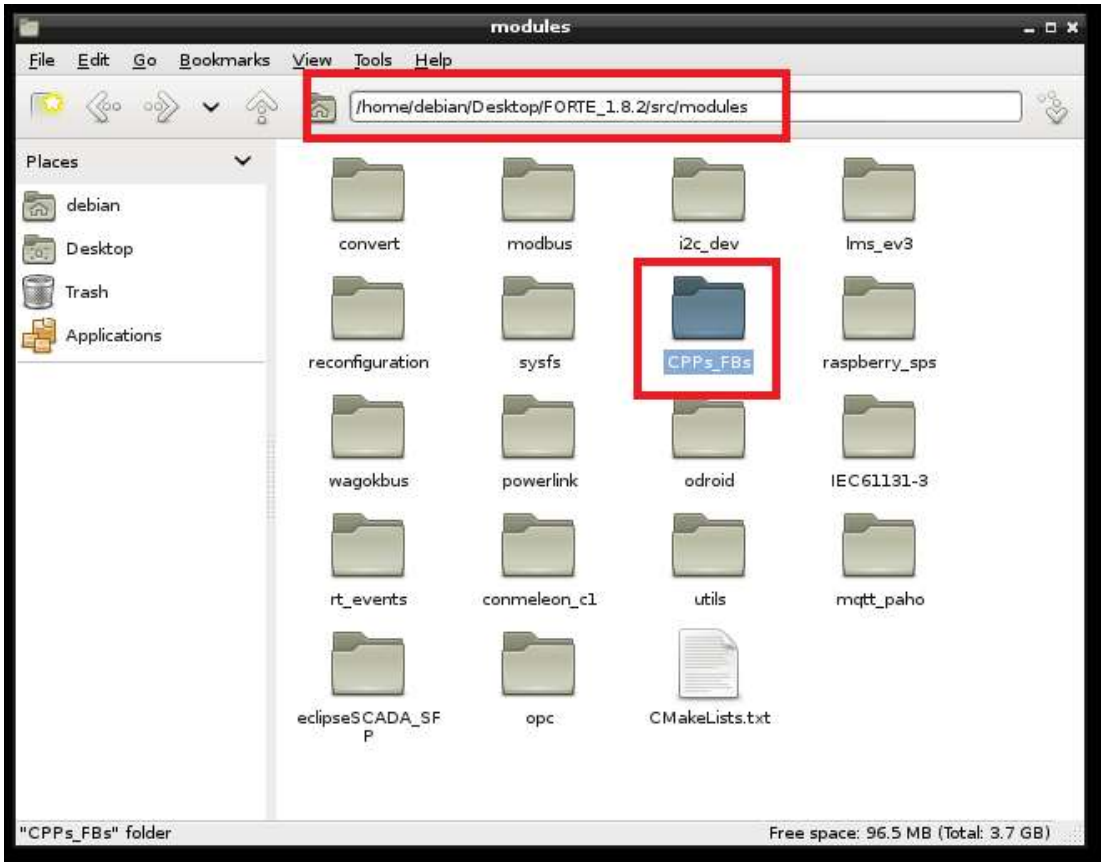



*Figura. 48 Carpeta CPPs_FBs creada en la localización /src/modules para contener los archivos del nuevo módulo de funciones en desarrollo.*

Dentro de la carpeta de módulo generada se debe añadir una carpeta en donde se almacenarán los archivos exportados y un archivo de texto con el nombre "*CMakeLists"* como se muestra en la Figura. 49. El archivo *CMakeLists* es destinado a contener las directrices de compilación del nuevo

módulo, por lo cual es necesario escribir una cabecera informativa y la codificación de compilación como se muestra en la Figura. 50.

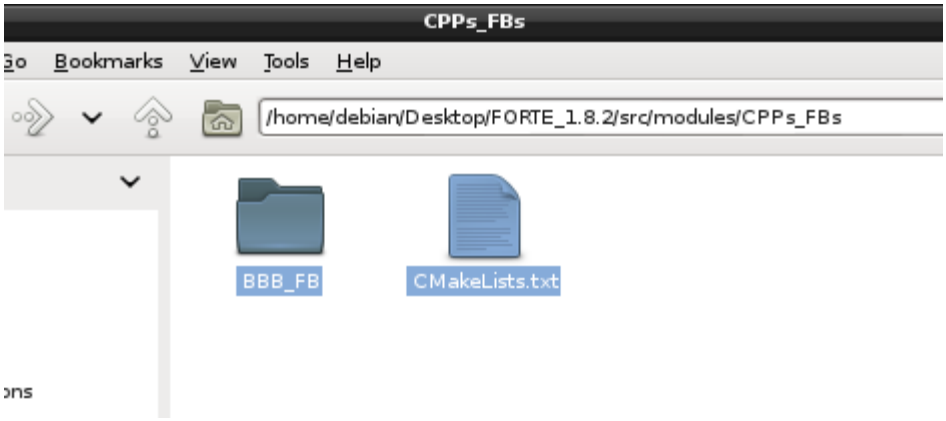



*Figura. 49 Carpeta BBB_FB contenedora de funciones y archivo CMakeLists.txt generados dentro de carpeta de módulo*


*Figura. 50 Contenido del archivo CMakelists.txt*

Las directrices de compilación *forte_add_sourcefile_h(Carpeta_de_archivos/Archivo.h)* y

*forte_add_sourcefile_hcpp(Carpeta_de_archivos/Archivo)* deben ser añadidas acorde al número de archivos exportados que se posean, es decir un par de instrucciones por cada par de archivos exportados y que se hayan añadido en la carpeta que contiene el resto de ficheros.

En caso de utilizar funciones de librerías dentro de la codificación de los archivos fuente exportados, dentro de su carpeta contenedora se debe crear otra carpeta en donde se ubicarán las librerías utilizadas como se muestra en la Figura.  51 y la Figura.  52.

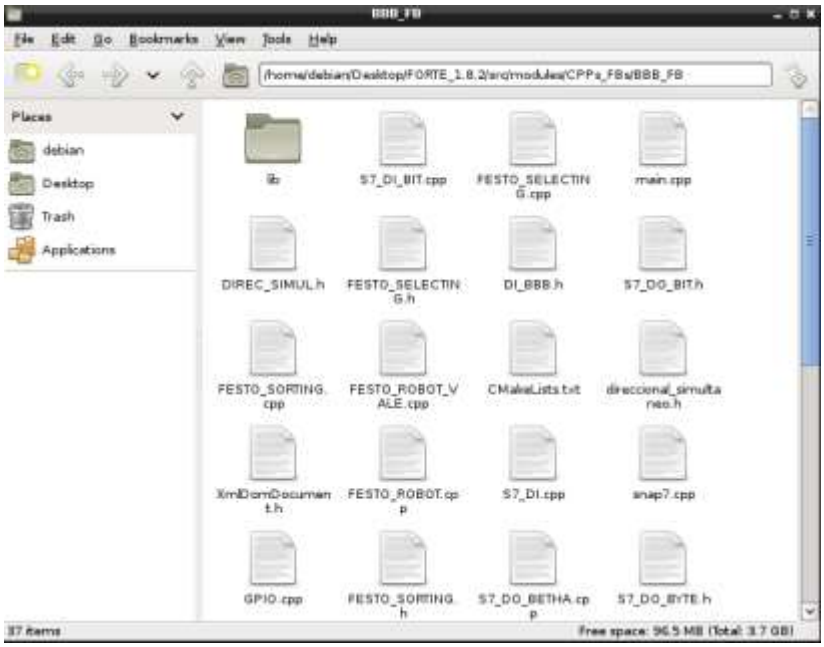



*Figura.  51 Carpeta de librerías y archivos añadidos a la carpeta de funciones BBB_FB*

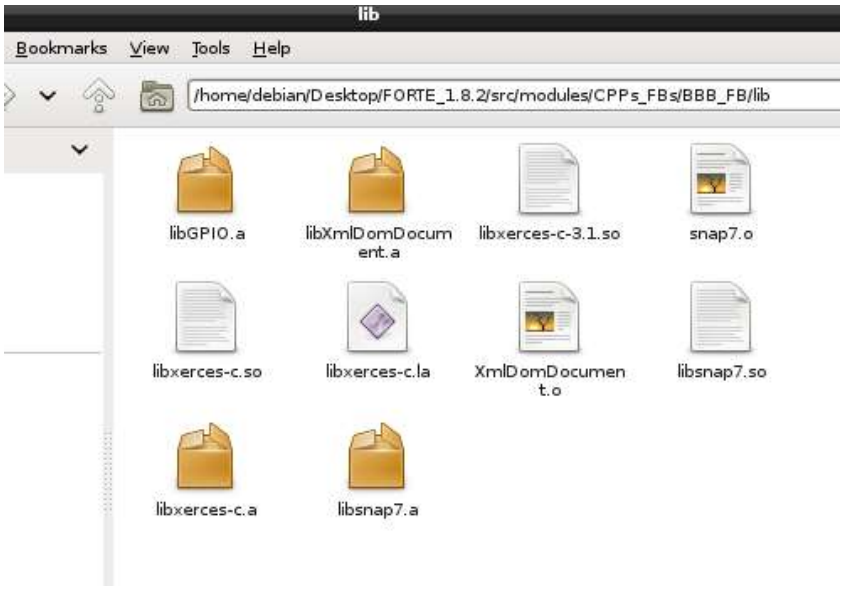



*Figura. 52 Librerías contenidas en la carpeta lib*

Una vez que se tienen los archivos base de FORTE el único requisito restante para poder compilarlos, es tener instalado un compilador gráfico en las distribuciones de Linux de las tarjetas. Para lo cual solo es necesaria la ejecución del comando *apt-get cmake-gui install* en una terminal con privilegios de superusuario. Una vez culminada la instalación del compilador, al dirigirnos al menú de inicio en el apartado *Programming* observaremos el acceso directo a *CMake* como se muestra en Figura. 53.

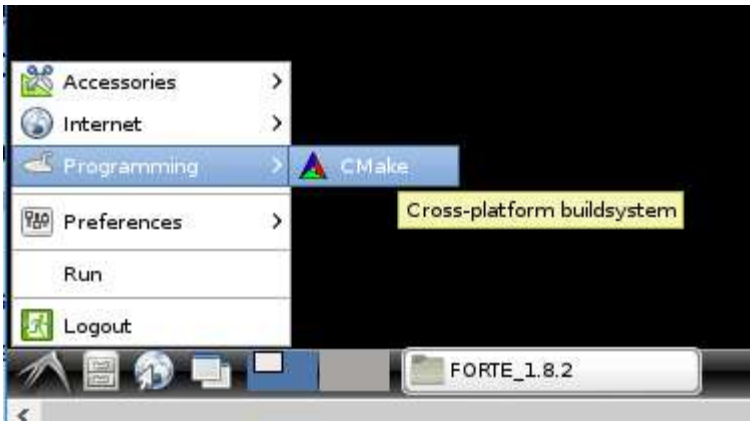



*Figura. 53 Acceso directo a CMake en el menú de inicio de la tarjeta Beaglebone Black*

Al ingresar a *CMake* aparecerá la ventana del compilador gráfico, en donde como primeras tareas se deben definir las direcciones de los archivos a compilar (carpeta contenedora de archivos base de FORTE) y el lugar

donde generarán los binarios de compilación (directorio existente o nuevo, de preferencia ubicado en el escritorio) como se muestra en la Figura. 54.

Al presionar el botón *Generate* aparecerá una ventana pidiendo determinar el generador de proyecto, el cual debe ser seteado en *Unix Makefiles* como se muestra en la Figura. 55 para posteriormente presionar el botón *Finish*. Inmediatamente comenzará la generación de los archivos binarios de compilación, pero se verá truncada por un error en la configuración de proceso como se muestra en la Figura. 56. Para poder solucionar este error en el proceso de configuración se debe seleccionar el botón *OK* del mensaje de error y en las opciones de configuración: 1) cambiar la arquitectura FORTE a *Posix* y 2) seleccionar el módulo añadido con los archivos fuente de los FBs desarrollados para que sea añadido en la generación de binarios como se muestra en la Figura. 57.

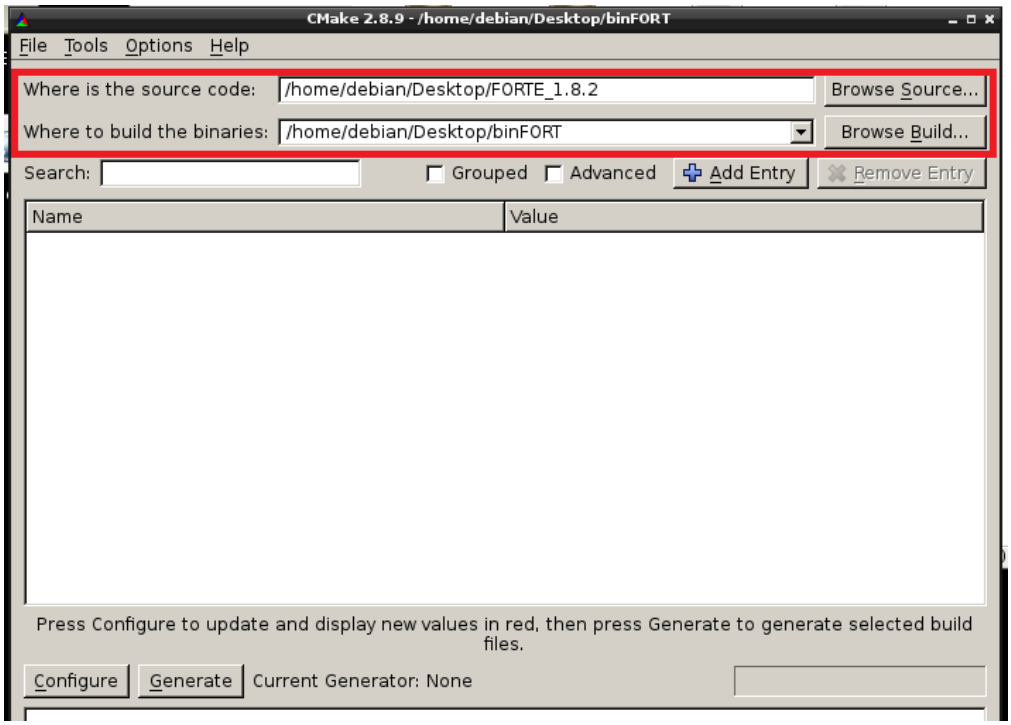



*Figura. 54 Ventana de compilador gráfico CMake con direcciones archivos base y salida de binarios seleccionadas*

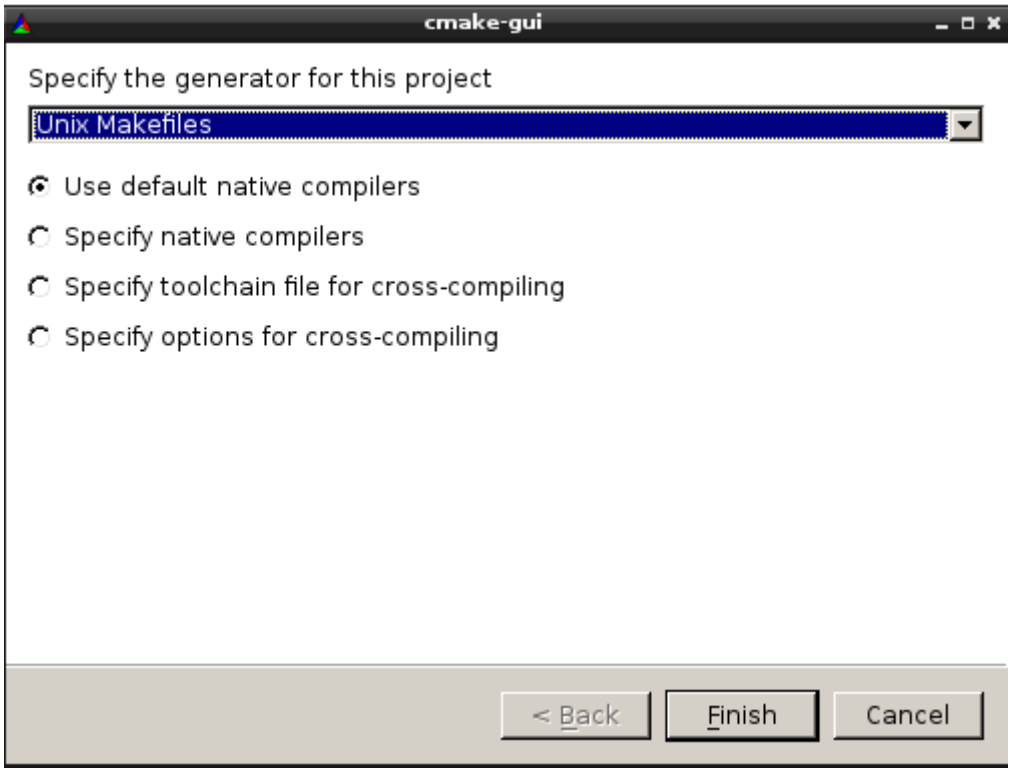



*Figura. 55 Ventana de selección de generador*

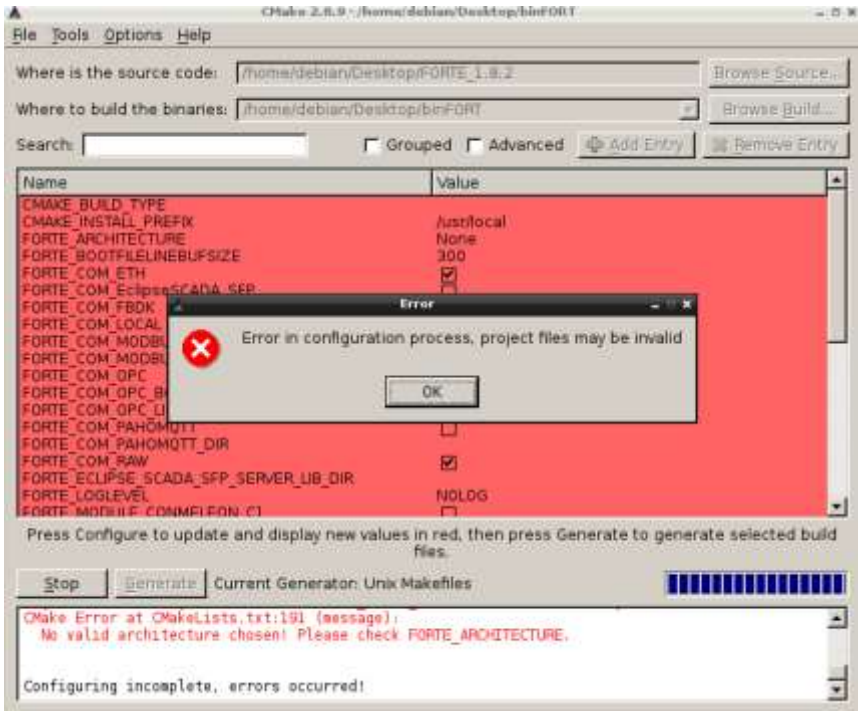



*Figura. 56 Error en proceso de configuración*

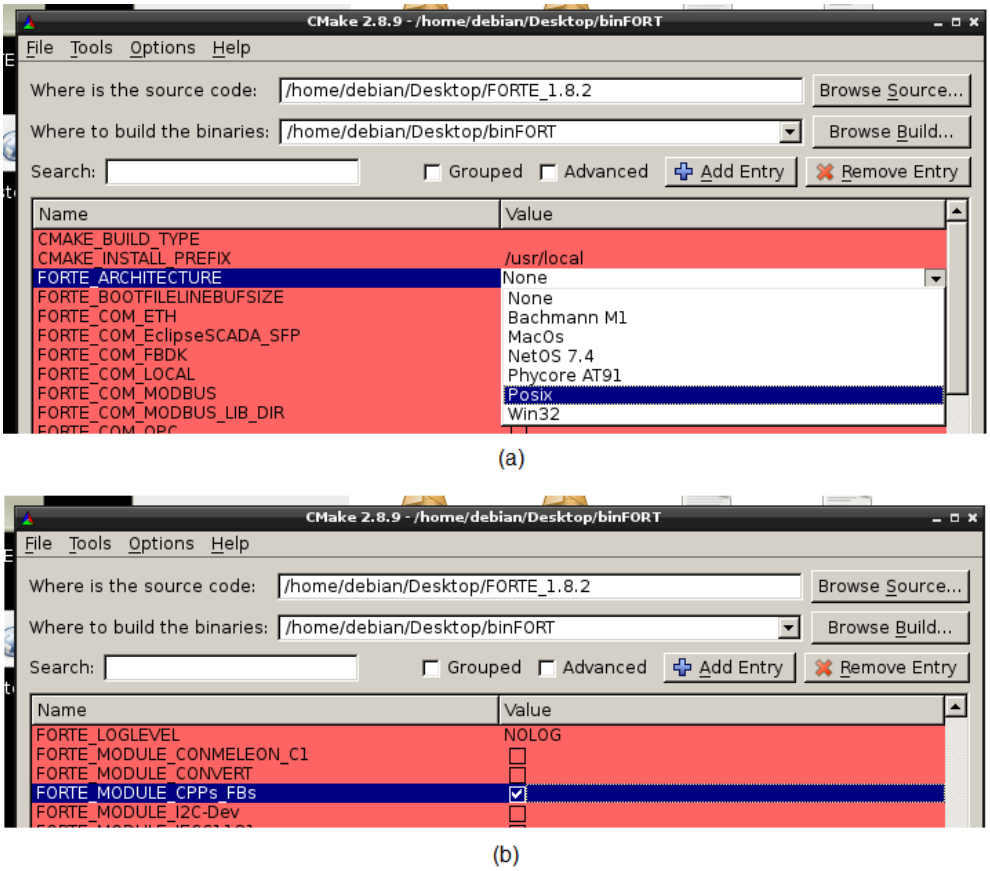



*Figura. 57 Modificaciones necesarias para solucionar el error en proceso de generación (a) Cambio de arquitectura FORTE a Posix (b) Selección de módulo contenedor de archivos fuente de FBs desarrollados*

Con las modificaciones hechas, al presionar de nuevo el botón *Generate* por segunda vez el proceso de generación se completará exitosamente y dentro de la carpeta destinada a contener los binarios de compilación aparecerán dichos archivos como se muestra en la Figura. 58.

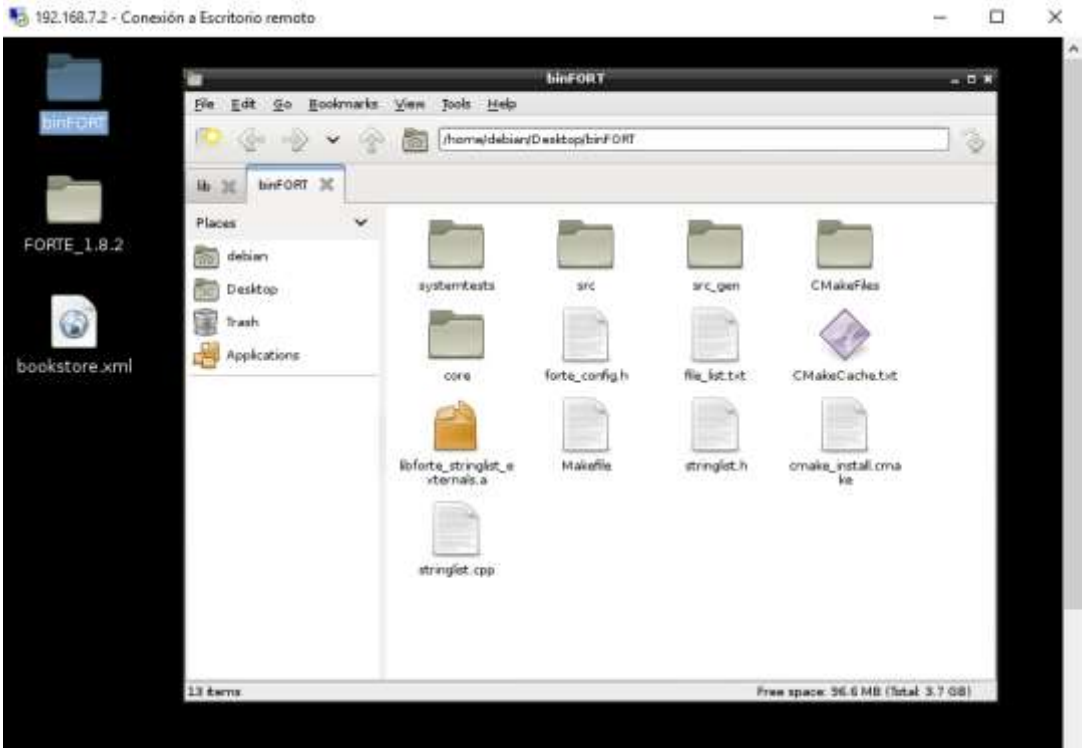



*Figura. 58 Carpeta binFORT con los archivos binarios de compilación generados con CMake*

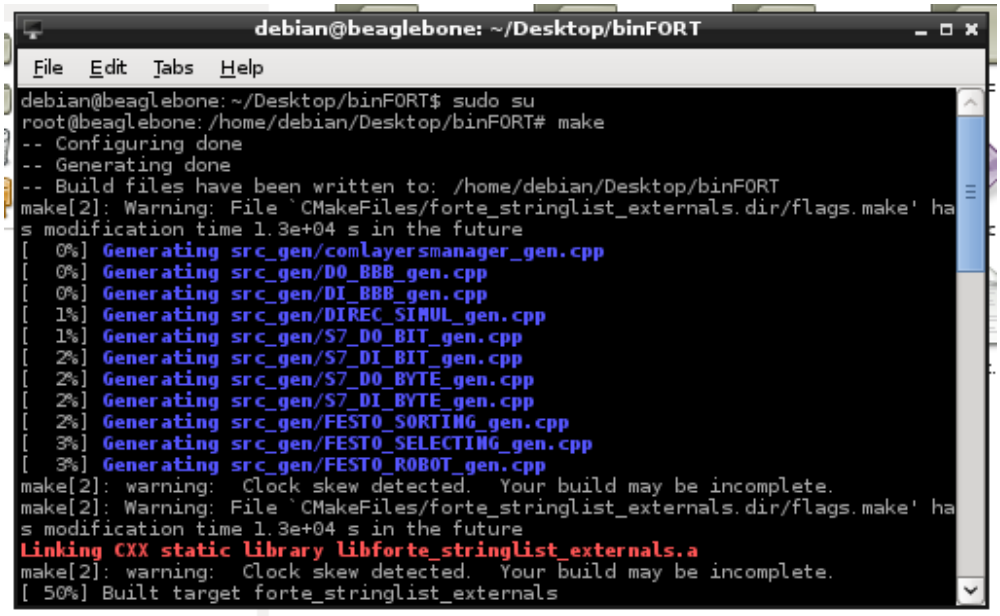



*Figura. 59 Inicio de ejecución de comando make*

Finalmente para poder generar el archivo ejecutable FORTE es necesario abrir un terminal con ubicación en la carpeta contenedora de binarios de compilación, y con permisos de superusuario ejecutar el comando *make* como se muestra en la Figura. 59.

El tiempo de ejecución del proceso de compilación depende en gran cantidad de las capacidades de memoria RAM que posea el dispositivo en el cual se ejecute la acción. En el caso de las tarjetas Raspberry y Beaglebone seleccionadas para este proyecto de investigación la media de ejecución se encontraba alrededor de los 15 minutos gracias a sus buenas capacidades de memoria. Una vez culminada la ejecución exitosa del proceso de compilación se tendrá en la pantalla de terminal una vista como la de la Figura. 60.

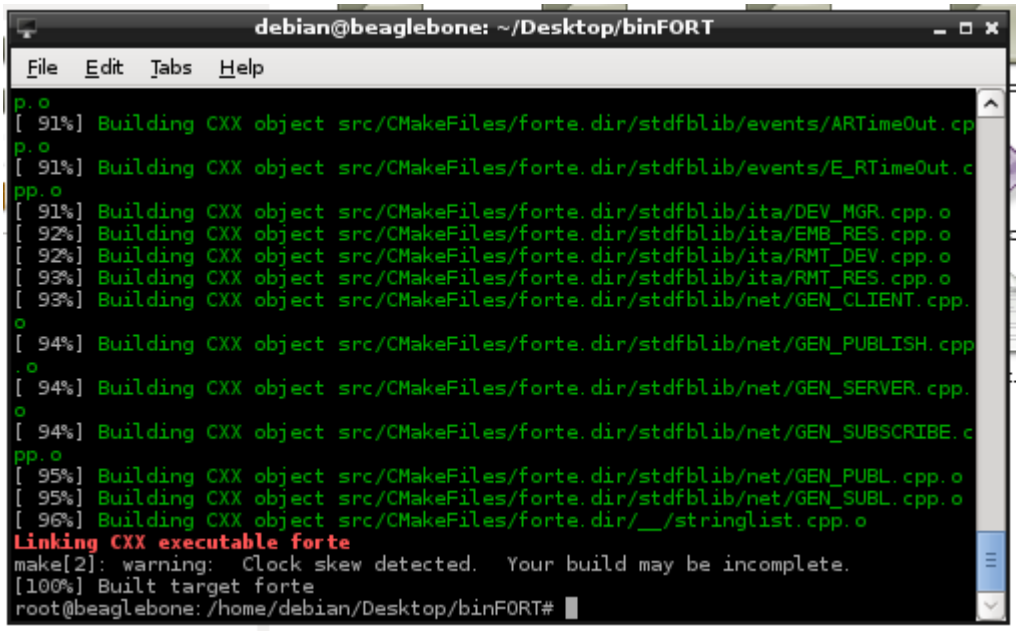



*Figura. 60 Proceso de compilación culminado exitosamente*

## Prueba de FBs

Tras una correcta compilación de la aplicación de FORTE en el dispositivo de control, se pueden probar los FBs de los módulos utilizados en la compilación, para lo cual se debe:

1. En la tarjeta de control abrir un terminal con privilegios de superusuario y ubicarse en la carpeta donde se generaron los binarios de compilación.
2. Ejecutar la aplicación *forte* localizada en la subcarpeta *src* mediante la línea de código *./src/forte.*


*Figura. 61 Sesión de terminal abierta con privilegios de superusuario en la ubicación de los binarios de compilación.*

3. Inmediatamente, dirigirse a 4DIAC-IDE y abrir la pestaña del FB que se desea probar. Como se puede observar en la Figura. 62 en la parte inferior del área de trabajo se puede apreciar un grupo de pestañas, de las cuales se encuentra seleccionada *Interface.* Para poder acceder a la interfaz de prueba es necesario seleccionar la pestaña *FBTester.*

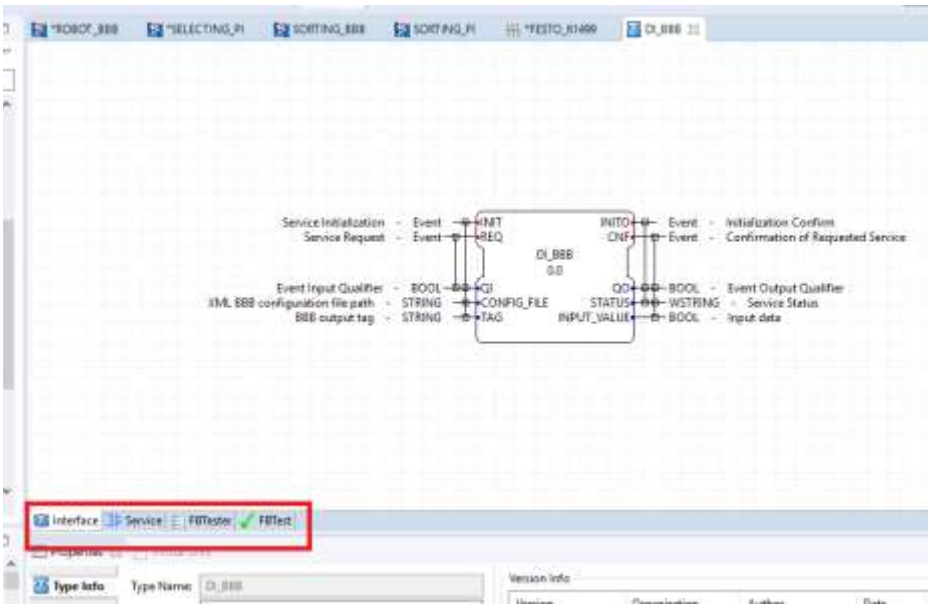



*Figura. 62 Pestaña de trabajo de FB DI_BBB*

4. Una vez seleccionada la pestaña *FBTester,* la interfaz mostrada en la Figura. 63 aparecerá en la pestaña de trabajo; en la cual, es necesario cambiar la configuración de prueba (*Test Configuration*) a *FORTE*

*Remote Tester* para poder desplegar las configuraciones de prueba, como se muestra en la Figura. 64.

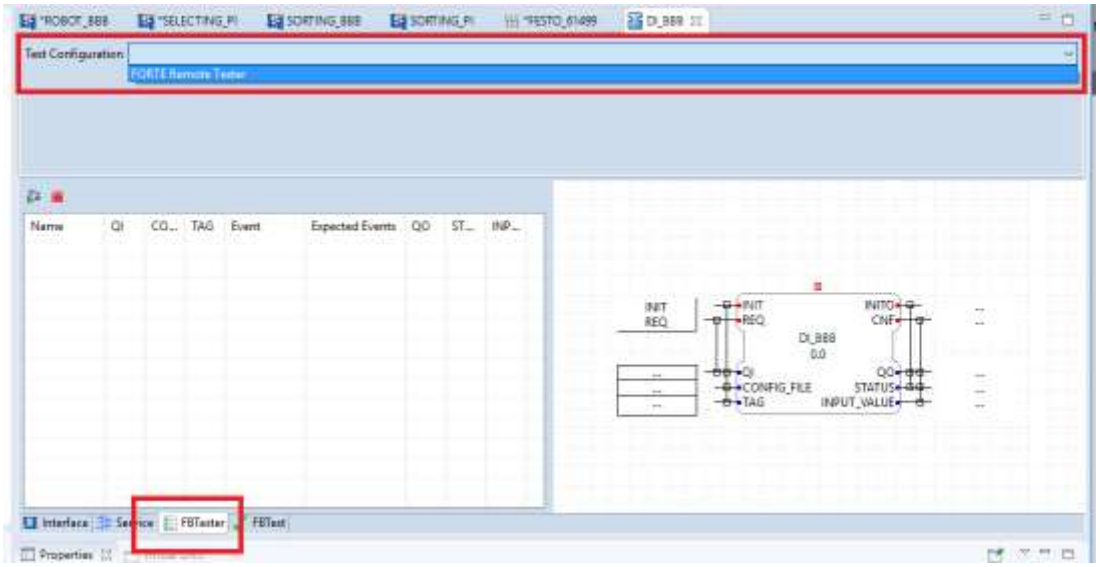



*Figura. 63 Interfaz de prueba de FBs de 4DIAC-IDE*

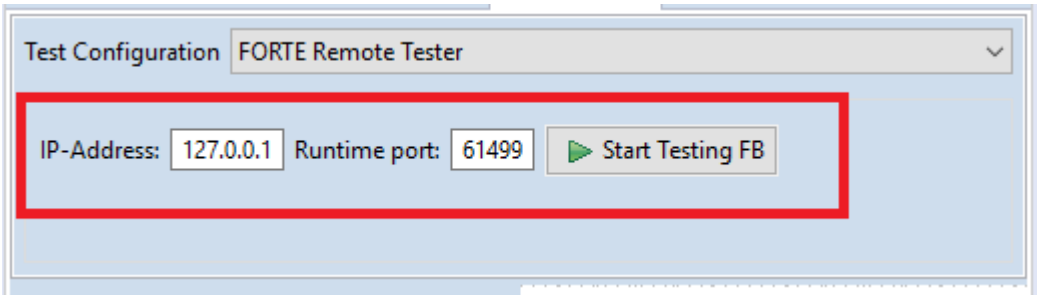



*Figura. 64 Opciones de configuración de prueba habilitadas tras el cambio de la configuración de prueba*

5. En las opciones de configuración habilitadas el único parámetro que se debe modificar es la dirección IP, en donde se debe ingresar la dirección del dispositivo en el cual se desea correr el FB a probar.
6. Una vez ingresada la dirección IP deseada, se debe pulsar el botón *Start Testing FB* para empezar la prueba del FB.

Con la herramienta de prueba en ejecución, se habilita la sección de ejecución manual del FB desplegado en la ventana de prueba. Como se puede apreciar en la Figura. 65, en esta sección el FB aparece con una serie de botones que al ser pulsados simulan la llegada de un evento a sus

entradas, mientras que en las entradas de datos se encuentran un conjunto de cuadros destinados al seteo por teclado de sus respectivos valores.

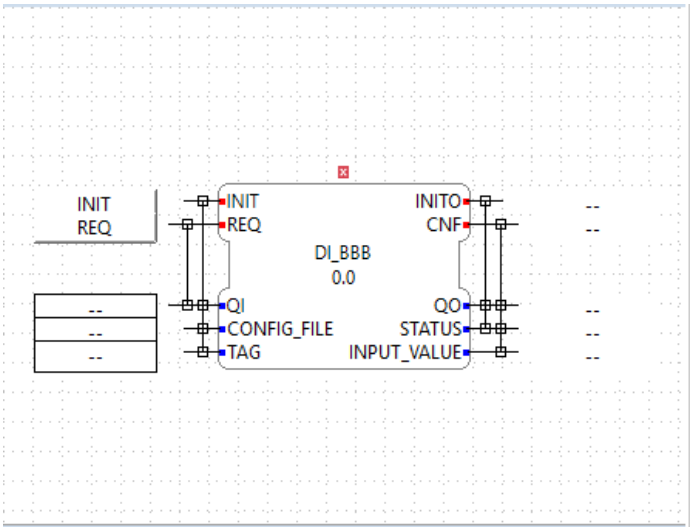



*Figura. 65 Sección de ejecución manual de FB de la ventana de prueba*

Para poder tener un correcto desarrollo de la prueba se debe recordar que la metodología de trabajo de un FB se basa en usar los datos relacionados a un evento cuando a este le llegue su señal de activación, por lo cual es necesario saber la correcta secuencia de ejecución de eventos y sus datos de entrada relacionados para no caer en errores de ejecución.

## 4.3. Propuesta de Software: FBs Desarrollados

### FBs para manejo de GPIO

Debido a que la propia kernel de las tarjetas seleccionadas no permite un manejo directo de entradas y salidas de sus puertos GPIO, fue primordial el desarrollo de un conjunto de FBs que encapsulen la programación necesaria para poder manipular físicamente la lectura y escritura de niveles lógicos de voltaje en los puertos.

A pesar de sus diferencias físicas, las tarjetas Raspberry PI y Beaglebone Black comparten la misma metodología de trabajo para la manipulación de sus puertos GPIO, la cual se encuentra representada en la Figura. 66. Genéricamente en ambas tarjetas para empezar a manipular los pines de sus puertos GPIO se debe encontrar la ubicación del archivo de

mapeo de puertos, en donde se procede a exportar la carpeta de configuración individual del pin a manipular. La carpeta exportada contiene varios archivos en los cuales se almacena la información de sus atributos y pueden ser editados para alterar su funcionamiento.

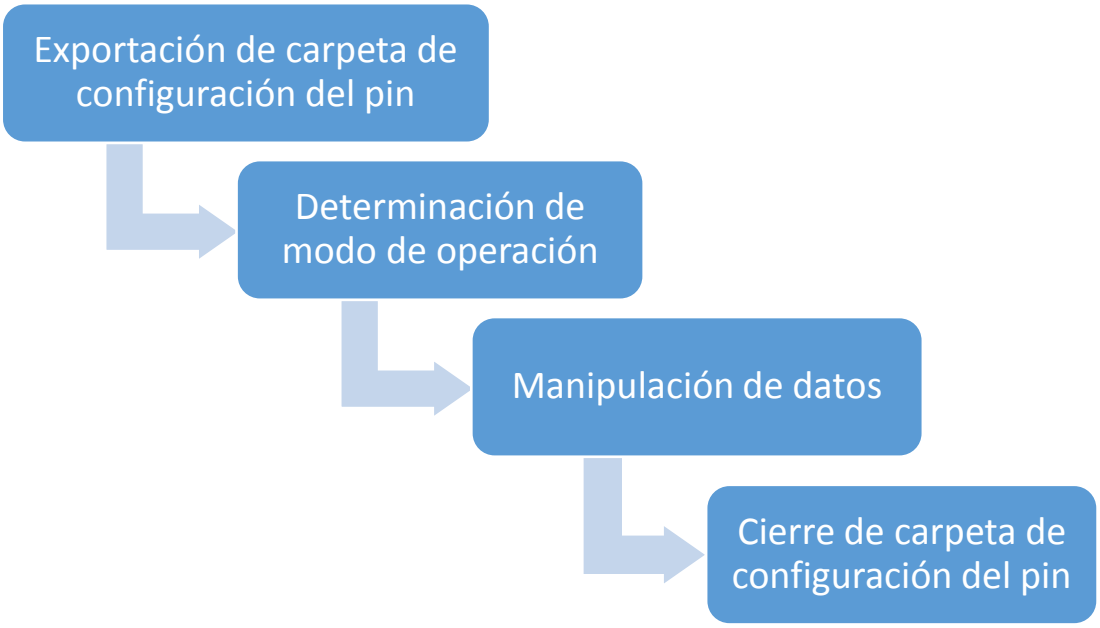
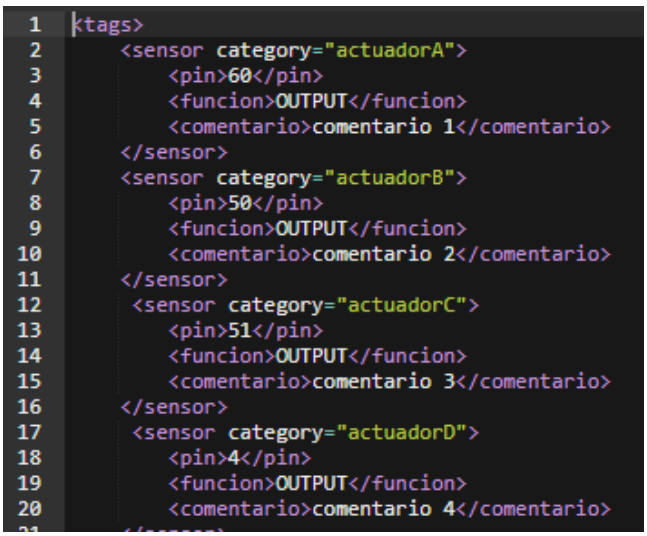



*Figura. 66 Metodología de manipulación de puertos GPIO de las tarjetas Raspberry PI y Beaglebone Black*

Esencialmente los atributos que deben ser alterados para trabajar con la lectura y escritura de niveles lógicos de tensión son: mode (modo de operación), direction (en el caso de trabajar como pin de I/O determina si es entrada o salida) y value (valor lógico generado desde la tarjeta, o ingresado a la misma). Tanto *mode* como *direction* deben ser modificados como parte de las primeras tareas de configuración realizadas antes de empezar con las acciones de lectura y escritura; mientras que *value* debe ser manipulado durante la ejecución de las tareas. Una vez culminadas las transacciones de lectura y escritura se deben cerrar todas las carpetas exportadas y así finalizar correctamente la lectura y escritura.

Los FBs desarrollados encapsulan estos procesos mediante la implementación de librerías dentro de su codificación en C++. Dentro de

internet existe gran cantidad de librerías desarrolladas específicamente para el manejo de puertos GPIO de las tarjetas Raspberry PI y Beaglebone Black, por lo que la selección de una en específico depende del criterio de los programadores. Para los FBs a discutir se utilizaron las librerías de manejo de GPIO desarrolladas por Derek Molloy tanto para Beaglebone como para Raspberry, debido a la gran documentación que entrega el autor y la facilidad existente para modificar el código fuente de las mismas.



```
<tags>
    <sensor category="actuadorA">
        <pin>60</pin>
        <funcion>OUTPUT</funcion>
        <comentario>comentario 1</comentario>
    </sensor>
    <sensor category="actuadorB">
        <pin>50</pin>
        <funcion>OUTPUT</funcion>
        <comentario>comentario 2</comentario>
    </sensor>
     <sensor category="actuadorC">
        <pin>51</pin>
        <funcion>OUTPUT</funcion>
        <comentario>comentario 3</comentario>
    </sensor>
     <sensor category="actuadorD">
        <pin>4</pin>
        <funcion>OUTPUT</funcion>
        <comentario>comentario 4</comentario>
```

*Figura. 67 Archivo de configuración XML de tarjeta Beaglebone Black vista en Cloud9*

En el campo industrial, los dispositivos de control compatibles con IEC-61131 trabajan con archivos en los cuales se almacenan los datos informativos sobre el pin a manipular, optimizando de esta manera la forma en la que se programa. Es por esto que, de igual manera el direccionamiento de entradas y salidas en los FBs desarrollados se lo realiza utilizando el nombre de la variable almacenado en un archivo de configuración XML individual para cada tarjeta, todo esto gracias a la utilización de la librería *Xerces* para manejo de archivos XML en Linux. En dichos archivos se encuentra el listado de: el nombre de la variable, el pin de la tarjeta referido a dicha variable, el modo de funcionamiento de dicho pin (si trabaja como entrada o salida digital), y un comentario informativo sobre su funcionamiento como se muestra en Figura. 67; de esta manera se efectúa un referenciado de forma dinámica acorde a los procedimientos utilizados

actualmente por la mayoría de dispositivos de control compatibles con IEC-61131.

### *DO_BBB & DO_RPI FBs*

Los FBs mostrados en la Figura. 68 fueron generados para configurar los puertos GPIO requeridos como salidas digitales, escribiendo en ellos un valor lógico TRUE o FALSE. Al igual que los FB estándar, estos FBs reciben los eventos INIT y REQ para iniciar sus operaciones, y generan los eventos INITO y CNF. Estos FB funcionan con las siguientes variables de E/S:

- **CONFIG_FILE (Entrada, STRING):** nombre de archivo XML con la información de la asignación de cada tarjeta.
- **TAG (Entrada, STRING):** Nombre del puerto de salida a manipular.
- **OUTPUT_VALUE (Input, BOOL):** Valor booleano que se escribirá en el puerto GPIO.
- **STATUS (Output, WSTRING):** Código de descripción de error siempre que haya uno.

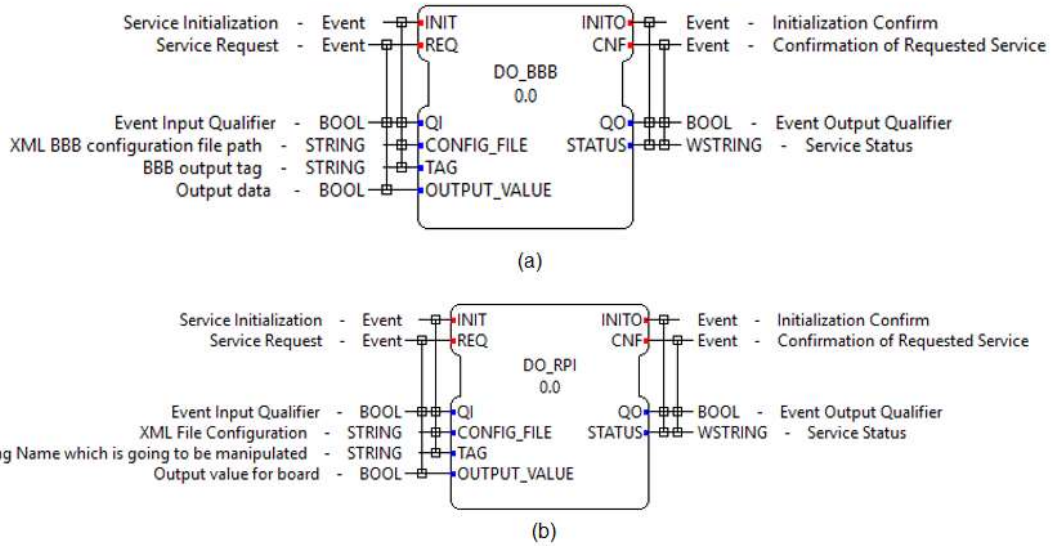



*Figura. 68 (a) DO_BBB FB (b) DO_RPI FB*

### *DI_BBB & DI_RPI FBs*

Los FBs mostrados en la Figura. 69 fueron generados para configurar el GPIO de la tarjeta correspondiente como entrada de datos, leyendo el

valor digital de entrada en el puerto al ser llamado. Al igual que los FB estándar, estos manejan los eventos INIT y REQ para iniciar sus operaciones y generan los eventos INITO y CNF. Estos FBs funcionan con las siguientes variables de E/S:

- **CONFIG_FILE (Entrada, STRING):** nombre de archivo XML con la información de la asignación de cada tarjeta.
- **TAG (Entrada, STRING):** Nombre del puerto de salida a manipular.
- **INPUT_VALUE (Output, BOOL):** Valor booleano leído en el puerto GPIO.
- **STATUS (Output, WSTRING):** Código de descripción de error siempre que haya uno.

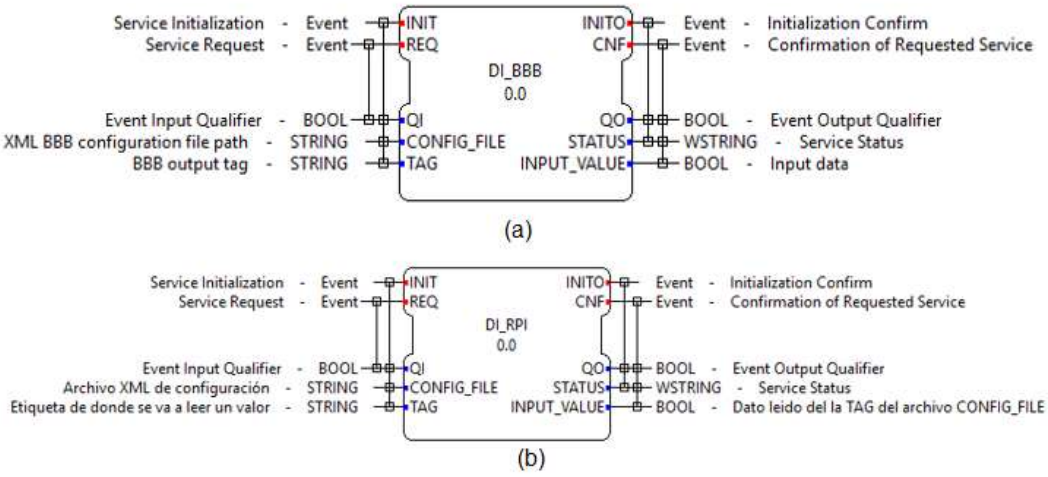



*Figura. 69 (a) DI_BBB FB (b) DI_PRI FB*

## FBs para manipulación S7

Al analizar el funcionamiento de la estación de clasificación, una de las principales complicaciones que se tiene es que su sistema de válvulas trabaja bajo señales de Profibus. Las tarjetas Raspberry PI y Beaglebone no son compatibles físicamente con este protocolo de comunicación, y adicionalmente al no existir un shield que permita la compatibilización del mismo en las tarjetas, se tuvo que buscar otra alternativa para lograr la comunicación entre las tarjetas de desarrollo y el sistema de válvulas Profibus.

Una de las marcas con mayor aceptación industrial y académica es SIEMENS, por lo cual esta casa comercial a lo largo de los años ha venido brindando soluciones de control y automatización a todos sus clientes. Sus dispositivos de control trabajan bajo un protocolo de comunicación desarrollado por la misma casa comercial denominado S7, el cual es considerado la columna vertebral de las comunicaciones de Siemens.

Su implementación se la realiza mediante Ethernet y se basa en ISO TCP (RFC1006), el cual por diseño está orientado a bloques. Cada bloque se denomina PDU (Protocol Data Unit), su longitud máxima depende del CP (Comunication Processors) y se negocia durante la conexión.

Si el tamaño de un comando no cabe en una PDU, entonces debe dividirse en más PDUs. Cada comando consiste en:

- Un encabezado.
- Un conjunto de parámetros.
- Datos de parámetros.
- Un bloque de datos.

Los primeros dos elementos están siempre presentes, los otros son opcionales.

Para una mejor comprensión se presenta el ejemplo a continuación: ***Escribir*** estos ***datos*** en ***DB 10*** comenzando desde el offset ***4.*** Esto se traduce a una línea de comandos con la estructura <<***Escribe, DB, 10, 4 y los datos>>*** que están formateados en un mensaje de acuerdo con las especificaciones del protocolo.

Los protocolos S7, ISO TCP y TCP / IP siguen la regla de encapsulación de que: cada telegrama es la parte de "carga útil" del protocolo subyacente.

Con este entendimiento sobre el funcionamiento del protocolo de comunicaciones S7, el investigador italiano Davide Nardella ha desarrollado una librería compatible con el lenguaje de programación C++ que puede ser instalada en una amplia serie de dispositivos entre los cuales se encuentran

los que funcionan bajo arquitecturas ARM V7 y V8. La librería se llama Snap7 y se la puede encontrar en la página sourceforge oficial del autor https://sourceforge.net/projects/snap7/files/.

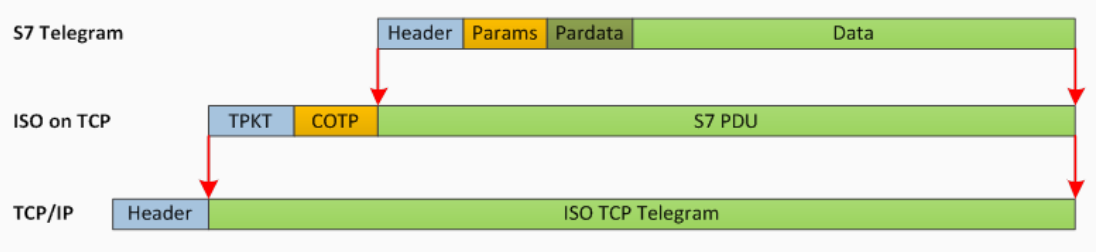



*Figura. 70 Encapsulación de protocolos bajo la regla cada telegrama es la parte de "carga útil" del protocolo subyacente*

Gracias al uso de esta librería se puede superar la problemática de comunicación entre el sistema de válvulas Profibus de la estación de clasificación y las tarjetas de desarrollo, utilizando como intermediario al dispositivo de control SIEMENS que puede comunicarse a través de Profibus con el sistema de válvulas y a través de Ethernet con las tarjetas de desarrollo.

En forma técnica se puede superar el problema mediante la generación de FBs que incluyan en su programación la librería Snap7 para encapsular en su programación el protocolo S7, y de esta manera poder volver al autómata SIEMENS en una extensión adicional de la tarjeta de desarrollo; teniendo como resultado la posibilidad de combinar la robustez física para ámbitos industriales de controladores considerados obsoletos por sus capacidades lógicas, con la potencia computacional de tarjetas de desarrollo sin capacidades industriales.

### *S7_DI_BYTE y S7_DO_BYTE*

Los FBs de la Figura. 71 se diseñaron basados en el protocolo de comunicación S7 de Siemens para poder tener la lectura y escritura de las localidades y bancos de memoria de los autómatas Siemens volviéndolos una extensión del controlador principal. Al igual que los FBs estándar, estos manejan los eventos INIT y REQ para iniciar sus operaciones y generan los

eventos INITO y CNF. Estos FBs funcionan con las siguientes variables de E/S:

- **S7_IP (Entrada, STRING):** Dirección IP del dispositivo Siemens a ser controlado, conectado a la red.
- **Q_BYTE (Entrada, INT):** Número del byte a manipular
- **Q_0 … 7 [S7_DO_BYTE] (Entrada, BOOL):** Valor booleano a escribir en la localidad de memoria.
- **Q_0 … 7 [S7_DI_BYTE] (Salida, BOOL):** Valor booleano a leer de la localidad de memoria.

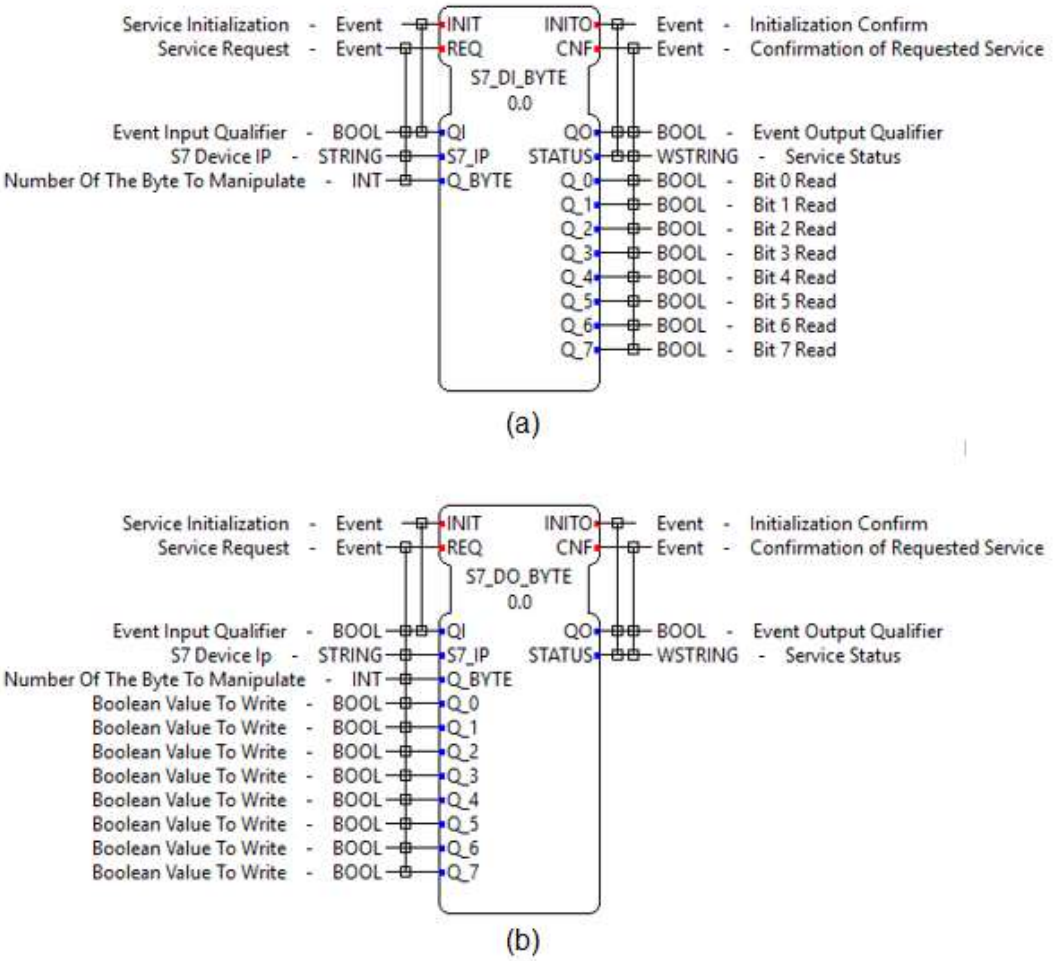



*Figura. 71 (a) S7_DI_BYTE (b) S7_DO_BYTE*

### *S7_DI_BIT y S7_DO_BIT*

A diferencia de sus versiones de manejo de BYTES, los FBs mostrados en la Figura. 72 permiten el manejo de las localidades de memoria BIT A BIT. Al igual que los FB estándar, estos manejan los eventos INIT y REQ para iniciar sus operaciones y generan los eventos INITO y CNF. Estos FBs funcionan con las siguientes variables de E/S:

- **S7_IP (Entrada, STRING):** Dirección IP del dispositivo Siemens a ser controlado, conectado a la red.
- **Q_BYTE (Entrada, INT):** Número del byte a manipular
- **Q_BIT (Entrada, BOOL):** Número del bit a manipular.
- **OUT_VALUE (Entrada, BOOL):** Valor booleano a escribir en la dirección especificada.
- **IN_VALUE (Salida, BOOL):** Valor booleano a leer de la localidad especificada.

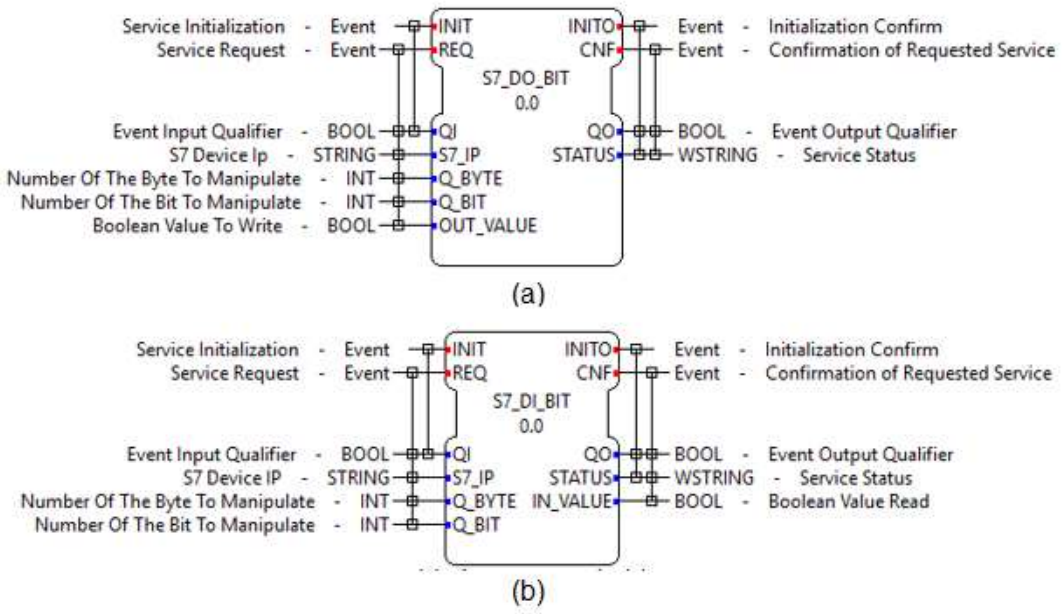



*Figura. 72 (a) S7_DO_BIT (b) S7_DI_BIT*

## FBs de Control Bajo ISA-88

Como se discutió en el apartado ***Estándar ISA-88***, este estándar brinda un modelo de análisis jerárquico para el desarrollo de un sistema de control.

Para el desarrollo de este grupo de FBs se realizó el análisis individual del funcionamiento de cada una de las estaciones para poder determinar sus *funciones* y *operaciones*. De esta manera se facilita el desarrollo de una codificación basada en máquinas de estado, que encapsule las operaciones determinadas en la etapa de análisis.

### *FESTO_SELECTING FB*

Dentro de este FB se encapsularon los algoritmos de control correspondientes al proceso de la estación de selección. Las operaciones resultantes del análisis jerárquico se presentan en la Tabla 7, en la cual se da una descripción de las funciones involucradas que permitieron determinar el número de entradas y salidas del FB al igual que su comportamiento. De esta manera se obtiene un FB que en forma análoga a los FBs estándar recibe los eventos INIT y REQ para iniciar sus operaciones, y genera los eventos INITO y CNF, pero que asimismo funciona con las entradas y salidas mostradas en la Figura.  73.

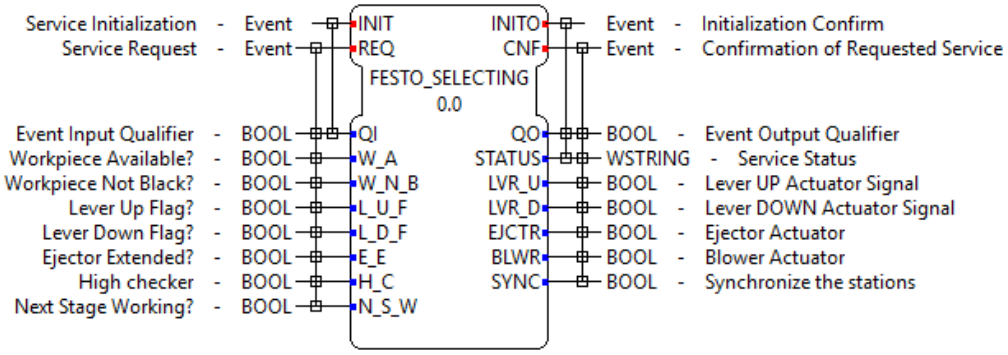



*Figura.  73 FESTO_SELECTING FB*

*Tabla 7*

*Análisis de funciones y operaciones del proceso de la estación de selección*

| Operación | Descripción de fases involucradas |
|---|---|
| **Detección de pieza de trabajo** | Verifica si la siguiente estación se encuentra ocupada (entrada N_S_W) y si existe una pieza de trabajo disponible para hacer la verificación (entrada W_A). |
| **Prueba de color** | En caso de existir una pieza de trabajo, verifica si es de color negro (entrada W_N_B). |
| **Generación de señal de sincronismo** | Tras haberse generado una prueba de color exitosa en la cual se detectó una pieza de trabajo lista para su manipulación, se activa la bandera encargada de informar el estado “ocupado” en el cual se encuentra la estación (salida SYNC). |
| **Elevación de plataforma** | Activa el actuador neumático encargado de elevar la plataforma al nivel superior (salida LVR_U). Desactiva la bandera de control de ubicación de la plataforma en el nivel inferior (salida L_D_F), y activa la bandera de control de ubicación en el nivel superior una vez que la plataforma alcanza este nivel (salida L_U_F). Una vez activada ésta última bandera, se desactiva el actuador encargado de elevar la plataforma (salida LVR_U) |
| **Prueba de tamaño** | Verifica si existen piezas de trabajo con alturas indeseadas para el proceso siguiente (entrada H_C). |
| **Eyección de pieza de** | Activa el actuador neumático encargado de extender el pistón dispensador (salida EJCTR) en conjunto con el |

| | |
|---|---|
| **trabajo** | soplador del andén de recibimiento para facilitar el movimiento de la pieza de trabajo (salida BLWR). |
| **Descenso de la plataforma** | Activa la válvula encargada de liberar la presión de aire encargada de mantener la plataforma en el nivel superior (salida LVR_D). Desactiva la bandera de control de ubicación de la plataforma en el nivel superior (salida L_U_F), y activa la bandera de control de ubicación en el nivel inferior una vez que la plataforma alcanza este nivel (salida L_D_F). Una vez activada ésta última bandera, se desactiva la válvula encargada del descenso de la plataforma (salida LVR_D) |
| **Desactivación de señal de sincronismo** | Tras haber culminado el proceso de manipulación de la pieza de trabajo, se desactiva la bandera encargada de informar el estado “ocupado” en el cual se encuentra la estación (salida SYNC). |

### *FESTO_ROBOT FB*

Dentro de este FB se encapsularon los algoritmos de control correspondientes al robot manipulador de la estación de almacenamiento. Las operaciones resultantes del análisis jerárquico se presentan en la Tabla 8, en la cual se da una descripción de las funciones involucradas que permitieron determinar el número de entradas y salidas del FB al igual que su comportamiento. De esta manera se obtiene un FB que al igual que los FBs estándar recibe los eventos INIT y REQ para iniciar sus operaciones, y genera los eventos INITO y CNF, pero que asimismo funciona con las entradas y salidas mostradas en la Figura.  74.

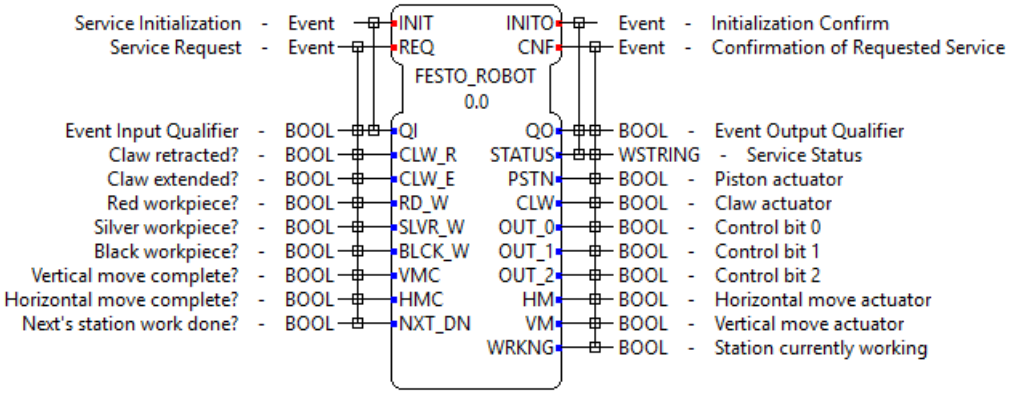



*Figura. 74 FESTO_ROBOT FB*

*Tabla 8*

*Análisis de funciones y operaciones del proceso de la estación de almacenamiento*

| Operación | Descripción de fases involucradas |
|---|---|
| **Prueba de color** | Verifica si la siguiente estación se encuentra ocupada (entrada NXT_DN) y si existe una pieza de trabajo disponible mediante la identificación del color por visión del recibidor (entradas RD_W, SLVR_W y BLCK_W). |
| **Generación de señal de sincronismo** | Tras haberse generado una prueba de color exitosa en la cual se detectó una pieza de trabajo lista para su manipulación, se activa la bandera encargada de informar el estado “ocupado” en el cual se encuentra la estación (salida WRKNG). |
| **Movimiento vertical** | Genera el valor binario que codifica una de las alturas de trabajo (salidas OUT_0, OUT_1, OUT_2), y genera el impulso designado para la ejecución del movimiento vertical (salida VM). |

Continúa

| | |
|---|---|
| **Captura de pieza de trabajo** | Envía la señal necesaria para activar el actuador encargado de capturar la pieza de trabajo (salida CLW), y la señal encargada de retraer el pistón neumático (salida PSTN). |
| **Movimiento horizontal** | Genera el valor binario que codifica una de las posiciones horizontales de trabajo (salidas OUT_0, OUT_1, OUT_2), y genera el impulso designado para la ejecución del movimiento vertical (salida HM). |
| **Liberación de pieza de trabajo** | Envía la señal necesaria para extender el pistón neumático (salida PSTN), y la señal delegada a desactivar el actuador encargado de capturar la pieza de trabajo (salida CLW). |
| **Desactivación de señal de sincronismo** | Tras haber culminado el proceso de manipulación de la pieza de trabajo, se desactiva la bandera encargada de informar el estado “ocupado” en el cual se encuentra la estación (salida WRKNG). |

### *FESTO_SORTING FB*

Dentro de este FB se encapsularon los algoritmos de control correspondientes a la estación de clasificación. Las operaciones resultantes del análisis jerárquico se presentan en la Tabla 9, en la cual se da una descripción de las funciones involucradas que permitieron determinar el número de entradas y salidas del FB al igual que su comportamiento. De esta manera se obtiene un FB que al igual que los FBs estándar recibe los eventos INIT y REQ para iniciar sus operaciones, y genera los eventos INITO y CNF, pero que asimismo funciona con las entradas y salidas mostradas en la Figura.  75.

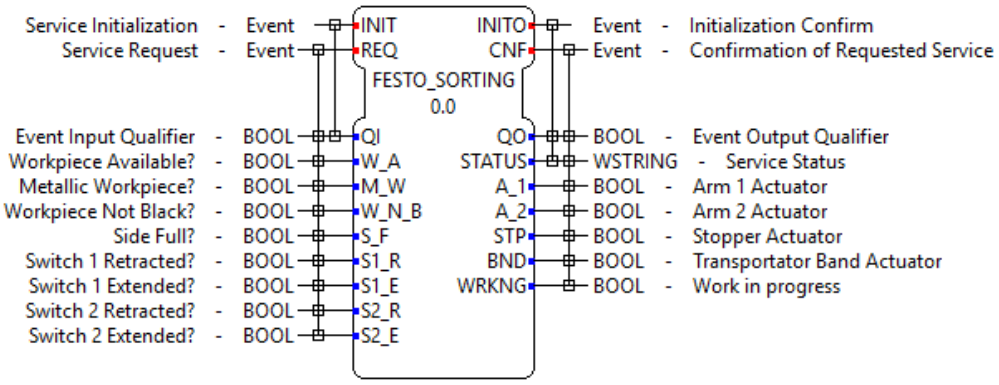



*Figura. 75 FESTO_SORTING FB*

*Tabla 9*

*Análisis de funciones y operaciones del proceso de la estación de separación*

| Operación | Descripción de fases involucradas |
|---|---|
| **Detección de pieza de trabajo** | Verifica si existe una pieza de trabajo disponible para hacer la clasificación (entrada W_A). |
| **Inicio de prueba** | Tras haber detectado una pieza de trabajo lista para su clasificación, se activa la bandera encargada de informar el estado “ocupado” en el cual se encuentra la estación (salida WRKNG), y se activa la banda transportadora (salida BND) durante un periodo de tiempo de 8 segundos. |
| **Selección de camino** | Lee el valor de los sensores infrarrojo (entrada W_N_B) e inductivo (M_W) para determinar si se trata de una pieza de color: roja, negra o plateada. |
| **Liberación de pieza de** | En base a la detección de color generada, activa uno de los dos obturadores encargados de desviar la trayectoria |

| | |
|---|---|
| **trabajo** | de la pieza de trabajo en base a su color (salidas A_1 y A_2). De igual manera activa la bandera encargada de informar el estado extendido del obturador (salidas S1_E o S2_E), en paralelo con la desactivación de la bandera encargada de informar el estado retraído del obturador previamente manipulado (salidas S1_R o S2_R). Tras estas acciones de control se desactiva el actuador del stopper encargado de limitar el paso de la pieza de trabajo (salida STP). |
| **Vuelta a estado inicial** | Una vez que una pieza de trabajo cruce el haz de luz infrarroja correspondiente al sensor encargado de verificar el paso por los rieles de separación (entrada S_F), se producen las siguientes acciones en paralelo: activación del stopper encargado de limitar la circulación de las piezas (salida STP), desactivación de la banda transportadora (salida BND), alternación de las banderas indicadoras de la posición de los obturadores (salidas S1_R, S1_E, S2_R y S2_E) y la desactivación de la bandera que indica el estado de “ocupado” de la estación (salida WRKNG). |

## 4.4. Propuesta de Hardware: Tarjeta de Acople

En vista de la imposibilidad de las tarjetas Raspberry Pi y Beaglebone de ser utilizadas directamente en aplicaciones industriales debido a sus limitaciones estructurales, es necesario desarrollar dos tarjetas de acople que sirvan como puente de conexión entre los circuitos de control y de potencia.

El diseño de las dos tarjetas de control se enfocó en la necesidad de convertir los niveles de voltaje pequeños de las tarjetas (3.3 V) en voltajes de

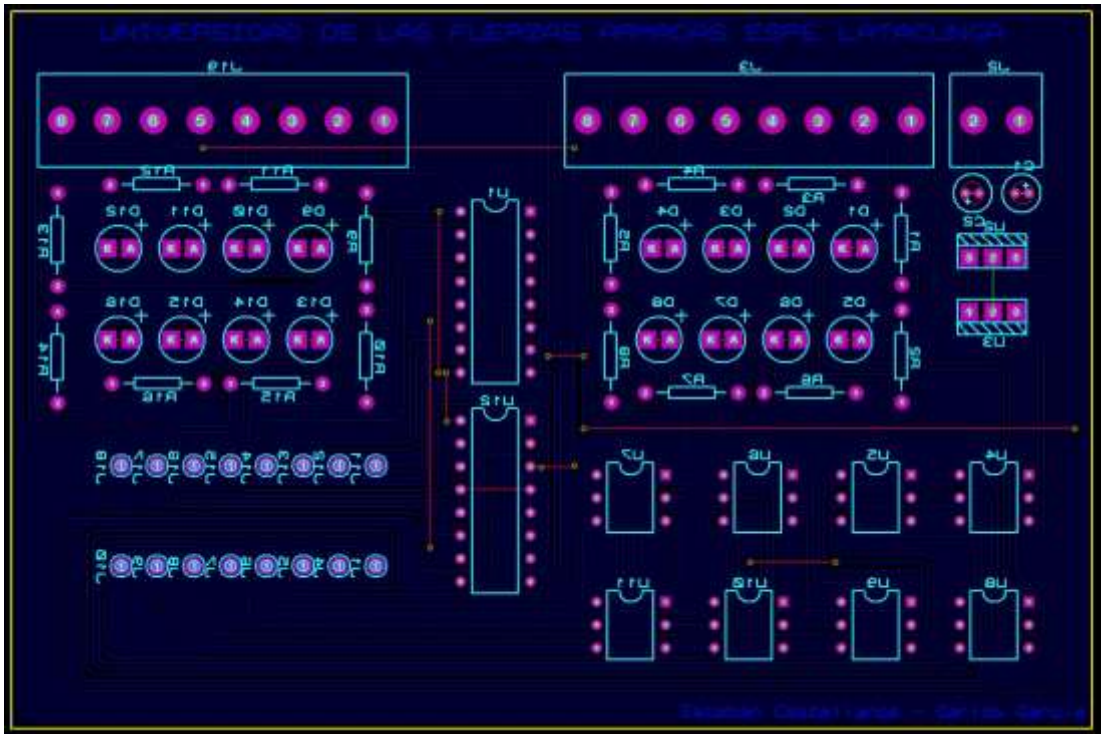
mayor magnitud utilizados generalmente en la industria (24V) y viceversa. Para esto, se generó un circuito sencillo basado en el uso de drivers de motores L293D y optotransistores 4N25, los cuales brindan los aislamientos necesarios para asegurar la protección individual de las tarjetas de control.

Como se puede observar en el diagrama esquemático de la Figura. 77, los materiales utilizados para la creación de cada tarjeta de acople son:

- 1 Regulador de voltaje a 5V LM-7805
- 1 Regulador de voltaje a 3V LM-1904
- 8 Opto transistores 4N25
- 8 Resistencias de 1.2[KΩ] a ½W
- 16 Resistencias de 10[KΩ] a ½W
- 16 LEDs
- 2 Drivers cuádruples L293D
- 2 capacitores electrolíticos de 100nF

Los cuales son distribuidos en la baquelita de acuerdo al esquema de la Figura. 76.

(a)



(b)

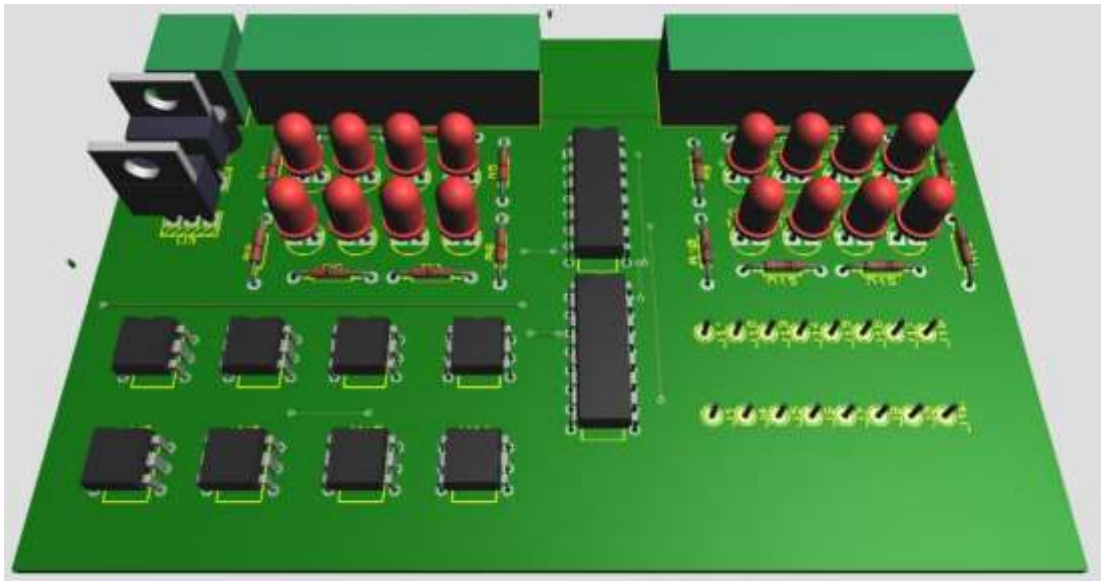


*Figura. 76 (a) Vista de conexiones inferiores (b) Diagrama de distribución de elementos en baquelita.*

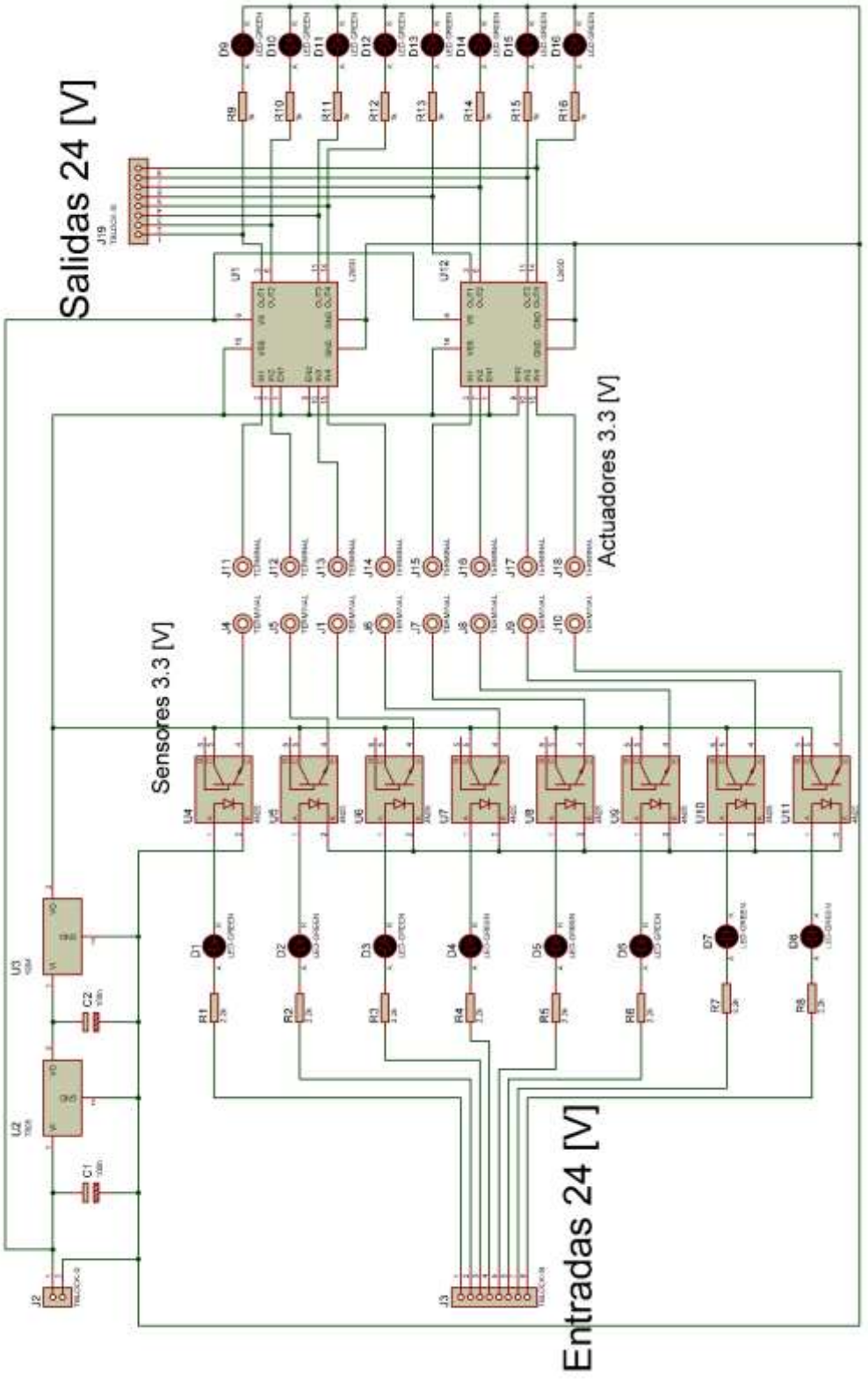



*Figura. 77 Esquema de distribución de tarjetas de acople*

# CAPÍTULO V

# 5. Implementación de Sistema & Resultados

## 5.1. Generación de un sistema en 4DIAC-IDE

### Creación de aplicaciones

Una vez desarrollados todos los FBs necesarios junto a sus respectivos códigos fuente, el proceso para generar aplicaciones funcionales se basa en una programación gráfica básica, en donde se deben interconectar las diferentes entradas y salidas de los bloques a través de cables de conexión.

La metodología inicial de generación de aplicaciones en 4DIAC-IDE es bastante similar a la que se indicó en el apartado ***Exportación de FBs***, en donde se debe:

1. En el apartado del explorador de sistemas, desplegar las opciones del sistema en el cual se desee crear una nueva aplicación. En las propiedades que se despliegan seleccionar *New* y finalmente *New Application* como se muestra en la Figura. 78.

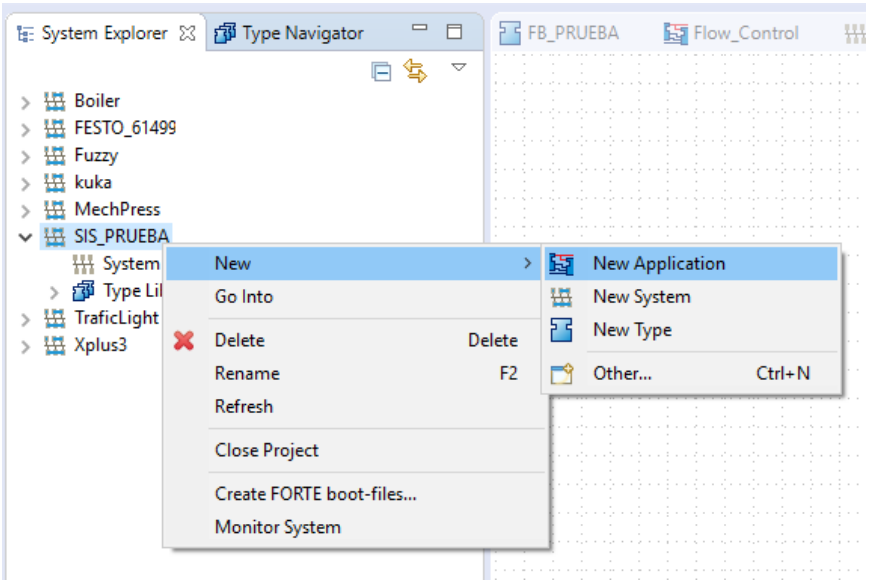



*Figura. 78 Generación de una nueva aplicación en el sistema SIS_PRUEBA*

2. En la ventana emergente, dar un nombre a la nueva aplicación en proceso de creación como se muestra en la Figura. 79, tras lo cual se debe seleccionar *Finish*. Se recomienda que, al momento de nombrar la nueva aplicación se verifique que el sistema correcto se encuentre marcado en la ventana.

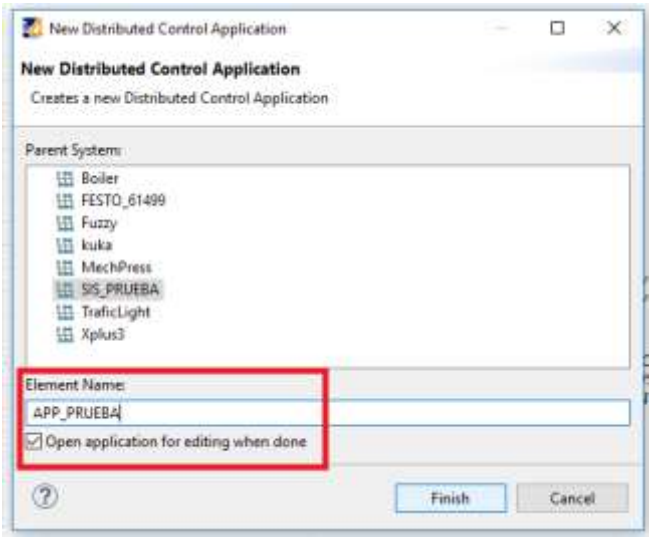



*Figura. 79 Nombramiento de una nueva aplicación a través del asistente de creación de aplicaciones*

Una vez culminado el último paso indicado, automáticamente aparecerá la nueva aplicación creada en el explorador de sistema como se muestra en la Figura. 80. De igual manera, una pestaña que corresponde a la nueva aplicación generada se abrirá en el espacio de trabajo como se muestra en la Figura. 81.

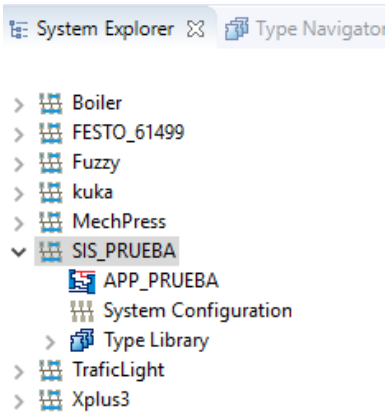



*Figura. 80 Nueva aplicación APP_PRUEBA vista en el árbol del sistema SIS_PRUEBA*

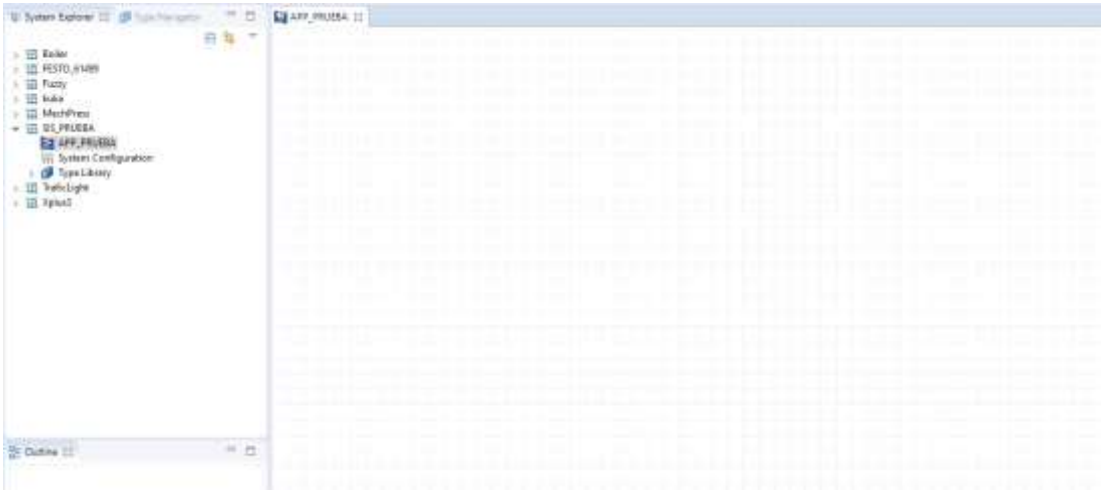


*Figura. 81 Pestaña de aplicación APP_PRUEBA lista para su manipulación en el espacio de trabajo*

Al igual que en todos los programas que utilizan el lenguaje de programación gráfico, la metodología de programación en la nueva pestaña abierta se basa en: 1) arrastrar las funciones disponibles desde la carpeta de funciones hasta el área de trabajo y 2) interconectar las diferentes entradas y salidas con cables generados con el puntero del mouse.

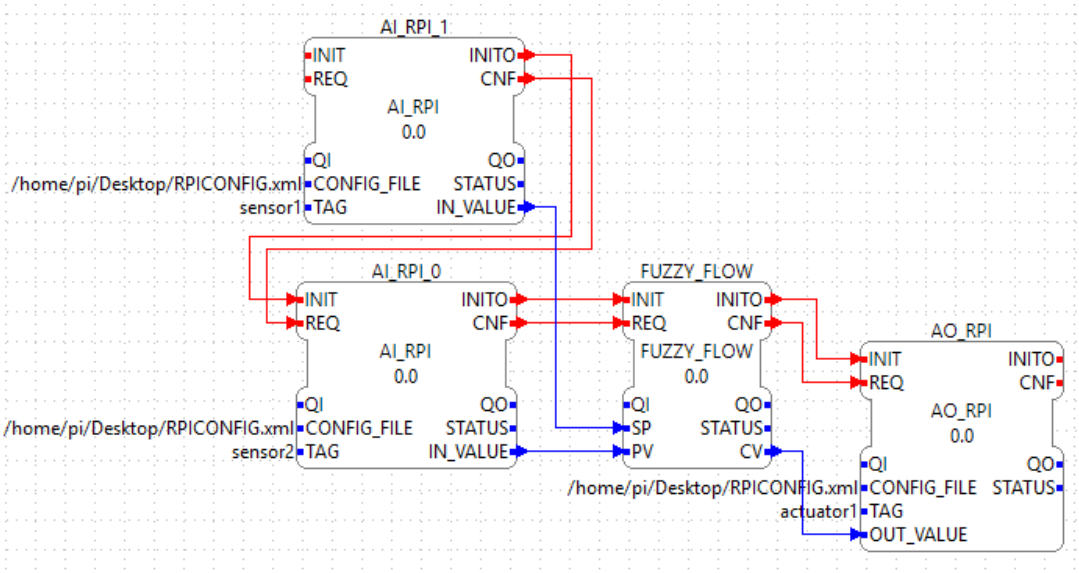



*Figura. 82 Aplicación de control generada en base a la interconexión de FBs de: lectura, control y escritura*

Cuando se habla de la generación de una aplicación de control se deben tener en cuenta tres tipos de elementos para su estructura: los encargados de la lectura de datos, los que encapsulan los algoritmos de

control y los que se destinan a escribir las acciones de control resultantes. Al interconectar correctamente los elementos previamente mencionados en función a su flujo de datos se obtiene una estructura similar a la mostrada en la Figura. 82, la cual viene a ser una aplicación que aún necesita ser relacionada al dispositivo controlador de un sistema para poder ser funcional.

## Configuración de un sistema

En el apartado ***Exportación de FBs*** se presentó la metodología necesaria para crear un nuevo sistema en 4DIAC-IDE, en donde se pueden hacer las primeras configuraciones de FBs conjuntamente con el desarrollo base de aplicaciones. Como pasos finales para obtener un sistema funcional es necesario:

1. En el explorador de sistema expandir los componentes del sistema a configurar y elegir *System Configuration* para abrir, como una nueva pestaña, el espacio de trabajo de configuración del sistema como se muestra en la Figura. 83.

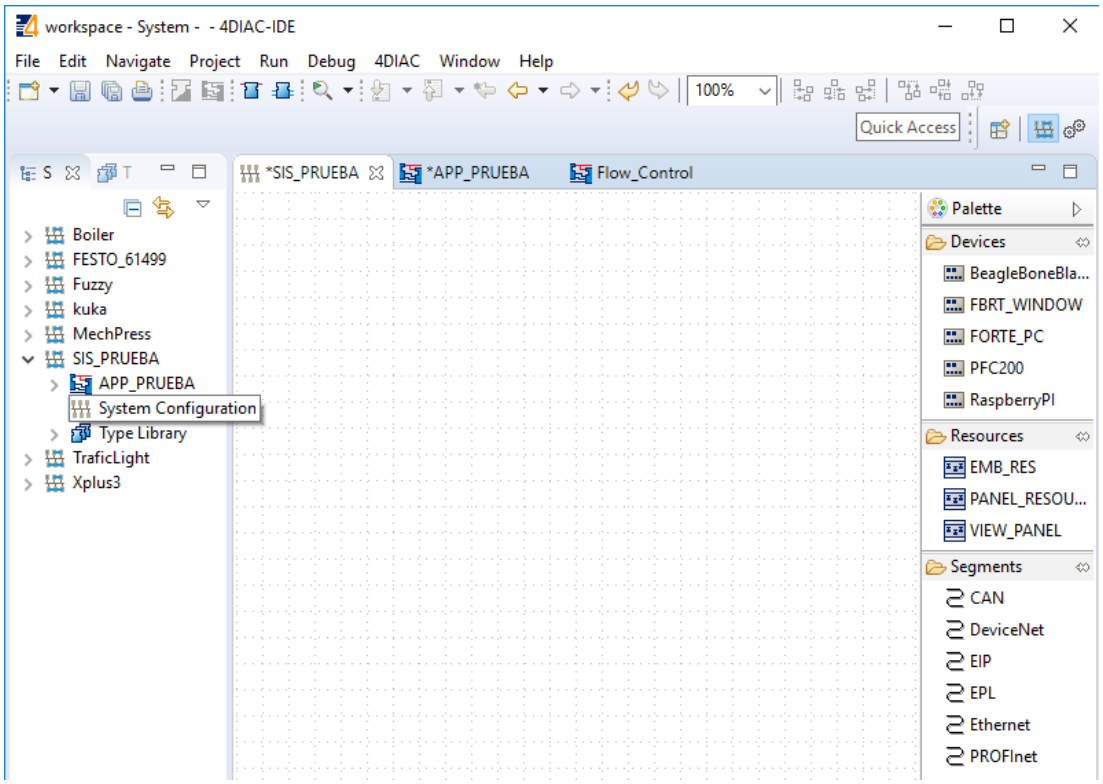



*Figura. 83 Espacio de trabajo de configuración de sistema*

2. A continuación, es necesario ingresar la configuración de red física del sistema distribuido en el cual será descargado el sistema lógico en desarrollo. Para esto, con la ayuda de los componentes disponibles a la derecha de la ventana se debe: 1) de la carpeta *Segments* arrastrar el bus correspondiente al protocolo usado físicamente para interconectar los dispositivos y 2) de la carpeta *Devices* arrastrar los dispositivos involucrados en la distribución física del sistema hasta obtener una distribución como se muestra en la Figura. 84.

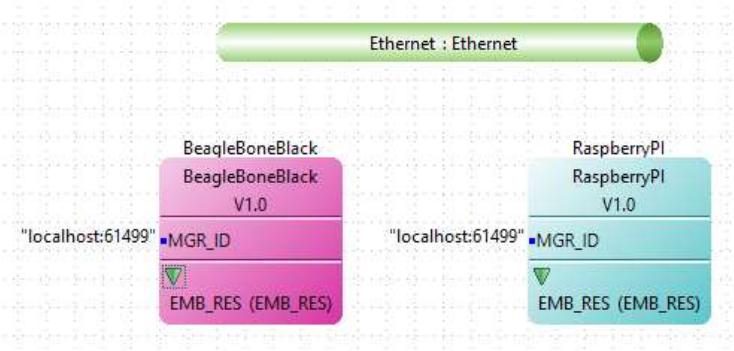



*Figura. 84 Representación de la disposición física del sistema implementado en el espacio de trabajo de configuración del sistema aun sin conexiones*

3. Ingresar el identificador de red utilizado en cada uno de los dispositivos (configurado como IP estática) reemplazando la palabra *localhost* en la entrada *MGR_ID;* y de la misma manera en la que se procedió con las aplicaciones, con la ayuda del puntero del mouse realizar las respectivas conexiones de los dispositivos al bus de comunicaciones para obtener un resultado como el que se muestra en la Figura. 85.

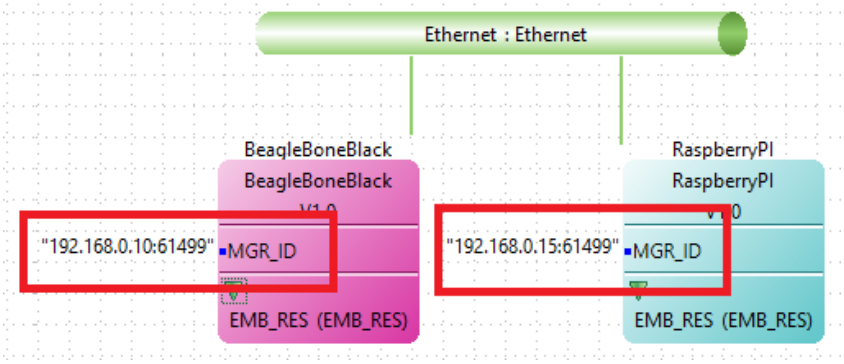



*Figura. 85 Asignación de IDs de red a dispositivos y generación de interconexiones*

4. Como requisito final para poder implementar la programación generada en los dispositivos de control, es necesario mapear las aplicaciones a sus respectivos dispositivos. Para esto, en la ventana de desarrollo de la aplicación se deben marcar todos sus FBs estructurales, desplegar sus opciones y seleccionar *Hardware Mapping* como se muestra en la Figura. 86. El número de elementos que se desplieguen de la acción anterior, dependerá específicamente del número de dispositivos especificados en la configuración del sistema; pero se debe seleccionar

uno de ellos para indicar el destino en donde será descargada la aplicación. Tras la selección del destino de mapeo, los FBs pasarán a tener el color con el que el dispositivo estaba representado en la ventana de configuración del sistema como se ven en la Figura.  87.

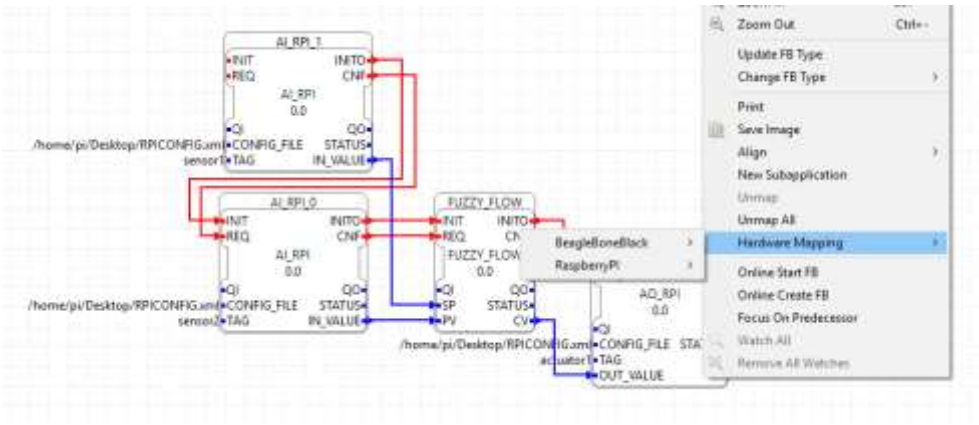



*Figura.  86 Opciones de mapeo para aplicación relacionadas a los dos dispositivos especificados en la configuración del sistema.*

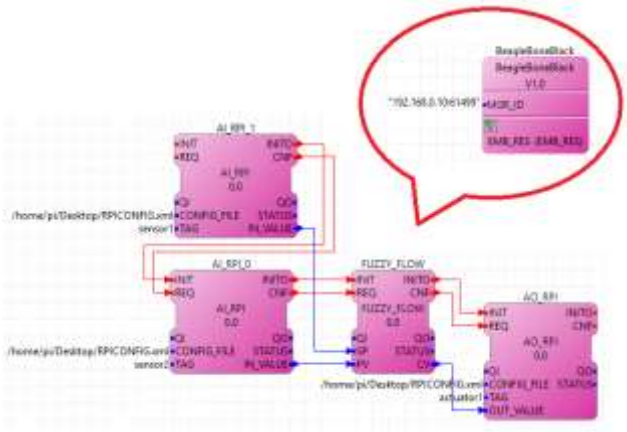


*Figura.  87 Aplicación mapeada a dispositivo BeagleboneBlack, con FBs del mismo color que la representación del dispositivo en la ventana de configuración de sistema*

## Descarga del sistema

Para poder descargar un sistema completo o sus aplicaciones de manera individual se debe utilizar la herramienta *Deployment* de 4DIAC-IDE, a la cual se accede con la ayuda del botón representado por dos engranajes que se encuentra ubicado en la esquina superior derecha de la ventana del programa.


*Figura. 88 Representación y ubicación del botón Deployment en 4DIAC-IDE*

Al presionar el botón *Deployment* aparecerá la ventana de despliegue de aplicaciones de 4DIAC-IDE como se muestra en la Figura. 89, en la cual se pueden apreciar sus dos componentes principales: 1) la sección de descarga ubicada en el lado izquierdo de la ventana, en donde se encuentran todas las aplicaciones disponibles para la descarga y 2) la consola de despliegue ubicada en el lado derecho de la ventana, en donde se indicarán mensajes relacionados al proceso de descarga.

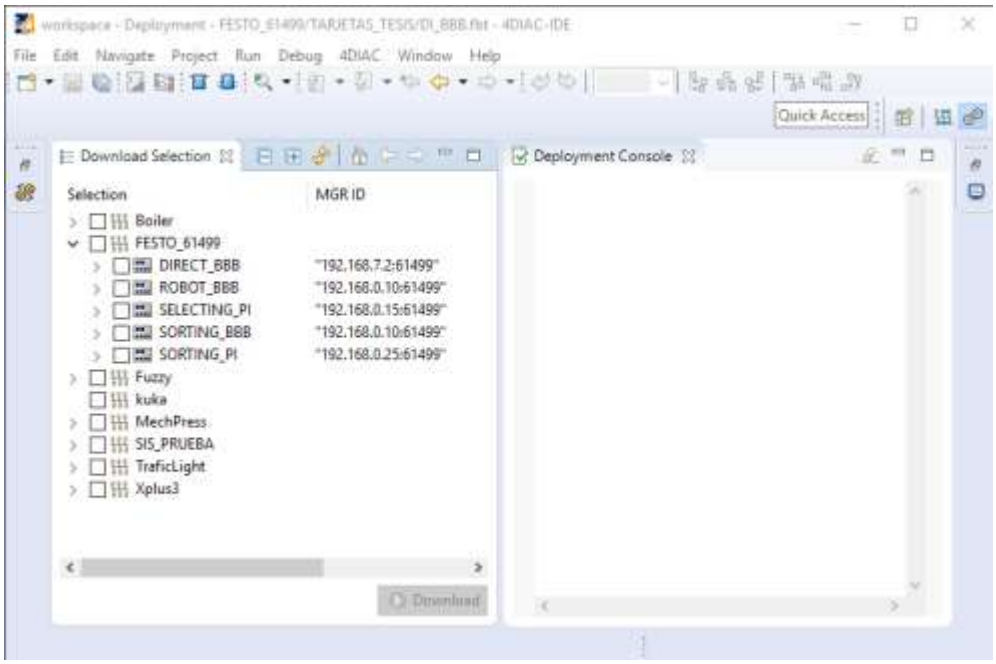



*Figura. 89 Ventana de despliegue de sistemas de 4DIAC-IDE*

Al igual que se hizo con la prueba de FBs individuales, para poder descargar un sistema o aplicación individual es necesario que la aplicación FORTE se encuentre en ejecución; por lo cual, como primer paso del proceso de descarga es necesario repetir la metodología presentada en el apartado

***Prueba de*** FBs para ejecutar la aplicación *FORTE* a través de un terminal.

Con la aplicación en ejecución, la siguiente parte de la descarga del sistema se lleva a cabo en la ventana *Deployment* de 4DIAC-IDE. Ya sea el caso de que se requiera la descarga de un sistema completo o bien solo una de sus aplicaciones en los dispositivos, esto se lo determina en la sección de descarga mediante la activación de las casillas relacionadas a los sistemas y aplicaciones. Al momento de presionar el botón *Download* la herramienta de despliegue automáticamente descargará en las direcciones de red especificadas toda configuración cuya casilla se encuentre marcada, iniciando de forma inmediata la ejecución de la programación involucrada.

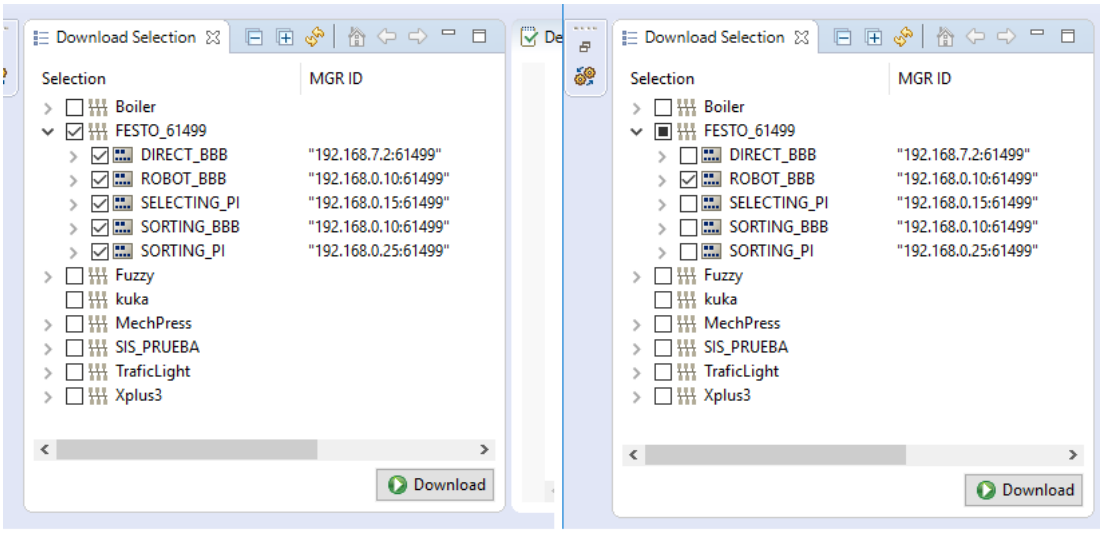



*Figura. 90 (a) Selección de un sistema completo para descarga (b) Selección de una aplicación individual para descarga*

## 5.2. Resultados

La distribución física final del sistema de control distribuido desarrollado se muestra en la Figura. 91, la cual se basa en el uso de una red Ethernet para interconectar:

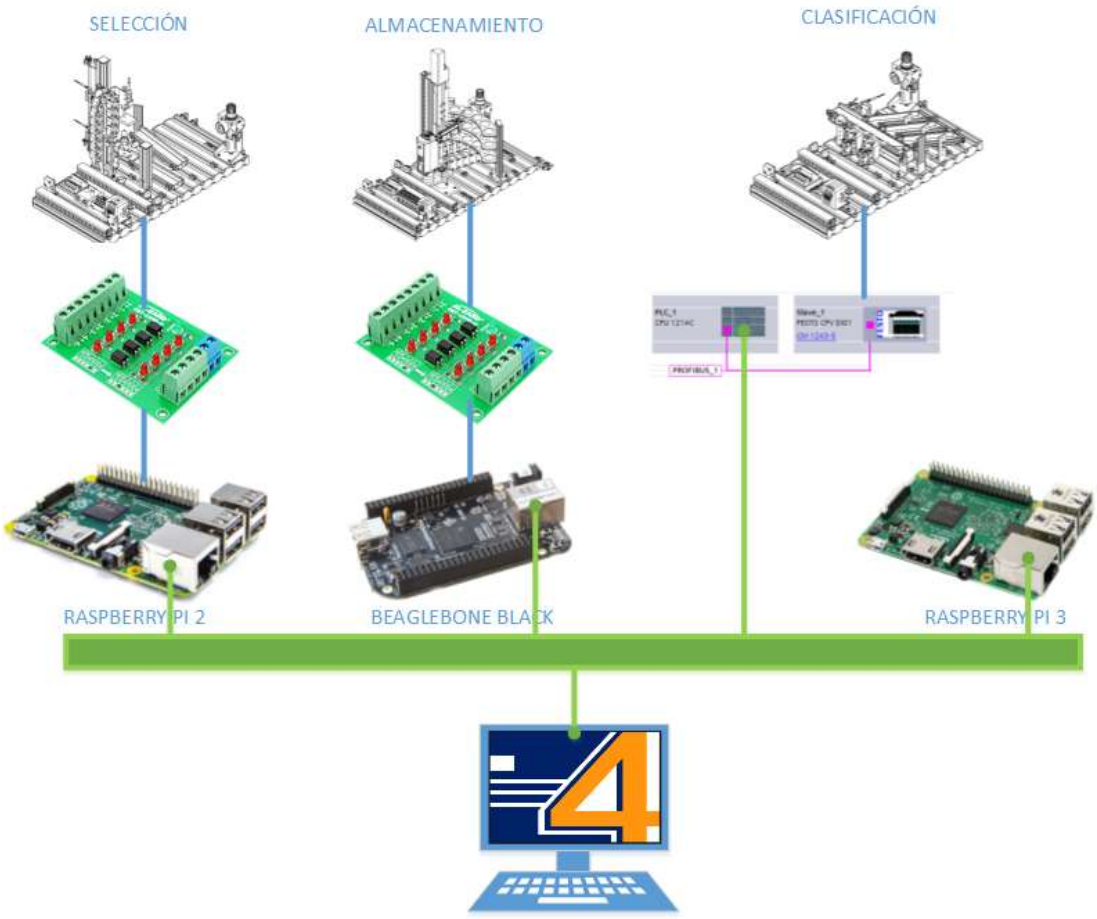



*Figura. 91 Distribución física del sistema distribuido implementado*

1. El ordenador en el cual se ejecuta 4DIAC-IDE para la modificación y descarga del sistema distribuido con sus aplicaciones.
2. La tarjeta de desarrollo Raspberry Pi 2 que se encuentra interactuando con el sistema de entradas y salidas del módulo de selección a través de una de las dos tarjetas de acople desarrolladas.
3. La tarjeta de desarrollo Beaglebone Black que se encuentra interactuando con el sistema de entradas y salidas del módulo de almacenamiento a través de una de las dos tarjetas de acople desarrolladas.

4. La tarjeta de desarrollo Raspberry Pi 3 destinada al control del módulo de clasificación.
5. Un PLC SIEMENS S7-1200 que actúa como maestro Profibus del sistema de entradas y salidas del módulo de clasificación.

La presencia del PLC dentro de la red del sistema distribuido se debe a la naturaleza de las funciones generadas en el apartado ***FBs para manipulación S7***. Dichos FBs permiten combinar las capacidades físicas de un PLC (incluyendo sus compatibilidades de comunicación) con las capacidades de procesamiento de una de las tarjetas de desarrollo, siempre y cuando exista una conexión Ethernet entre ambos dispositivos. Este tipo de configuración en la que se utiliza un PLC como una extensión de la tarjeta de control, demuestra que los FBs resultantes del desarrollo del presente proyecto de investigación más allá de brindar las características de: interoperabilidad, reconfiguración y portabilidad; permiten la reutilización de controladores de marca SIEMENS totalmente funcionales físicamente pero considerados obsoletos por sus capacidades de procesamiento, al volverlos extensiones físicas de tarjetas con sistemas operativos embebidos y mejores capacidades de procesamiento.

La configuración final del sistema distribuido desarrollado se muestra en la Figura. 92, en la cual se muestran las siguientes 4 aplicaciones disponibles para su despliegue en tres posibles dispositivos a través de un bus Ethernet:

1. Aplicación de control del módulo de selección (rosada), configurada para ser desplegada en la tarjeta Raspberry Pi 2 con dirección IP 192.168.0.15; cuya programación se presenta en la Figura. 93.
2. Aplicación de control del módulo de almacenamiento (celeste), configurada para ser desplegada en la tarjeta Beaglebone Black con dirección IP 192.168.0.10, cuya programación se presenta en la Figura. 94.
3. Aplicación de control del módulo de clasificación (verde), configurada para ser desplegada en la tarjeta Raspberry Pi 3 con dirección IP 192.168.0.25; cuya programación se presenta en la Figura. 95.

4. Aplicación de control del módulo de clasificación (café), configurada para ser desplegada en la tarjeta Beaglebone Black con dirección IP 192.168.0.10; cuya programación se presenta en la Figura. 96.

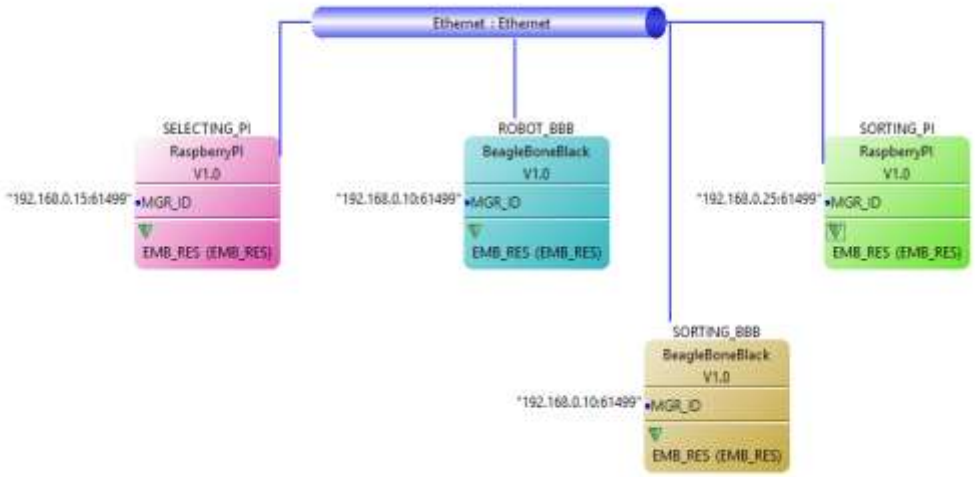



*Figura. 92 Sistema de control distribuido final desarrollado en 4DIAC-IDE, destinado al control de las estaciones de selección, almacenamiento y clasificación del sistema modular FESTO presentado en el apartado CASO DE ESTUDIO*

Al analizar a detalle la codificación de ambas aplicaciones destinadas al control del módulo de clasificación presentadas en la Figura. 95 y la Figura. 96, se puede apreciar que se trata del mismo código mapeado a diferentes dispositivos. Esto se lo realizó con la finalidad de poder comprobar las características de reconfiguración y portabilidad que brinda la norma IEC-61499 a sus sistemas.

La característica de portabilidad se comprueba: 1) al no tener que generar una codificación distinta para ambas aplicaciones de control a pesar de que sean destinadas a dos dispositivos de marcas comerciales diferentes, y 2) al obtener el mismo comportamiento por parte de ambas aplicaciones ejecutadas en sus respectivas tarjetas.

La característica de reconfiguración se comprueba: 1) al poder realizar cualquier cambio en la programación de la aplicación y poder descargarlo inmediatamente sin incidir en complicaciones de funcionamiento, y 2) al poder cambiar totalmente la aplicación a ejecutar en un controlador para que

este gobierne el funcionamiento de un módulo diferente. Esta segunda capacidad se la comprueba con la ayuda de las tarjetas Beaglebone Black y Raspberry Pi 3 que se encuentran controlando los módulos de almacenamiento y clasificación respectivamente. El procedimiento consiste en detener la ejecución de ambas aplicaciones, posteriormente con la ayuda de 4DIAC-IDE se descarga la aplicación de control del módulo de clasificación en la tarjeta Beaglebone Black y sin necesidad de realizar algún cambio físico o lógico, la tarjeta pasa a tomar el control del módulo y genera exactamente el mismo funcionamiento que con la tarjeta Raspberry Pi.

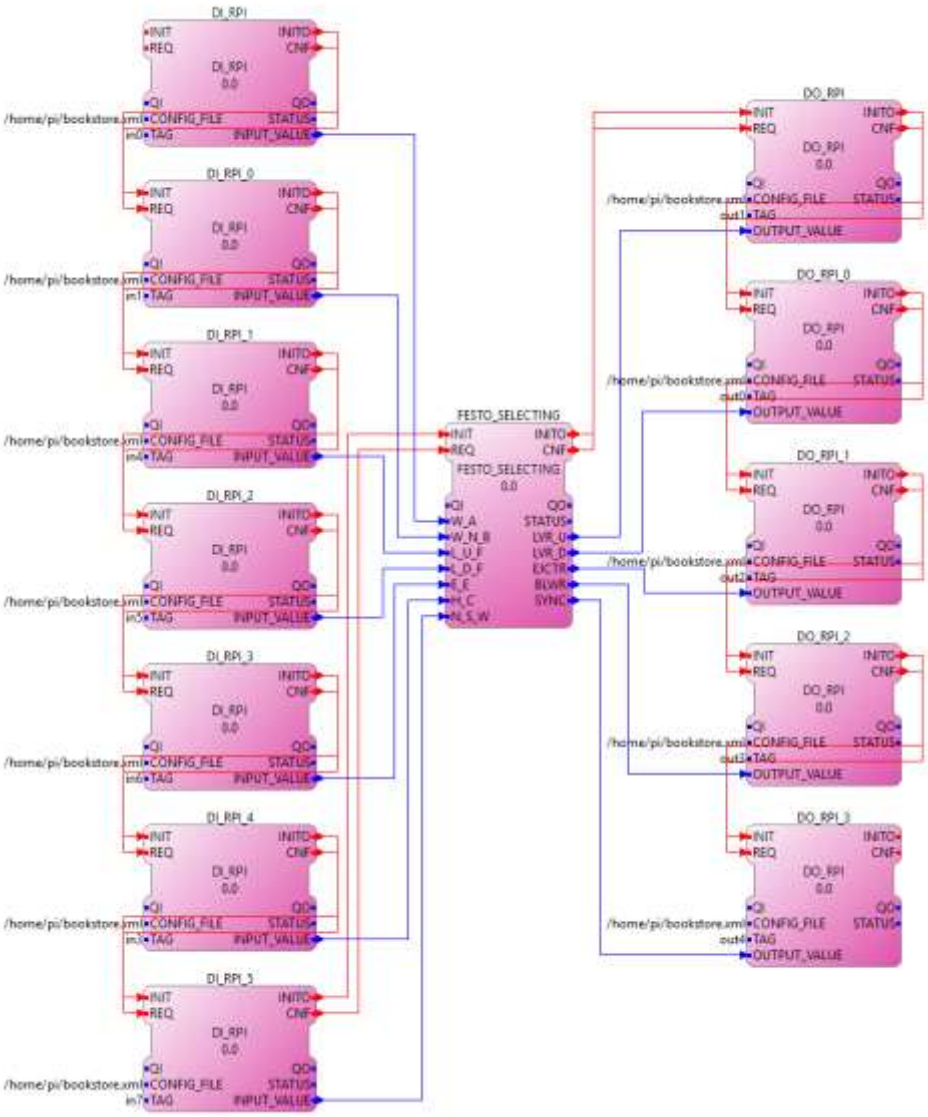



*Figura. 93 Aplicación de control de módulo de selección mapeada a tarjeta Raspberry Pi 2, generada a partir de los FBs para manejo de los GPIO de la tarjeta Raspberry y el FB de control FESTO_SELECTING*

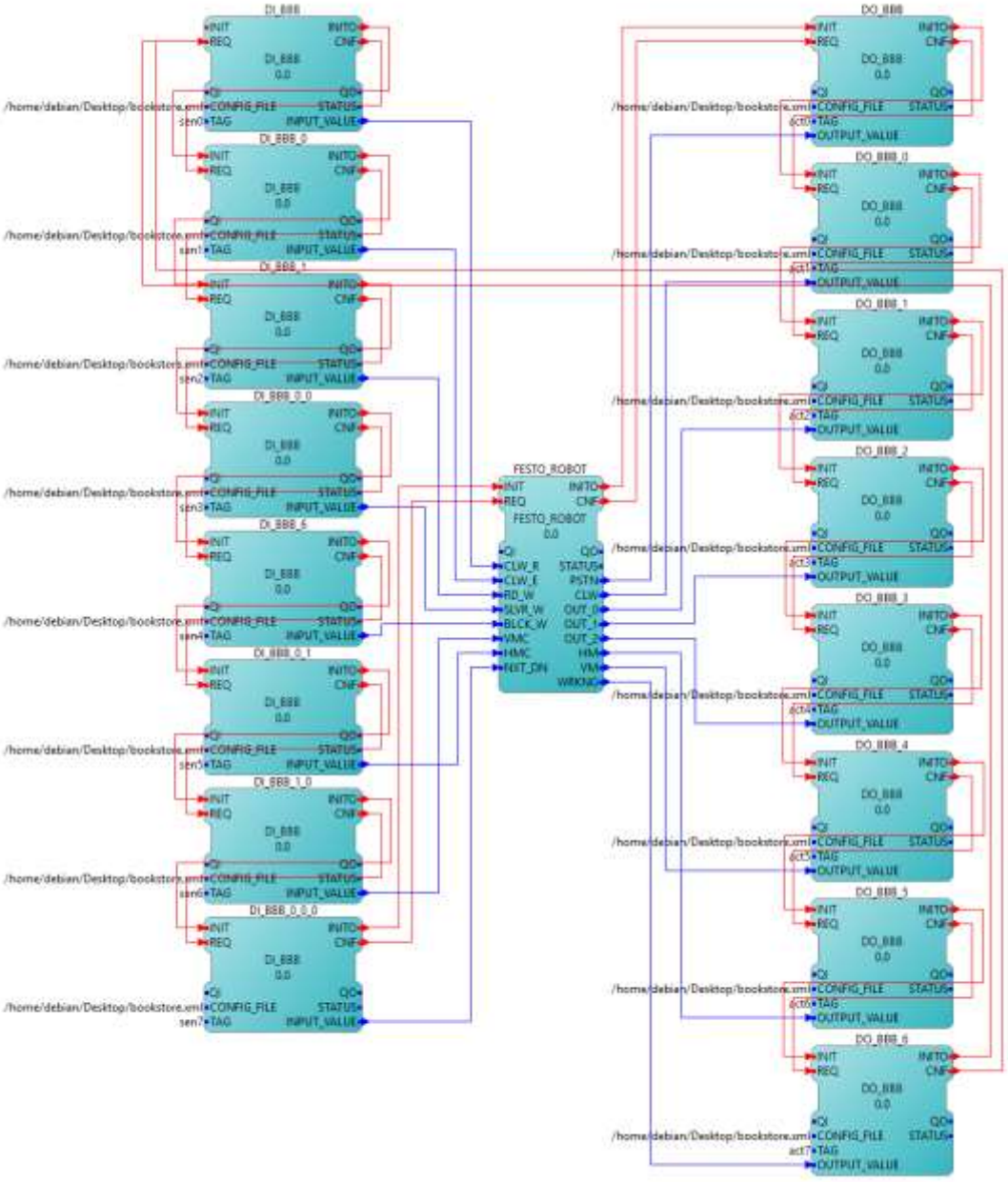



*Figura. 94 Aplicación de control de módulo de almacenamiento mapeada a tarjeta Beaglebone Black, generada a partir del uso de los FBs para el manejo del puerto GPIO de la tarjeta Beaglebone Black y el FB de control FESTO_ROBOT*

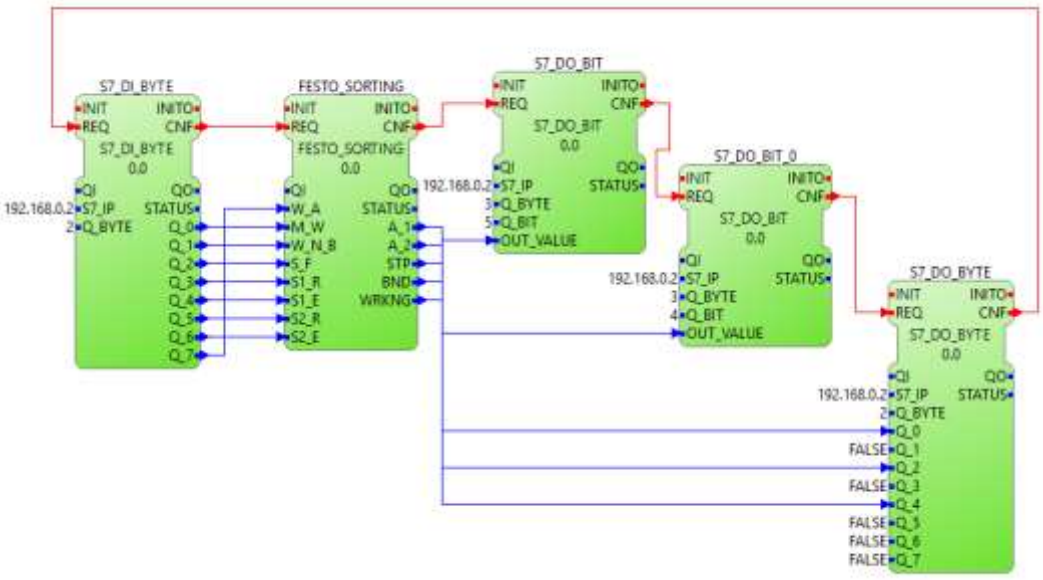



*Figura. 95 Aplicación de control de módulo de clasificación mapeada a tarjeta Raspberry Pi 3, generada a partir de los FBs para manipulación S7 y el FB de control FESTO_SORTING*

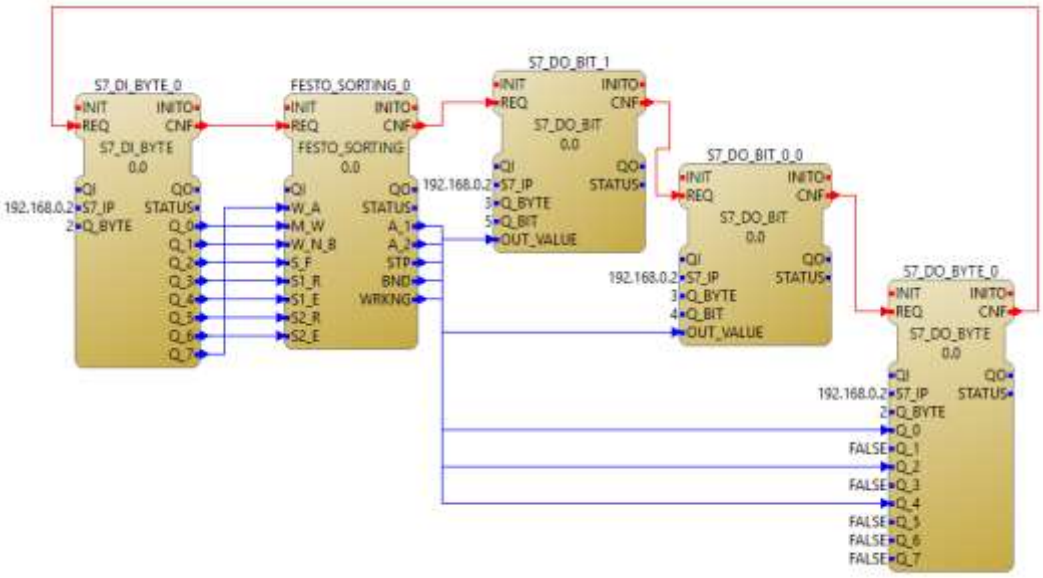



*Figura. 96 Aplicación de control de módulo de clasificación mapeada a tarjeta Beaglebone Black, generada a partir de los FBs para manipulación S7 y el FB de control FESTO_SORTING*

## Pruebas comparativas y de funcionalidad

Para poder tener un punto de contraste entre el sistema de control original desarrollado bajo IEC-61131 y el nuevo sistema de control distribuido desarrollado bajo IEC-61499, fue necesaria la ejecución de una serie de pruebas para poder obtener datos de análisis.

### *Tiempo de generación de la misma aplicación para dos dispositivos*

En la prueba a describir intervino un grupo humano comprendido por 30 estudiantes universitarios y 3 docentes con conocimientos de programación y manipulación de PLCs. Este grupo humano recibió una capacitación básica por parte de los autores del presente trabajo de investigación, sobre la programación utilizando FBs y el despliegue de las aplicaciones generadas utilizando 4DIAC-IDE.

El objetivo final de esta prueba era medir el tiempo en que cada uno de los participantes del grupo humano se demoraba en generar la misma aplicación de control para dos dispositivos tanto en IEC-61131 como en IEC-61499, la que consistía en: escribir la codificación necesaria para leer un byte de las entradas digitales y reflejar ese valor en las salidas digitales del dispositivo, posterior a lo cual se debía configurar la programación para ser descargada en dicho controlador. La primera etapa de esta prueba buscaba generar la misma aplicación de control para un PLC SIEMENS y para un PLC FESTO. Mientras que en la siguiente etapa se usaron como dispositivos las tarjetas Beaglebone Black y Raspberry Pi, dando así los resultados presentados en la Tabla 10:

*Tabla 10*

*Resultados expresados en minutos de los tiempos invertidos por los participantes en la prueba de generación de la misma aplicación para dos dispositivos*

| ***Participante*** | **Dispositivos IEC-61131** | **Dispositivos IEC-61499** |
|---|---|---|
| *Estudiante 1* | 5,05 | 2,00 |
| *Estudiante 2* | 4,53 | 1,73 |
| *Estudiante 3* | 6,28 | 2,90 |
| *Estudiante 4* | 5,62 | 2,22 |
| *Estudiante 5* | 5,34 | 2,13 |
| *Estudiante 6* | 5,86 | 1,53 |
| *Estudiante 7* | 5,74 | 1,43 |
| *Estudiante 8* | 5,32 | 1,63 |
| *Estudiante 9* | 5,17 | 2,21 |
| *Estudiante 10* | 5,78 | 2,00 |
| *Estudiante 11* | 6,34 | 2,20 |
| *Estudiante 12* | 4,60 | 2,09 |

Continúa

| | | |
|---|---|---|
| *Estudiante 13* | 4,45 | 2,03 |
| *Estudiante 14* | 4,40 | 1,58 |
| *Estudiante 15* | 5,14 | 2,20 |
| *Estudiante 16* | 4,44 | 2,50 |
| *Estudiante 17* | 4,67 | 2,14 |
| *Estudiante 18* | 5,02 | 1,90 |
| *Estudiante 19* | 5,01 | 1,82 |
| *Estudiante 20* | 5,26 | 1,40 |
| *Estudiante 21* | 5,42 | 2,45 |
| *Estudiante 22* | 5,68 | 2,04 |
| *Estudiante 23* | 5,28 | 2,22 |
| *Estudiante 24* | 5,22 | 1,53 |
| *Estudiante 25* | 4,95 | 1,45 |
| *Estudiante 26* | 4,31 | 2,08 |
| *Estudiante 27* | 4,87 | 2,11 |

Continúa

| | | |
|---|---|---|
| *Estudiante 28* | 5,61 | 1,97 |
| *Estudiante 29* | 5,36 | 1,83 |
| *Estudiante 30* | 5,09 | 2,32 |
| *Docente 1* | 4,65 | 1,21 |
| *Docente 2* | 4,78 | 2,10 |
| *Docente 3* | 5,09 | 2,04 |

Con los datos recopilados se realizaron los cálculos necesarios para obtener los datos presentados en la Tabla 11:

*Tabla 11*

*Tiempos: mínimo, máximo y promedio de los resultados presentados en la Tabla 10*

| | **Tiempo mínimo** | **Tiempo máximo** | **Tiempo Promedio** |
|---|---|---|---|
| *Programación basada en IEC-61131* | 4,31 min | 6,34 min | 5,16 min |
| *Programación basada en IEC-61499* | 1,21 min | 2,9 min | 1,96 min |

### *Tiempo de descarga de programación en dispositivos*

La finalidad de esta prueba era la de comparar los tiempos necesarios para descargar las programaciones generadas por cada uno de los participantes, los cuales se detallan en Tabla 12:

*Tabla 12*

*Tiempo de descarga de las aplicaciones desarrolladas en sus correspondientes dispositivos, expresados en minutos*

| ***Participante*** | **Dispositivos IEC-61131** | **Dispositivos IEC-61499** |
|---|---|---|
| *Estudiante 1* | 1,02 | 0,34 |
| *Estudiante 2* | 1,06 | 0,32 |
| *Estudiante 3* | 1,11 | 0,33 |
| *Estudiante 4* | 1,02 | 0,4 |
| *Estudiante 5* | 1,15 | 0,39 |
| *Estudiante 6* | 1,14 | 0,37 |
| *Estudiante 7* | 1,09 | 0,3 |
| *Estudiante 8* | 1,09 | 0,38 |
| *Estudiante 9* | 1,1 | 0,37 |
| *Estudiante 10* | 1,12 | 0,32 |

Continúa

| | | |
|---|---|---|
| *Estudiante 11* | 1,09 | 0,39 |
| *Estudiante 12* | 1,06 | 0,3 |
| *Estudiante 13* | 1,1 | 0,32 |
| *Estudiante 14* | 1,17 | 0,34 |
| *Estudiante 15* | 1,17 | 0,3 |
| *Estudiante 16* | 1,04 | 0,4 |
| *Estudiante 17* | 1,01 | 0,38 |
| *Estudiante 18* | 1,02 | 0,36 |
| *Estudiante 19* | 1,04 | 0,36 |
| *Estudiante 20* | 1,11 | 0,36 |
| *Estudiante 21* | 1,05 | 0,35 |
| *Estudiante 22* | 1,2 | 0,31 |
| *Estudiante 23* | 1,02 | 0,4 |
| *Estudiante 24* | 1,17 | 0,36 |
| *Estudiante 25* | 1,02 | 0,32 |

Continúa

| | | |
|---|---|---|
| *Estudiante 26* | 1,18 | 0,33 |
| *Estudiante 27* | 1,16 | 0,35 |
| *Estudiante 28* | 1,09 | 0,36 |
| *Estudiante 29* | 1,03 | 0,37 |
| *Estudiante 30* | 1,06 | 0,3 |
| *Docente 1* | 1,17 | 0,37 |
| *Docente 2* | 1,08 | 0,4 |
| *Docente 3* | 1,15 | 0,31 |

Con los datos recopilados se realizaron los cálculos necesarios para obtener los datos presentados en la Tabla 12:

*Tabla 13*

*Tiempos: mínimo, máximo y promedio de los resultados presentados en la Tabla 12*

| | **Tiempo mínimo** | **Tiempo máximo** | **Tiempo Promedio** |
|---|---|---|---|
| *Programación basada en IEC-61131* | 1,01 min | 1,2 min | 1,09 min |
| *Programación basada* | 0,3 min | 0,4 min | 0,35 min |

Continúa

| | | | |
|---|---|---|---|
| *en IEC-61499* | | | |

### *Tiempo de culminación de un ciclo de trabajo de los sistemas*

La finalidad de esta prueba consistía en cronometrar los tiempos de 20 ciclos de trabajo del sistema basado en IEC-61131 y del sistema basado en IEC-61499 en donde la pieza no sea filtrada en el módulo de selección; los resultados se presentan en la Tabla 14:

*Tabla 14*

*Tiempos de iteraciones medidos en los sistemas basados en IEC-61131 e IEC-61499 expresados en segundos*

| No. De Iteración | Sistema IEC-61131 | Sistema IEC-61499 |
|---|---|---|
| **1** | 41,5 | 36,5 |
| **2** | 42,2 | 38,9 |
| **3** | 41,7 | 38,9 |
| **4** | 40,1 | 38 |
| **5** | 41,9 | 36,5 |
| **6** | 42,1 | 37,3 |
| **7** | 41,3 | 38,8 |
| **8** | 40,1 | 38,8 |

| | | |
|---|---|---|
| **9** | 41,4 | 37,7 |
| **10** | 40,4 | 37,9 |
| **11** | 42,3 | 38,3 |
| **12** | 42,8 | 36,8 |
| **13** | 40,9 | 36,3 |
| **14** | 42,1 | 37,5 |
| **15** | 40,7 | 38 |
| **16** | 41,1 | 37,6 |
| **17** | 41,9 | 37,5 |
| **18** | 40,5 | 37,4 |
| **19** | 41,7 | 38,1 |
| **20** | 40,7 | 37,8 |

Con los datos recopilados se realizaron los cálculos necesarios para obtener los datos presentados en la Tabla 15:

*Tabla 15*

*Tiempos: mínimo, máximo y promedio de los resultados presentados en la Tabla 14*

| | **Tiempo mínimo** | **Tiempo máximo** | **Tiempo Promedio** |
|---|---|---|---|
| *Iteraciones de sistema IEC-61131* | 40,1 seg | 42,8 seg | 41,37 seg |
| *Iteraciones de sistema IEC-61499* | 36,3 seg | 38,9 seg | 37,73 seg |

## Análisis de resultados

### *Verificación de características distintivas de la norma IEC-61499*

Como se indicó en el apartado ***Estándar IEC-61499,*** la norma IEC-61499 se generó con la finalidad de solventar las limitaciones en sistemas de control creados en base a la norma IEC-61131 mediante: 1) el remplazo de paradigmas centralizados y descentralizados por arquitecturas distribuidas, y 2) al buscar que sus sistemas de control posean las características de: interoperabilidad, portabilidad y reconfiguración.

A lo largo del desarrollo del sistema de control planteado como meta final del presente trabajo de investigación, se pudieron apreciar las ventajas de un sistema con arquitectura distribuida, teniendo así:

- ***Una mayor facilidad al momento de analizar el proceso y generar las aplicaciones de control.*** En sistemas de control basados en arquitecturas centralizadas, mientras mayor complejidad tenga el proceso mayor dificultad presenta su análisis, dando como resultado

codificaciones muy extensas. Por el contrario, cuando se maneja una arquitectura distribuida, el proceso se puede dividir en partes, colocando un controlador con codificaciones reducidas en cada una de ellas que por su costo y capacidad de procesamiento resulta muy conveniente además de facilitar el análisis del mismo.

- ***La eliminación del riesgo de paro total de la planta de producción ante un problema del controlador.*** El componente con mayor importancia de los sistemas de control desarrollados en base a la norma IEC-61131 es el controlador, debido a que si este sufre algún daño considerable se pierde totalmente el control de la planta. El sistema implementado demuestra que al tener más de un controlador disponible, es posible la reasignación de control en caso se falla, como se lo hizo al momento de cambiar el gobierno del módulo de clasificación de la tarjeta Raspberry a la Beaglebone.

Todas las casas comerciales de soluciones de control compatibles con IEC-61131 buscan obtener el mayor rédito posible de sus ventas, para lo cual cerraron la manera en la que sus dispositivos de control se comunican y programan; desencadenando en la imposibilidad de: 1) configurar los autómatas de control con software ajeno a la casa comercial, 2) comunicar directamente dispositivos de control multimarca y por ende 3) poder integrar controladores de distintas marcas dentro de un mismo sistema de control.

Con el desarrollo de funciones genéricas destinadas a ser implementadas en dispositivos de diferentes casas comerciales utilizando la misma herramienta de software, se puede observar de primera mano la característica de interoperabilidad; la misma que se ve reflejada de nuevo al momento de descargar la configuración del sistema en los dispositivos de control, en donde no fue necesario el uso de una herramienta de despliegue por cada marca de dispositivo de control utilizado. Con esto se obtiene una solución a la problemática de integración característica de los sistemas de control basados en IEC-61131.

Finalmente, uno de los mayores problemas que se tiene con los sistemas de control tradicionales, se presenta al momento de querer cambiar el dispositivo de control por uno de una marca distinta. En estos casos, para poder implementar la misma codificación de un controlador a otro es necesario repetir las mismas líneas de código en el software de configuración del nuevo dispositivo; es decir, se debe realizar una completa traducción de software. Con la ayuda de las dos aplicaciones encargadas del control del módulo de clasificación que se generaron, se observó y comprobó la característica de portabilidad; estas aplicaciones nos demuestran que, al trabajar con funciones genéricas se puede utilizar exactamente la misma programación para configurar dos dispositivos no relacionados comercialmente, sin necesidad de una gran inversión de tiempo en su programación y configuración.

### *Comparación Sistema de Control IEC-61131 Vs Sistema de Control IEC-61499*

El verdadero trasfondo de la primera prueba en la que se buscaba medir los tiempos invertidos en la generación de la misma aplicación para dos dispositivos diferentes, era el de corroborar con datos numéricos las ventajas que se obtienen al eliminar la necesidad de traducir de un software de programación a otro las codificaciones desarrolladas.

Respecto a los tiempos relacionados a la programación en software compatible con IEC-61131 se puede destacar 4 aspectos importantes: (1) el menor tiempo invertido fue de 4,31 minutos, (2) el mayor tiempo utilizado fue de 6,34 minutos, (3) la diferencia entre los tiempos máximo y mínimo es de casi 2 minutos y (4) la media de tiempo invertido es de 5,16 minutos. Basándose en conversaciones posteriores a las pruebas con los participantes, se puede atribuir la mayor o menor demora durante la prueba al nivel de interacción que tenían con las herramientas de software utilizadas, teniendo personas que manejaban fluidamente ambos programas, así como los que solo tenían conocimiento del manejo de uno de ellos. De igual manera, de las conversaciones mantenidas con los participantes después de las pruebas se pudo determinar que cada uno de los

participantes atribuían su tiempo obtenido a la necesidad de utilizar dos programas para desarrollar la misma codificación.

En contraste, de los tiempos relacionados a la programación en software compatible con IEC-61499 se puede destacar: (1) el menor tiempo invertido fue de 1,21 minutos, (2) el mayor tiempo empleado fue de 2,9 minutos, (3) la diferencia entre los tiempos máximo y mínimo de los resultados es de 1,69 minutos y (4) la media de tiempo invertido de los participantes es de 1,96 minutos. Tras las conversaciones post-prueba, se pudo apreciar la conformidad de los participantes al ya no tener que utilizar varias herramientas de software, y aprovechar de mejor manera el tiempo de programación al solo tener que generar una sola programación para posteriormente mapearla a los dispositivos diferentes; demostrando no solo a través de conversaciones, sino también a través de los 3,2 minutos de diferencia entre las medias de tiempos usados, que la característica de portabilidad verdaderamente otorga mejoras durante las configuraciones de procesos al ahorrar el tiempo desperdiciado en una traducción de software de un código ya generado.

Antes de empezar con el análisis de la segunda prueba en donde se realizaron las descargas de las codificaciones generadas por los participantes, se debe mencionar que la finalidad de dicha prueba era la de comprobar uno de los beneficios de las arquitecturas distribuidas, que se obtiene al integrar dispositivos multimarca dentro de un mismo sistema. A diferencia de los resultados obtenidos de la primera prueba, las medias obtenidas de tiempo en el sistema IEC-61131 y en el sistema IEC-61499 casi no tenían mucha diferencia con sus correspondientes máximos y mínimos. En donde sí se puede apreciar una diferencia considerable es al momento de contrastar los valores de ambos promedios obtenidos. El promedio de tiempo del proceso de descarga en el sistema IEC-61131 es de 1,09 minutos, mientras que el del sistema IEC-61499 casi llegó a ser su tercera parte con un valor de 0,35 minutos; esta diferencia se la atribuye a la capacidad del sistema IEC-61499 de integrar dispositivos multimarca dentro de un mismo sistema y poder utilizar la misma herramienta de software para

descargar sus respectivas programaciones, evitando tiempos de espera provocados al dividir los recursos computacionales del ordenador para que dos programas distintos realicen la misma operación de descarga simultáneamente.

Inicialmente, una de las características deseadas en el sistema distribuido resultante, era que su funcionamiento se asemeje en mayoría de posibilidad al del sistema de control que utilizaba en un inicio el sistema FESTO. Al comparar la media de ciclo del sistema IEC-61131 de 41.37 segundos con la media de ciclo del sistema IEC-61499 de 37,73, se puede observar que más allá de obtener una réplica de funcionamiento por parte del sistema distribuido, se obtiene adicionalmente una pequeña mejora al completar un ciclo aproximadamente 3,64 segundos más pronto que el sistema de control original. Cabe destacar que esta diferencia podía tener un mayor valor, debido a que, por las capacidades de procesamiento de las tarjetas de desarrollo fue necesaria la introducción de retardos en la codificación de los FBs de control para que su operación no sea demasiado acelerada y se generen errores de lectura o escritura; con lo que se muestra que se puede inclusive mejorar los tiempos de ejecución al llevar los sistemas de control tradicional a un paradigma distribuido bajo IEC-61499.

### *Comprobación de hipótesis*

A lo largo del desarrollo del sistema de control distribuido en dispositivos de bajo costo para procesos discretos de la maqueta FESTO FMS-200, se pudo comprender de mejor manera que al combinar el uso de una arquitectura distribuida con la generación de funciones genéricas para la programación de dispositivos se brindan las siguientes características a un sistema de control:

- Mayor facilidad de análisis del proceso al dividirlo en secciones funcionales.
- Descentralización del control y eliminación del riesgo latente de paro total del proceso si el controlador sufre un desperfecto, al involucrar más de un dispositivo para el gobierno de procesos.

- Posibilidad de integrar dispositivos multimarca dentro de un mismo sistema.
- Poder utilizar el mismo código de programación en dos dispositivos de casas comerciales no relacionadas y poder programarlos con la intervención de una misma herramienta de software.
- Reconfiguraciones de dispositivos más rápidas reduciendo así tiempos de paro ante desperfectos.
- Mejoras en tiempos de ejecución de sistemas de control al utilizar dispositivos con mejores capacidades lógicas.

Con dichas características, no solo se puede decir que la implementación de sistemas basados en IEC-61499 puede ser una alternativa para generar sistemas de control similares a IEC-61131, sino que se puede comprobar que la nueva norma es la respuesta para poder generar sistemas de control hechos a la medida de los procesos, con los mejores dispositivos provenientes de varias casas comerciales y con capacidades de fácil adaptación a medida que la planta vaya creciendo; características que con la actual norma estandarizada a nivel global para la automatización IEC-61131 son imposibles de obtener.

De esta manera se valida que a través del desarrollo del sistema de control distribuido en dispositivos de bajo costo para procesos discretos de la maqueta FESTO FMS-200 propuesto en el presente trabajo de investigación, se comprobó una muy recomendada aplicabilidad de la norma IEC-61499 en sistemas industriales.

## Difusión del conocimiento generado.

Los resultados obtenidos en el presente proyecto de investigación fueron recopilados en cuatro papers científicos, los cuales fueron analizados por investigadores a nivel mundial y tuvieron el prestigio de ser aceptados en cuatro diferentes congresos. Los papers publicados se enlistan a continuación:

1. *CPPS on Low Cost Devices For Batch Process under IEC-61499 and ISA-88* publicado en el congreso alemán **2017 IEEE 15th International Conference on Industrial Informatics (INDIN),** llevado a cabo del 24 al 26 de Julio de 2017.
2. *Enabling an Automation Architecture of CPPs based on UML combined with IEC-61499* publicado en el congreso koreano **2017 17th International Conference on Control, Automation and Systems (ICCAS),** llevado a cabo del 18 al 21 de Octubre de 2017**.**
3. *Designing Automation Distributed Systems based on IEC-61499 and UML* en el congreso mexicano **2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT 2017)**, llevado a cabo del 25 al 27 de Octubre de 2017.
4. *Fuzzy Control Implementation in Low Cost CPPS Devices* publicado en el congreso koreano **2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017),** llevado a cabo del 16 al 18 de Noviembre de 2017**.**

De igual manera, con los resultados obtenidos durante el desarrollo del presente trabajo de investigación se obtuvieron reconocimientos a nivel nacional, teniendo así:

- 1er lugar en el “Evento Académico Zonal” del concurso “Galardones Nacionales 2017” en el área de “Ingeniería e Industria”, llevado a cabo del 17 al 21 de Julio de 2017.
- 3er lugar en el concurso “Galardones Nacionales 2017” en el área de “Ingeniería e Industria”, llevado a cabo el 14 de Septiembre de 2017.

# CAPÍTULO VI

# 6. Conclusiones & Recomendaciones

## 6.1. Conclusiones

- La norma IEC-61499 permite generar sistemas de control hechos a la medida de los procesos de las industrias, incluyendo en ellos la capacidad de ser modificados fácilmente ante cualquier nueva necesidad de la planta de producción, sin incidir en grandes inversiones económicas, temporales o de mano de obra.
- La norma IEC-61499 es aplicable en dispositivos que operen bajo un software embebido teniendo excelentes resultados de integración en distribuciones de Linux, debido a que, al ser software libre permite a los desarrolladores una manipulación más profunda de sus componentes.
- Se seleccionaron las tarjetas de desarrollo Raspberry Pi y Beaglebone Black para el presente trabajo de investigación, debido a que el resto de tarjetas de bajo costo del mercado que funcionan con un software embebido, basan sus diseños físicos y lógicos en estas dos tarjetas.
- Al generar la codificación de FBs bajo lenguajes de programación de alto nivel como C++, se puede incluir una amplia gama de funciones y librerías que permiten el desarrollo de cualquier aplicación deseada, permitiendo generar desde FBs encargados de operaciones aritméticas hasta FBs que encapsulen algoritmos de control avanzado.
- El modelo jerárquico presentado en la norma ISA-88 permite desarrollar en forma más sencilla las aplicaciones de control basadas en lenguajes de programación de alto nivel, esto se debe a que sus resultados radican en un grupo de recetas que especifican tanto las acciones requeridas como su correcto orden de ejecución.

- Una arquitectura distribuida permite dividir el proceso global en pequeños módulos que permiten cambiar el paradigma de la utilización de un solo dispositivo para el control global, por el uso de múltiples dispositivos. Esto conlleva a la generación de programaciones más simples y a un análisis de procesos menos complicado.
- Usando 4DIAC-IDE como única herramienta de software para configurar todos los dispositivos involucrados en el sistema y generar las aplicaciones a ser descargadas en cada uno de ellos, se observa claramente la característica de interoperabilidad. De igual manera, 4DIAC-IDE da las facilidades al desarrollador para simular individualmente la operación de cada uno de los FBs de sus librerías, facilitando la depuración de errores que puedan ser acarreados a aplicaciones y posteriormente al sistema final.
- Al tener que cambiar únicamente el mapeo de una aplicación para descargarla en cualquier cantidad de controladores, se hizo presente la característica de portabilidad, que implica utilizar la misma codificación en dispositivos de casas comerciales diferentes sin caer en la obligación de hacer una traducción de software; mientras que al cambiar con un tiempo sumamente reducido el dispositivo que controla un módulo se evidencia la característica de reconfiguración.
- Las funciones generadas que encapsulan el protocolo de comunicación S7, presentan una alternativa que ayuda a la reutilización del hardware de las industrias, al combinar la robustez física para ámbitos industriales de controladores considerados obsoletos por sus capacidades lógicas, con la potencia computacional de tarjetas de desarrollo sin capacidades industriales.
- Con la implementación de un sistema de control distribuido y la generación de nuevas soluciones para reutilización de hardware industrial, no solo se corroboran las características de la norma IEC-61499 sino que también se aporta al escalamiento de la misma con la generación de nuevos estudios que demuestran todos sus beneficios.

## 6.2. Recomendaciones

- Tras la implementación del protocolo de comunicación S7 de SIEMENS, se recomienda generar investigaciones sobre otros protocolos nativos de casas comerciales y de esta manera, poder generar nuevas funciones compatibles con IEC-61499 que permitan la reutilización y repotenciación de una mayor variedad de controladores industriales con excelentes capacidades físicas, pero deficientes capacidades de procesamiento.
- Este proyecto de investigación consiguió la implementación de un sistema distribuido bajo estrategias de control básicas; sin embargo, se recomienda realizar los estudios necesarios para poder integrar estrategias de control avanzadas dentro de la codificación de los FBs.
- Se recomienda realizar el desarrollo de funciones compatibles con IEC-61499 que permitan el uso de PLCs con software embebido en sistemas de control distribuido, para colaborar con la expansión de la norma IEC-61499 y aportar en forma significativa a la evolución del control a nivel industrial.
- De igual manera, se recomienda trabajar a futuro en el desarrollo de sistemas distribuidos en cuyas programaciones se encapsulen los algoritmos necesarios para generar reconfiguraciones dinámicas ante fallos de los controladores; lo que permitiría disminuir los tiempos de paro de producción (ocasionados por desperfectos en los controladores) hasta casi volverlos inexistentes.

# Referencias Bibliográficas

Barret, S. F., & Kridner, J. (2016). *Bad to the Bone: Crafting Electronic Systems with BeagleBone Black* (2da Edició). Morgan & Claypool. https://doi.org/10.2200/S00675ED1V01Y201509DCS046

Denis, A. K. (2016). *Raspberry Pi Computer Architecture Essentials* (First Edit). United Kingdom: Packt Publishing.

Dubinin, V., & Vyatkin, V. (2006). Formal Semantic Model of IEC 61499 Function Blocks. *Interface*, 6–11.

ElectroniLab. (2016). BeagleBone Black ARM Cortex-A8 - 1GHz. Retrieved May 1, 2017, from https://electronilab.co/tienda/beaglebone-black-arm-cortex-a8-1ghz/

François, M. (2016). *Raspberry Pi 2: Utilice todo el potencial de su nano-ordenador* (1era Edici). Barcelona.

Kaghazchi, H., Joyce, R., & Heffernan, D. (2007). A function block diagnostic framework for a multi- vendor PROFIBUS environment. *Assembly Automation*, *27*(3), 240–246. https://doi.org/10.1108/01445150710763268

Lewis, R. (2008). *Modelling Control Systems Using IEC 61499* (First). London: The Institution of Engineering and Technology.

Lewis, R. W. (2001). Modelling Control Systems Using IEC 61499. *IEEE Publishing*.

Otto, A., & Hellmann, K. (2009). IEC 61131: A general overview and emerging trends. *IEEE Industrial Electronics Magazine*, *3*(4), 27–31. https://doi.org/10.1109/MIE.2009.934793

Ramanathan, R. (2014). The IEC 61131-3 programming languages features for industrial control systems. *World Automation Congress Proceedings*, 598–603. https://doi.org/10.1109/WAC.2014.6936062

Raspberry, O. (2015). Jessie is here. Retrieved May 1, 2017, from https://www.raspberrypi.org/blog/raspbian-jessie-is-here/

SENPLADES. (2013). Plan Nacional Buen Vivir.pdf. Retrieved May 1, 2017, from www.planificacion.gob.ec%5Cnsemplades@semplades.gob.ec%5Cnwww.buenvivir.gob.ec%5Cnwww.buenvivir.gob.ec

Sunder, C., Zoitl, A., Christensen, J. H., Steininger, H., & Fritsche, J. (2008). Considering IEC 61131-3 and IEC 61499 in the context of component frameworks. *IEEE International Conference on Industrial Informatics (INDIN)*, (i), 277–282. https://doi.org/10.1109/INDIN.2008.4618109

Vyatkin, V., & Hanisch, H.-M. (2003). Veri ® cation of distributed control systems in intelligent manufacturing. *Structure*, *61499*, 123–136.

# ANEXOS

## DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

### CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

# CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por los señores:

**ESTEBAN XAVIER CASTELLANOS NARVÁEZ**

**CARLOS ANDRÉS GARCÍA SÁNCHEZ**

En la ciudad de Latacunga, a los 11 días del mes de diciembre del 2017

**Aprobado por:**

Ing. Eddie Galarza
**DIRECTOR DEL PROYECTO**

Ing. Franklin Silva
**DIRECTOR DE CARRERA**

Dr. Rodrigo Vaca
**SECRETARIO ACADÉMICO**