ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

"DESARROLLO DE UN SISTEMA DISTRIBUIDO BAJO LA NORMA IEC-61499 PARA CONTROL DE ROBOT KUKA MODELO YOUBOT"

GABRIELA MERCEDES MAFLA MEDINA

Trabajo de Titulación modalidad: Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACIÓN INDUSTRIAL

Riobamba - Ecuador

Marzo 2019

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, denominado: "DESARROLLO DE UN SISTEMA DISTRIBUIDO BAJO LA NORMA IEC-61499 PARA CONTROL DE ROBOT KUKA MODELO YOUBOT", de responsabilidad de la señorita Gabriela Mercedes Mafla Medina, ha sido minuciosamente revisado y se autoriza su presentación.

Ing. Oswaldo Martínez Guashima, M.Sc.

PRESIDENTE _____

Ing. Marcelo García Sánchez, Ph.D.

DIRECTOR _____

Ing. Pablo Lozada Yánez, M.Sc.

MIEMBRO _____

Ing. Franklin Salazar Logroño, M.Sc.

MIEMBRO _____

Riobamba, marzo 2019

DERECHOS INTELECTUALES

Yo, Gabriela Mercedes Mafla Medina, soy responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica de Chimborazo.

_____

GABRIELA MERCEDES MAFLA MEDINA

No. Cédula: 060394242-6

DECLARACIÓN DE AUTENTICIDAD

Yo, Gabriela Mercedes Mafla Medina, declaro que el presente proyecto de investigación, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación de Maestría.

_____

GABRIELA MERCEDES MAFLA MEDINA

No. Cédula: 060394242-6

DEDICATORIA

A todas aquellas personas especiales que han sabido demostrar cariño, entrega y confianza en mí, en las diferentes etapas de mi vida. Algunas están aquí conmigo, otras en mis recuerdos y en el corazón. Sin importar en donde estén… ¡esto es para ustedes!

Gaby

AGRADECIMIENTO

Gracias a todas aquellas personas que con su amistad, apoyo, ánimo y compañía han colaborado en la realización del presente trabajo. Gracias por formar parte de mí vida, por todo lo que me han brindado y por sus buenos deseos.

Gaby

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo investigativo tuvo como finalidad el empleo del estándar IEC-61499 para el desarrollo de un sistema de control distribuido para el control de un robot Kuka modelo YouBot. Este estándar consiste en una arquitectura de referencia para el desarrollo de aplicaciones de control con lógica descentralizada. Está basada en bloques funcionales (FBs) que se definen como la unidad estructural básica de los modelos, tiene un alto nivel de versatilidad para el diseño de sistemas, ya que permite combinar software independiente del hardware utilizado. Para modelar el sistema se utilizó el software 4DIAC-IDE y se crearon bloques funcionales (FBs) para el control de la plataforma omnidireccional, el brazo robótico y el gripper de agarre que en conjunto conforman el robot; posteriormente con la ayuda del programa PuTTy se implementó de manera remota el runtime FORTE el cual fue ejecutado en una PC que se encuentra a bordo del robot Kuka YouBot. Se evidenció en los resultados que la programación mediante FBs ha hecho que estos sean vistos como una herramienta efectiva para la automatización de sistemas flexibles y reconfigurables, obteniendo una efectividad del 98,89% de aciertos en las pruebas estipuladas. Se concluye que el estándar permite controlar el algoritmo mediante eventos, ya que tiene una conexión entre la ejecución del flujo de eventos y flujo de datos, de esta manera se puede establecer prioridades en el orden de ejecución de los bloques funcionales, con lo cual se probó el funcionamiento y las prestaciones que ofrecen este nuevo método de programación y las últimas versiones de software basadas en él disponibles actualmente.

PALABRAS CLAVES: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <ROBÓTICA>, <SISTEMA DISTRIBUIDO>, <NORMA IEC-61499>, <ROBOT KUKA YOUBOT>, <BLOQUES FUNCIONALES>, <4DIAC-IDE (SOFTWARE) >, <ENTORNO DE EJECUCIÓN (FORTE)>

SUMMARY

The present research work has as a purpose the use of the IEC-61499 standard for the development of a distributed control system for the control of a Kuka model Youbot robot. This standard consists of a reference architecture for the development of control applications with decentralized logic. It is based on functional blocks (FBs) that are defined as the basic structural unit of these models, they have a high level of versatility for the design of systems, since it allows to combine software independent of the hardware used. To model the system, the software was used 4DIAC-IDE and functional blocks (FBs) were created for the control of the omnidirectional platform, the robotic arm and the gripping gripper that together make up the robot, later with the help of the PuttY program the FORTE runtime was remotely implemented. Which was executed on a PC that is on board the robot Kuka YouBot. It was evidenced in the results that the programming by means of FBs has made that these are seen as an effective tool for the automation of flexible and reconfigurable systems, obtaining an effectiveness of 98.89% of correctness in the stipulated tests. It is concluded that the standard allows to control the algorithm through events, since it has a connection between the execution of the flow of events and data flow, in this way you can establish priorities in the order of execution of the functional blocks, which proved the operation and the features that offer This new programming method and the latest software versions based on the currently available.

KEYWORDS: <TECHNOLOGY AND SCIENCE OF ENGINEERING>, <ROBOTICS>, <DISTRIBUTED SYSTEM>, <NORM IEC-61499>, <ROBOT KUKA YOUBOT>, <FUNCTIONAL BLOCKS>, <4DIAC-IDE (SOFTWARE)>, <ENVIRONMENT EXECUTION (FORTE)>

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Planteamiento del problema

1.1.1 Situación problemática

En los últimos años, debido a los rápidos cambios en el uso de tecnología a nivel industrial, los manipuladores robóticos tuvieron que ir mejorando y adaptando su nivel de técnicas de control, dado que la manufactura automatizada de la industria y el comercio ha ido escalando a nivel de complejidad y robustez (Karl, Gonzales, Ing, & Carrera, 2015).

Los sistemas distribuidos han sido introducidos en el área de manufactura debido a su flexibilidad, confiabilidad y su mejor proporción en precio/desempeño. Sin embargo, un problema inherente es la comunicación entre elementos heterogéneos que los componen. Esto debido a la diversidad de proveedores y soluciones características tales como robots con arquitecturas de control propietarios. Representa un reto integrar este tipo de sistemas con otros componentes, sistemas de visión o la integración heterogénea entre brazos manipuladores robóticos (Stojmenovic, 2011).

El implementar sistemas distribuidos en el campo de la robótica no es una tarea fácil, debido a la diversidad de áreas de especialización interrelacionadas. Esto incrementa el grado de complejidad, conocimiento requerido y los tiempos de desarrollo e implementación de este tipo de sistemas.

La norma IEC-61499 fue desarrollada para diseñar Sistemas de Control y Medida de Procesos Industriales (IPMCS) distribuidos y abiertos. La finalidad es gestionar la creciente complejidad de los sistemas de automatización de última generación, a través de la creación de una aplicación y configuración de hardware independiente del proveedor (Steinegger, Plaschka, Melik-merkumians, & Schitter, n.d.).

En realidad aún se requiere mucho esfuerzo en la investigación y desarrollo de estos sistemas. Por lo que este trabajo se concentra en proponer una solución particular en el área de sistemas

robóticos distribuidos basado en la norma IEC-61499, partiendo de una estructura para el control de un robot Kuka youBot. La finalidad es crear bloques de funciones (FBs) o módulos de software, que permitan la conectividad y comunicación de los componentes de una manera más fácil.

1.2 Formulación del problema

¿La creación de bloques de funciones (FBs) o módulos de software bajo el estándar IEC-61499 en sistemas distribuidos, permitirá el control de un robot Kuka youBot?

1.3 Preguntas directrices o específicas de la investigación

¿Existen actualmente sistemas distribuidos que permitan el control de robots manipuladores móviles bajo la norma IEC-61499?

¿Qué tipo de arquitectura es la más óptima para el control de robots manipuladores móviles?

¿La programación de FBs permitirá la conectividad y comunicación de los componentes de una manera más fácil?

1.4 Justificación de la investigación

La falta de estándares y herramientas de desarrollo para ambientes distribuidos pueden crear graves problemas de compatibilidad, portabilidad e interconectividad en los sistemas distribuidos.

El estándar IEC-61499 busca conseguir un alto grado de control distribuido siendo este su principal objetivo. A diferencia de otros estándares de control, más antiguos y no adecuados a los nuevos componentes (sistemas distribuidos, PC industriales y redes más avanzadas), este puede ser utilizado para descentralizar la lógica de control y poder implantar controles multiagente, que hagan los sistemas más flexibles.

La investigación se centra en modelar y desarrollar componentes software para sistemas de control distribuidos usando la norma de automatización IEC-61499. Para desarrollar los componentes software distribuidos el estándar provee el modelo de Bloque de Función de Interfaz de Servicio (SIFB), que permite encapsular y abstraer al usuario del acceso al hardware, a las comunicaciones o a los recursos de las Interfaces de Programación de Aplicaciones (API),

facilitando la recopilación y el procesamiento de datos a nivel de sensores y actuadores (Doctoral, 2018), empleando los conceptos mencionados, se pretende entregar una aplicación basada en modelos innovadores de aprendizaje automatizado para el control de robots manipuladores móviles.

Los beneficiarios serán los investigadores, programadores y profesionales que se dediquen a la búsqueda de nuevos mecanismos y metodologías para el control de robots manipuladores móviles.

Adicionalmente, se puede indicar que se han realizado estudios relacionados a sistemas robóticos distribuidos en países como Estados Unidos, México, España, entre otros. En Ecuador se han registrado pocas publicaciones relacionadas a sistemas distribuidos bajo la norma IEC-61499 para control de robots manipuladores móviles específicamente, de tal manera que aún queda un largo camino por recorrer para comprobar los beneficios y prestaciones que ofrece este estándar para el desarrollo de aplicaciones de control distribuido.

1.5 Objetivos de la investigación

1.5.1 Objetivo General

Implementar un sistema de control distribuido para la manipulación de una plataforma robótica móvil bajo la norma IEC-61499.

1.5.2 Objetivos Específicos

- Analizar la composición en hardware y el desplazamiento del robot Kuka youBot para su completo control bajo la norma IEC-61499.

- Esquematizar el funcionamiento del sistema de control distribuido en bloques funcionales bajo la norma IEC-61499, para la manipulación del robot Kuka youBot mediante procesos parciales y complejos.

- Integrar el sistema lógico y físico, para evaluar el fiel cumplimiento de la norma IEC-61499 en el robot Kuka youBot.

1.6 Hipótesis

El desarrollo de un sistema distribuido bajo la norma IEC-61499, permite manipular el robot Kuka youBot cumpliendo con el concepto de control distribuido.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Antecedentes del problema

Para el desarrollo del proyecto, como referencia se recurre a los estudios anteriormente realizados sobre temas afines a la propuesta que respaldan la investigación y establecen métodos de análisis. Entre estos trabajos se tienen los siguientes:

En la tesis de investigación realizada por (García, 2013) con el tema "Implementación de Sistemas Empotrados de Control Distribuidos bajo el Estándar IEC-61499" de la Universidad del País Vasco propone: "la integración de una plataforma de control distribuido bajo el estándar IEC-61499 utilizando el runtime FORTE en dispositivos embebidos de bajo costo". El objetivo de 4DIAC es la obtención de un entorno abierto de automatización y control basado en el estándar IEC-61499.

Este trabajo aporta con las siguientes conclusiones más relevantes:

- El presente proyecto se ha centrado en el estándar IEC-61499, cuyo objetivo principal es ofrecer un entorno adecuado para el desarrollo de aplicaciones distribuidas. Para lo cual, se ha conseguido la integración de una plataforma IEC-61499 FORTE en un sistema embebido de bajo costo como es el Raspberry PI.
- A la hora de elegir el controlador embebido, se ha tenido en cuenta el bajo costo de esta placa, el sistema operativo embebido que posee, en este caso Debian, que presta varias herramientas de compilación y edición para FORTE como gcc, g++, etc, y su amplio uso en el ámbito académico actual.

Por otro lado, en la tesis de investigación realizada por (Mercé, 2015) con el tema "Desarrollo de un sistema de control en red mediante el estándar IEC-61499" de la Universidad Jaume I, tiene como objeto: "desarrollar un sistema de control distribuido según el estándar IEC-61499 con el que se puedan implementar diferentes algoritmos de control. Con ello se pretende probar el funcionamiento y comprobar las prestaciones que ofrecen este nuevo método de programación y

las últimas versiones de software basadas en él disponibles actualmente". Este estándar consiste en una arquitectura de referencia para el desarrollo de aplicaciones de control con lógica descentralizada. Está basada en bloques funcionales (FB) que se definen como la unidad estructural básica de los modelos. Una de las ventajas de usar este estándar es que permite el diseño centralizado de sistemas de control distribuidos.

De igual manera, este trabajo investigativo aporta con las siguientes conclusiones más relevantes:

- Llevando a cabo este proyecto también se ha demostrado la gran utilidad y buenas prestaciones que ofrecen el estándar IEC-61499 junto con las herramientas de software desarrolladas hasta la fecha para programar algoritmos de control de forma sencilla, rápida y eficaz.
- Sí que cabe comentar que con la utilización de la herramienta de desarrollo 4DIAC-IDE, aunque permite desarrollar sistemas de control muy capaces y variados, se tiene el inconveniente de no contar con un entorno HMI desarrollado, razón por la que ha sido necesario utilizar un osciloscopio y algún circuito auxiliar para comprobar el funcionamiento del controlador. Hay que tener en cuenta que tanto la forma de implementar modelos según el estándar IEC-61499 como las herramientas de software actualmente existentes para este fin, están en proceso de mejora. Otras herramientas de pago, a diferencia de 4DIAC-IDE, sí que disponen de un entrono HMI más desarrollado, como es el caso de ntxOne.

Mientras que en la tesis de investigación realizada por (Mata, 2005) con el tema "Sistema Robótico Distribuido CORBA: Caso de Estudio Robot Motoman UP6-Edición Única" del Tecnológico de Monterrey propone: "el desarrollo de componentes envolventes basado en tecnologías orientadas a objetos, permite implementar de una manera más simple y con un tiempo de desarrollo menor, otras marcas de brazos manipuladores robóticos de proveedores diversos".

Aportando con las siguientes conclusiones más relevantes:

- Se desarrolló una aplicación independiente para la operación remota del robot UP6, la cual está implementada en su totalidad en el lenguaje de programación C++.
- Se desarrolló el sistema cliente - servidor en arquitectura distribuida mediante CORBA, para obtener un sistema con todas las ventajas que estos tipos de sistemas proporcionan.

(Romero, 2016) en la tesis de investigación con el tema "Diseño e implementación de un sistema de control distribuido en el banco de pruebas neumático de la Escuela de Ingeniería Mecánica de la Universidad del Valle con base en la norma IEC-61499" de la Universidad del Valle propone: "demostrar y emplear las ventajas adquiridas al modelar sistemas complejos de manufactura, con la teoría de sistemas distribuidos en complemento con herramientas que facilitan el entendimiento de la dinámica de los procesos productivos". El propósito del estándar no es presentar metodologías sino una arquitectura y unos modelos para el diseño de aplicaciones de control.

Aportando con las siguientes conclusiones más relevantes:

- Los sistemas de control distribuido han demostrado ser ideales para la implementación de sistemas de control de variables complejas en procesos en la industria. Su concepción se desliga del control centralizado, con el objetivo de dar autonomía a cada una de las partes constitutivas del proceso, al no depender de una única fuente de procesamiento de información.
- Se pudo evidenciar la necesidad de utilizar un estándar especializado en sistemas de control distribuido: IEC-61499. Dicha normativa, está basada en una arquitectura de referencia diseñada para facilitar el desarrollo de aplicaciones con lógica descentralizada, a través de elementos conocidos como bloques funcionales. Sin embargo, la metodología allí descrita no es muy conocida entre los diseñadores de sistemas de control, y se concentra en desarrollo de prototipos en las aulas académicas y centros de investigación.

Y por último, en el artículo científico realizado por (Esteban Querol, 2014) con el tema "Norma IEC-61499 para el control distribuido. Aplicación al CNC" propone el desarrollo de un prototipo que permita la investigación en el campo de los sistemas de fabricación flexibles aplicando la norma IEC-61499 para el desarrollo de sistemas de control distribuidos.

Las conclusiones que aportó este estudio fueron:

- El estándar IEC-61499 aporta a los desarrolladores de software de control una arquitectura de referencia para poder diseñar más fácilmente las aplicaciones de control distribuido, gracias a esta arquitectura es posible llegar a implantar controles holónicos o multi-agente que hasta ahora permanecían en un plano teórico. Las posibilidades y ventajas de adoptar este tipo de arquitectura son múltiples y variadas, ente otras, podríamos destacar la capacidad de pasar del diseño a la implementación de forma

directa, a través del toolchain apropiado, o la capacidad de usar dispositivos embebidos y PCs para el control.

- Actualmente existe numerosas herramientas y tecnología que permiten la aplicación de la norma IEC-61499 al caso de los CNC distribuidos. En ese sentido, la existencia de los CNC abiertos permiten de forma efectiva la integración de aplicaciones desarrolladas bajo el paradigma IEC-61499 con las funcionalidades básicas de un control CNC, posibilitando el desarrollo de CNCs adaptables e inteligentes. Unos CNCs que además, al programarse mediante FBs pueden integrarse en un sistema de control distribuido basado basados en esta misma tecnología. Una opción que puede mejorar la flexibilidad, por ejemplo a nivel de rutas y secuencias, de los que los actuales Sistemas de Fabricación Flexible.

En resumen, estas investigaciones se basan en la coincidencia de las técnicas de búsqueda de nuevas metodologías para producir sistemas distribuidos bajo la norma IEC-61499, pero no se centran en la integración de bloques funcionales (FB), 4DIAC-IDE y el runtime FORTE para el control de manipuladores móviles.

2.2 Bases Teóricas

2.2.1 Sistemas Distribuidos

Un Sistema de Control Distribuido (SCD) es aquel que usualmente se refiere a procesos de manufactura, de grandes industrias o cualquier tipo de Sistema Dinámico, son fundamentalmente aquellos que operan en tiempo continuo y en el que los elementos de control no se encuentran centralizados en una locación específica, sino que se encuentran instalados a través de todo el sistema, en donde cada componente o subsistema es controlado por uno o más controladores, con el fin de utilizar uno de ellos para uno o más lazos de regulación en el propio sistema. Usualmente se emplea un solo canal de alta velocidad que también los monitorea y lleva a cabo el control integral de toda la infraestructura, los SCD se caracterizan por este atributo informático cuya estructura presenta una jerarquización muy marcada (Strasser & Froschauer, 2012).

La Figura 1-2 muestra la arquitectura básica de un SCD, se trata de una topología similar a la de un sistema SCADA. Existen varias unidades de control comunicadas entre sí que realizan las tareas del sistema, de esta forma, en caso de alguna falla dentro del sistema será posible la transmisión de la ejecución de las tareas correspondientes a otro controlador, con esto se logra evitar que una sola falla afecte el proceso completo en una planta (Jakovljevic, Mitrovic, & Pajic, 2017).

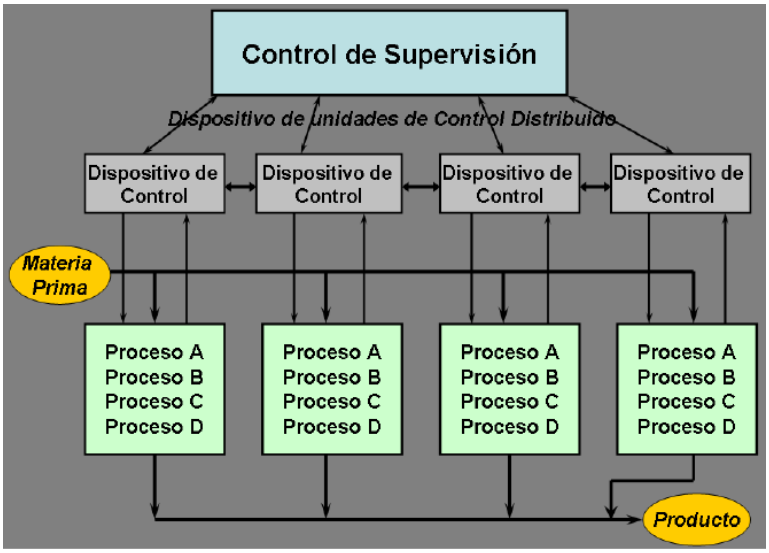

Figura 1-2: Arquitectura básica de un SCD

Fuente: https://docplayer.es/16956349-Capitulo-i-sistemas-de-control-distribuido-scd.html

2.2.1.1 Ventajas

Estos sistemas aportan diversas ventajas con respecto a otros ya que se desarrollan y diseñan a base de módulos seccionados que bien, pueden ser de hardware o de software que simplifica un cambio interno en su arquitectura y facilita la ubicación de fallas o averías, cuentan con un amplio campo de algoritmos de regulación o control, que generalmente, son seleccionables por medio de menús y también son de fácil mantenimiento.

Entre las principales ventajas de los sistemas distribuidos con respecto a las computadoras centralizadas se encuentran (Florencia Gutiérrez Keen, Adolfo Chércoles, Walter Daniel Zalazar, Jorge García, 2015):

• Economía: Los microprocesadores ofrecen una mejor relación precio/ rendimiento que las computadoras centrales.

• Velocidad: Un sistema distribuido puede tener mayor poder de cómputo que una computadora centralizada individual.

• Distribución inherente: Implica que un sistema distribuido puede emplear aplicaciones instaladas en computadoras remotas.

• Confiabilidad: El sistema es consistente, aun si una computadora del sistema deja de funcionar.

• Crecimiento proporcional: Cada vez que se requiera mayor poder de cómputo en el sistema, solo se pueden adicionar los incrementos de cómputo requeridos.

2.2.1.2 Desventajas

A pesar de los diferentes beneficios que introducen los sistemas distribuidos, todavía existen diferentes retos que deben ser resueltos como los siguientes (Florencia Gutiérrez Keen, Adolfo Chércoles, Walter Daniel Zalazar, Jorge García, 2015):

• Software: Gran parte del software para sistemas distribuidos está aún en desarrollo.

• Redes: Los problemas de transmisión en las redes de comunicación todavía son frecuentes en la transferencia de grandes volúmenes de datos (por ejemplo, multimedia).

• Seguridad: Se necesitan mejores esquemas de protección para mejorar el acceso a información confidencial o secreta.

• Tolerancia a fallas: Las fallas operativas y de componentes aún son frecuentes.

2.2.1.3 Desafíos

• ***Heterogeneidad de los componentes.-*** La interconexión, sobre todo cuando se usa Internet, se da sobre una gran variedad de elementos hardware y software, por lo cual necesitan de ciertos estándares que permitan esta comunicación (Kadera, Vrba, Biffl, Andrén, & Strasser, 2015).

• ***Extensibilidad.-*** Determina si el sistema puede extenderse y reimplementarse en diversos aspectos (añadir y quitar componentes). La integración de componentes escritos por diferentes programadores es un auténtico reto.

• ***Seguridad.-*** Reviste gran importancia por el valor intrínseco para los usuarios. Tiene tres componentes (Kadera et al., 2015):

- Confidencialidad.- Protección contra individuos no autorizados.
- Integridad.- Protección contra la alteración o corrupción.
- Disponibilidad.- Protección contra la interferencia con los procedimientos de acceso a los recursos.

• ***Escalabilidad.-*** El sistema es escalable si conserva su efectividad al ocurrir un incremento considerable en el número de recursos y en el número de usuarios.

• ***Tratamiento de Fallos.-*** La posibilidad que tiene el sistema para seguir funcionando ante fallos de algún componente en forma independiente. Para esto se tiene que tener alguna alternativa de solución. Algunos fallos son detectables, por ejemplo, usando comprobaciones (Convention, 2014).

- Enmascaramiento de fallos. Algunos fallos detectados pueden ocultarse o atenuarse.
- Tolerancia de fallos. Sobre todo en Internet se dan muchos fallos y no es muy conveniente ocultarlos, es mejor tolerarlos y continuar. Ej: Tiempo de vida de una búsqueda.
- Recuperación frente a fallos. Tras un fallo se deberá tener la capacidad de volver a un estado anterior.
- Redundancia. Se puede usar para tolerar ciertos fallos (DNS, BD, etc.)

• ***Concurrencia.*** Compartir recursos por parte de los clientes a la vez.

• ***Transparencia.*** Es la ocultación al usuario y al programador de aplicaciones de la separación de los componentes en un sistema distribuido. Se identifican 8 formas de transparencia:

- De Acceso. Se accede a recursos locales y remotos de forma idéntica.
- De ubicación. Permite acceder a los recursos sin conocer su ubicación.
- De concurrencia. Usar un recurso compartido sin interferencia.
- De replicación. Permite utilizar varios ejemplares de cada recurso.
- Frente a fallos. Permite ocultar los fallos.
- De movilidad. Permite la reubicación de recursos y clientes sin afectar al sistema.
- De prestaciones. Permite reconfigurar el sistema para mejorar las prestaciones según su carga.
- Al escalado. Permite al sistema y a las aplicaciones expandirse en tamaño sin cambiar la estructura del sistema o los algoritmos de aplicación.

2.2.2 Estándar IEC-61499

El estándar IEC-61499 representa una solución de componentes para sistemas de automatización industriales distribuidos destinados a la portabilidad, interoperabilidad, reutilización y la reconfiguración de aplicaciones distribuidas (Catalan, Serna, Blesa, Rams, & Colom, 2011). Hay

varios tipos de lenguajes de programación que trabajan con este estándar, uno de los más empleados es el bloque de funciones. Un bloque de funciones en general, proporciona una interfaz para eventos de E/S de datos. Por lo tanto, los bloques de función permiten metodologías de diseño modular (Catalán, Serna, Blesa, & Zaragoza, 2010).

IEC-61499 proporciona (Hussain & Frey, 2005):

1. Enfoque de modelo genérico para aplicaciones de sistemas distribuidos.
2. Concepto de bloques de funciones.
3. La separación de datos y flujo del evento.

2.2.2.1 Objetivos

Portabilidad: la capacidad de las herramientas de software para aceptar e interpretar correctamente elementos de la biblioteca producidos por otras herramientas de software (Suinder et al., 2006).

Configurabilidad*:* la capacidad de los dispositivos y sus componentes de software a configurar (seleccionados, ubicaciones asignadas, interconectadas y parametrizadas) por múltiples herramientas de software.

Interoperabilidad: la capacidad de los dispositivos de funcionar juntos para realizar las funciones especificadas por uno o más aplicaciones distribuidas.

Figura 2-2: Esquema de los objetivos de la norma IEC-61499 para el SCD

Fuente: (Miguel, n.d.)

En la Figura 2-2 se puede ver el impacto que tienen los tres principios comentados anteriormente y por los que se rige la norma sobre los SCD, minimizando los inconvenientes que este tipo de control presenta.

2.2.2.2 Estructura del estándar

En cuanto a la estructuración del estándar, este se encuentra distribuido en 4 partes, siendo cada una independiente de las otras y con objetivos diferentes. A continuación resumimos estas 4 partes y sus objetivos (Cengic, 2010).

Parte 1: Define la arquitectura de referencia para el desarrollo, reutilización e implementación de los bloques funcionales. Se definen, entre otros, los conceptos de bloque funcional básico, bloque funcional compuesto, aplicación, interfaz, dato, señal evento…

Parte 2: En esta parte se definen los requisitos que deben cumplir las herramientas de desarrollo para poder desarrollar las arquitecturas propuestas en la parte 1. No obstante, los requisitos definidos no permiten por ellos mismo garantizar la interoperabilidad, portabilidad y la configurabilidad. Entre otras cosas, se definen los esquemas XML para el intercambio de ficheros entre las distintas herramientas.

Parte 3: Información didáctica que tiene por objeto mostrar las funcionalidades y ejemplos de aplicación de la IEC-61499 en un amplio conjunto de dominios.

Parte 4: El objetivo de esta parte es definir unos perfiles de compilación de forma que se puedan cumplir los objetivos de la norma IEC-61499 en los dispositivos compatibles y las herramientas de desarrollo.

2.2.2.3 Arquitectura

En esta sección desarrollaremos algunos de los conceptos clave para entender la arquitectura del estándar IEC-61499.

Modelo de Sistema

Contiene a los demás modelos siendo el más alto en la estructura jerárquica. En él se encuentran los dispositivos y las diferentes aplicaciones que conforman el sistema de control, así como la

relación existente entre estos de manera conforme a lo definido en el estándar, Figura 3-2 (Vyatkin, 2009a).

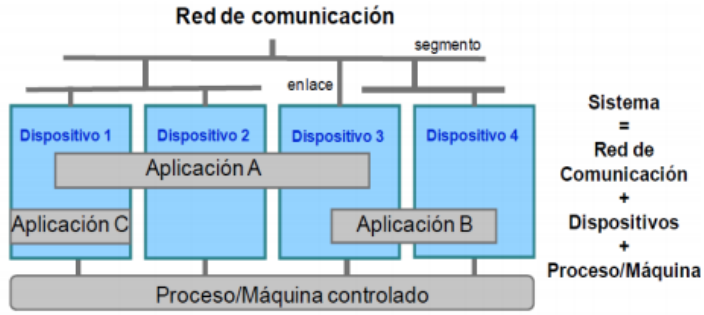

Figura 3-2: Modelo de Sistema

Fuente: (Doctoral, 2018)

Modelo de Dispositivo

El modelo de dispositivo que se observa en la Figura 4-2, representa la entidad física que va a ejecutar nuestra red de bloques funcionales. Lo podríamos asimilar a un sistema embebido o a un PC (Vyatkin, 2009a). El dispositivo es capaz de comunicarse con otros dispositivos o con sus periféricos a través de interfaces de comunicación. Al mismo tiempo es igualmente capaz de comunicar con la red de bloques funcionales a través de un interfaz de proceso.

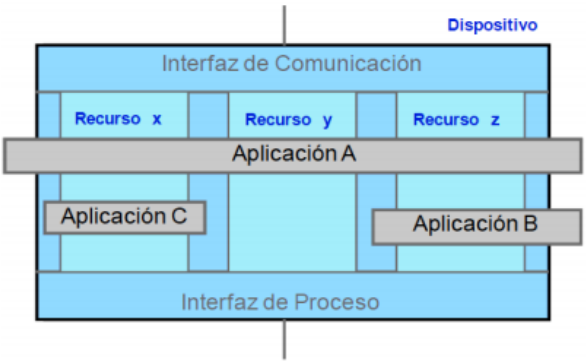

Figura 4-2: Modelo de Dispositivo

Fuente: (Doctoral, 2018)

Modelo de Recurso

Un recurso se puede definir como una parte del dispositivo designada a ejecutar un conjunto de bloques funcionales determinado, pertenecientes o no a la misma aplicación. El concepto de recurso podría ser visto como el conjunto de "recursos del dispositivo" dedicados exclusivamente al conjunto de bloques funcionales asignados a ese recurso. Cada recurso tiene la característica de ser independiente de los demás, esto implica que, un parte de los bloques funcionales que se

están ejecutando en un determinado dispositivo se pueden parar, ejecutar o redefinir sin que esto afecte a los otros bloques definidos en el resto de recursos del dispositivo como se muestra en la Figura 5-2 (Dai & Vyatkin, 2010).

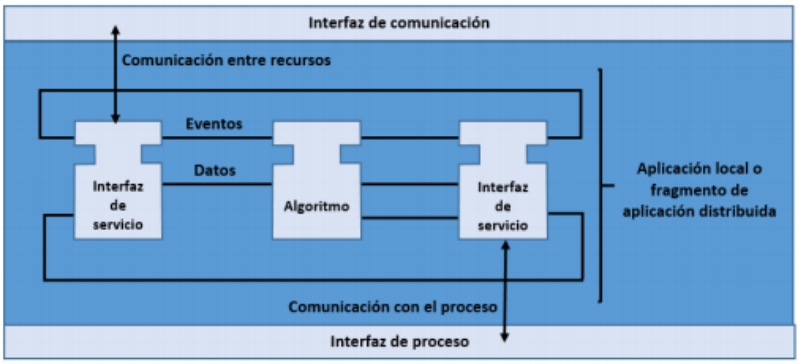

Figura 5-2: Modelo de Recurso

Fuente: (Yoong, Roop, Vyatkin, Salcic, & Member, 2009)

Modelo de Aplicación

Una aplicación es un conjunto de bloques funcionales interconectados entre si y unidos por un flujos de señales y datos (Yoong et al., 2009). Las aplicaciones como hemos visto se pueden distribuir entre diferentes recursos, del mismo dispositivo o de dispositivos diferentes, Figura 6-2.

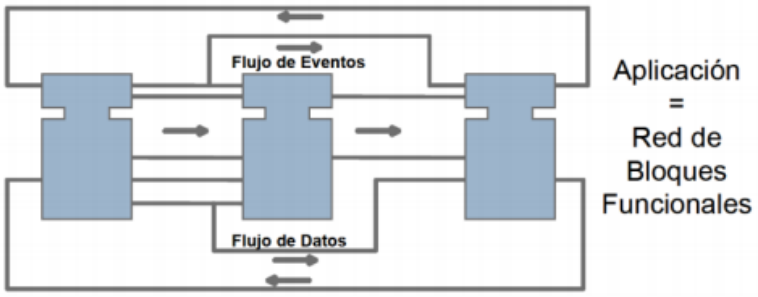

Figura 6-2: Modelo de Aplicación

Fuente: (Doctoral, 2018)

Modelo de Bloque Funcional

Es el elemento más pequeño en un sistema de control distribuido. El FB consiste en una cabeza que está conectada al flujo de eventos. Acepta entrada de eventos y genera salida de eventos. El cuerpo está conectado al flujo de datos, acepta datos de entrada y genera datos de salida. El comportamiento dinámico del FB está definido por la Gráfica de Control de ejecución (siglas en

inglés: ECC, Execution Control Chart) que procesa entrada de eventos y genera salida de eventos (Sünder, Zoitl, Christensen, Colla, & Strasser, n.d.).

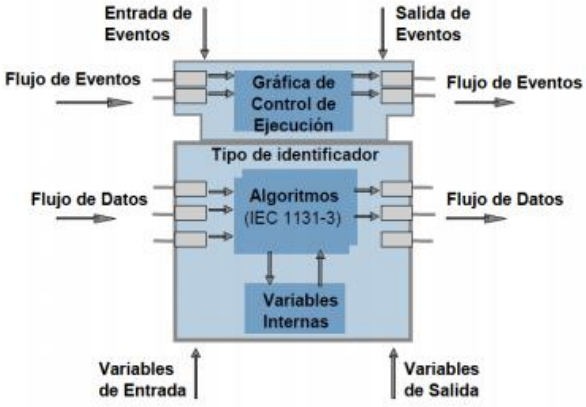

Figura 7-2: Modelo de Bloque Funcional

Fuente: (Doctoral, 2018)

Un FB en el IEC 61499-1, se mantiene pasivo hasta que es disparado por una entrada de evento, es decir, estos eventos son usados para activar un bloque funcional (Mizuya, n.d.). El FB ejecuta y produce eventos y datos de salida como se representa en la Figura 7-2.

En cuanto al comportamiento e interfaz con el resto de elementos de la norma existen tres tipos de FB:

FB Básico: los bloques de funciones básicos (se nombrará por las siglas BFB del inglés: Basic Function Block). Los BFB se caracterizan por tener una estructura y comportamiento, están definidos por medio de un gráfico de control de ejecución (se nombrará por sus siglas ECC del inglés: Execution Control Chart) (Dai & Vyatkin, 2010).

FB Compuesto: los bloques de funciones compuestos (se nombrará por sus siglas CFB del inglés: Composite Function Block). Los CFB se caracterizan por tener su estructura interna compuesta por instancias de otros FB y sus respectivas conexiones, su comportamiento queda definido así en función de FB, la unidad básica de programación de esta norma (Miguel, n.d.).

FB Interfaz de Servicio: Los bloques de funciones de interfaz de servicio (se nombrará por sus siglas SIFB del inglés: Service Interface Function Block). Los SIFB proveen una interfaz entre la norma y servicios externos como puede ser una comunicación entre FB en diferentes dispositivos (Zoitl, Smodic, & Sünder, 2006).

2.2.2.4 Entorno de desarrollo y ejecución de la Norma IEC-61499

El entorno de desarrollo para la norma denominado IDE (Integrated Development Environment), consta 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 (Vyatkin, 2009b).

A continuación, se describe el entorno de desarrollo y ejecución de código abierto más común en el ámbito académico para el empleo de la norma IEC-61499.

4DIAC-IDE

Es un entorno de desarrollo de código abierto basado en la plataforma de Eclipse. Entre las características más relevantes de este entorno podemos destacar la fácil distribución de las aplicaciones entre diferentes recursos de forma gráfica (Frey & Hussain, 2006). Puede ejecutar bloques funcionales de tipo FB básicos, FB compuestos, FB interfaces de servicio SIFBs y adaptadores. Cabe recalcar las funcionalidades de depuración y prueba por medio de un display online que permite fijar y leer los valores de las variables de manera remota (García, Irisarri, & Pérez, 2015).

FORTE

Desarrollada por el mismo equipo que 4DIAC, es una plataforma de ejecución de código abierto implementada en C++, está enfocada para ejecutarse en sistemas de tiempo real y pequeños sistemas embebidos (Sarkar, 2015). Posee los mecanismos necesarios para asegurar la ejecución limitada 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. Otro de los puntos fuertes de FORTE es que ha sido diseñada para ser independiente de la plataforma a utilizarse, puede ser ejecutada en diferentes tipos de hardware y sistemas operativos. En la actualidad FORTE se ha llevado a diferentes sistemas operativos como POSIX, windows32, threadX y eCos (Para et al., 2014).

2.2.3 Manipulador Móvil

El robot utilizado en esta aplicación es un KUKA youBot, ver Figura 8-2. Consiste de un brazo de 5 DoF con una pinza de dos dedos, montado en una plataforma omnidireccional. Fue desarrollado para su uso en investigación y educación. La comunicación de la base y el brazo del youBot están habilitados a través de EtherCAT que tiene capacidades en tiempo real (Locomotec, 2011).

Figura 8-2: Manipulador Móvil Omnidireccional Brazo KUKA youBot

Fuente: https://www.kuka.com/es-mx

2.2.3.1 Hardware

Como se muestra en la Figura 9-2, el manipulador se compone de cuatro partes principales:

1. Gripper de agarre con dos dedos
2. Brazo robótico de cinco grados de libertad
3. Plataforma omnidireccional
4. Zona de carga

Figura 9-2: Partes manipulador KUKA youBot

Fuente: https://www.kuka.com/es-mx

Brazo KUKA youBot

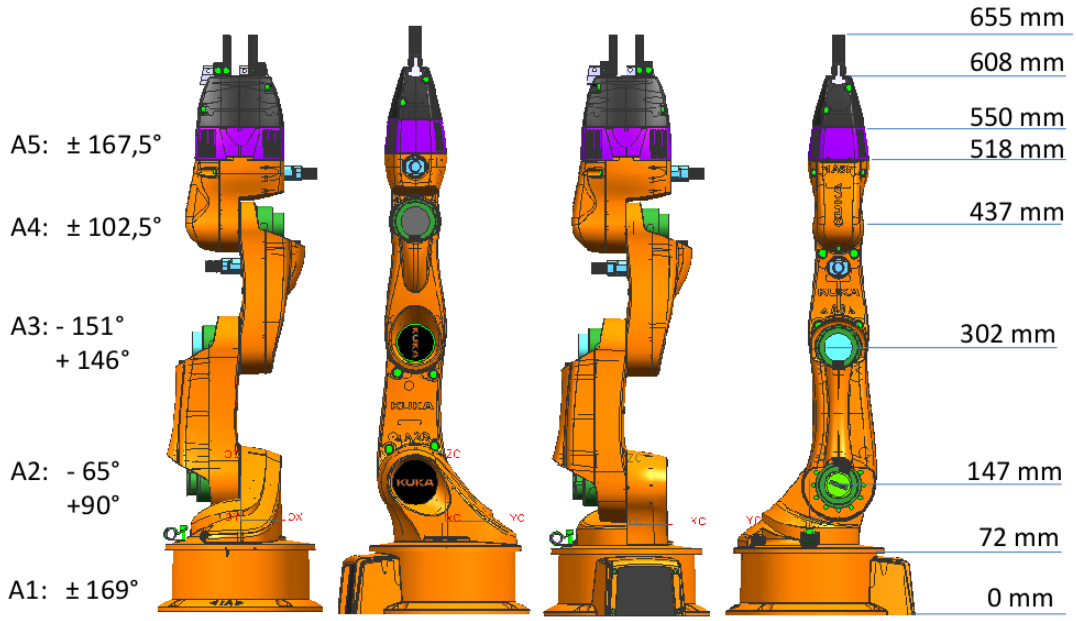

Figura 10-2: Panorámica de la estructura cinemática del brazo. La figura ilustra uniones con límites y la longitud de los enlaces entre las uniones.

Fuente: (Mirelez-Delgado, Morales-Díaz, Ríos-Cabrera, & Pérez-Villeda, 2015)

El brazo KUKA youBot se muestra en la Figura 10-2, se compone de cinco juntas rotatorias y una pinza de dos dedos como un efector final (Keiser, 2013). Cada articulación del brazo es administrado por un microcontrolador independiente (Trinamic TMCM -1632) supervisado por la Unidad de control (CU). Ellos efectúan el control del motor para cada articulación de bajo nivel y administran los sensores de unión. La comunicación entre la CU y el microcontrolador es a través de la comunicación EtherCAT (Di Napoli, Filippeschi, Tanzini, & Avizzano, 2016).

En la Tabla 1-2 se resume las características generales del Brazo KUKA youBot.

Tabla 1-2: Características Generales Brazo KUKA youBot

Cinemática Serial	5 Ejes
Altura	655 mm
Posición de trabajo	0.513 m^3
Peso	6.3 kg
Carga útil	0.5 kg
Estructura	Molde de magnesio
Repetibilidad de posicionamiento	1 mm

Comunicación	EtherCAT	
Voltaje de conexión	24 V	
Tren de potencia limitable a	80 W	
Datos del Eje	***Distancia***	***Velocidad***
Eje 1 (A1)	+/– 169°	90°/s
Eje 2 (A2)	+ 90°/– 65°	90°/s
Eje 3 (A3)	+ 146°/– 151°	90°/s
Eje 4 (A4)	+/– 102°	90°/s
Eje 5 (A5)	+/– 167°	90°/s
Gripper	Desmontable, 2 dedos	
Golpe de agarre	20 mm	
Distancia Gripper	70 mm	

Fuente: (Bischoff, Huggenberger, & Prassler, 2011)

Realizado por: Mafla, Gabriela, 2019

Plataforma KUKA youBot

La plataforma del KUKA youBot cuenta con una mini PC ITX a bordo como unidad de control, la cual posee las siguientes características: procesador Intel Atom Dual-Core, 2GB RAM y 32 GB SSD. Adicional la base cuenta con una batería y cuatro ruedas omnidireccionales con motores (Keiser, 2013). La plataforma puede moverse usando las ruedas omnidireccionales KUKA especiales. Es compatible con la comunicación EtherCAT. La Figura 11-2 ilustra el marco base adjunto, los valores positivos para la rotación alrededor del eje z dan como resultado un movimiento en sentido contrario a las agujas del reloj, como lo indica la flecha azul.

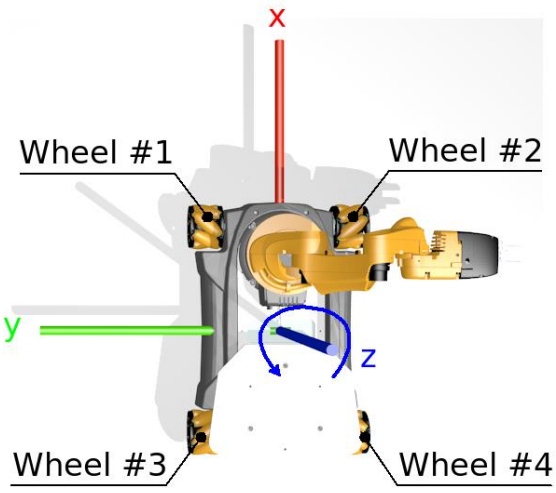

Figura 11-2: Descripción general de la base KUKA youBot

Fuente: https://www.kuka.com/es-mx

En la Tabla 2-2 se resume las características generales de la plataforma KUKA youBot.

Tabla 2-2: Características Generales Plataforma KUKA youBot

Cinemática omnidireccional	4 ruedas KUKA omnidireccionales
Longitud	580 mm
Anchura	380 mm
Altura	140 mm
Despeje	20 mm
Peso	20 kg
Carga útil	20 kg
Estructura	Acero
Velocidad	0.8 m/s
Comunicación	EtherCAT
Voltaje de conexión	24 V

Fuente: (Bischoff et al., 2011)

Realizado por: Mafla, Gabriela, 2019

2.2.3.2 Software

El robot KUKA youBot usa el sistema operativo Ubuntu, cuenta con una API (Application Programming Interface) desarrollada en C++, además se puede usar con ROS (Robot Operating System) (Zhang & Zhou, 2013). Las librerías que proporciona el fabricante cuentan con soporte para los sistemas operativos Linux y Windows, cuya función es establecer la comunicación a través del lenguaje nativo C++.

El control de nivel alto y bajo es posible a través de varias API desarrolladas por KUKA para ROS. Actualmente, esta biblioteca proporciona las funcionalidades básicas para configurar movimientos y velocidades de la plataforma y del brazo, así como a leer datos de los sensores respectivos (Pierucci et al., 2017).

La infraestructura de comunicación original entre la C.U. y los micro controladores utilizan EtherCAT, un sistema de bus de campo basado en Ethernet adecuado para los requisitos de tiempo real duros y suaves en la tecnología de automatización que necesita tiempos de actualización cortos, con baja fluctuación de la comunicación (Hochgeschwender et al., 2013).

2.2.4 Modelo Cinemático del KUKA youBot

La cinemática de un robot es el estudio de los movimientos de un robot. En un análisis cinemático la posición, velocidad y aceleración de cada uno de los elementos del robot son calculados sin considerar las fuerzas que causan el movimiento (Engineering, 2014).

La cinemática directa del manipulador se expresa de manera clara en la investigación realizada por (Dwiputra, Zakharov, Chakirov, & Prassler, 2014) de la cual se basó el modelo cinemático que se detalla a continuación.

2.2.4.1 Modelo cinemático de la plataforma omnidireccional

El robot manipulador móvil Kuka youBot cuenta con una plataforma omnidireccional de cuatro ruedas tipo Mecanum que permiten el libre desplazamiento del sistema en el plano Cartesiano, sin embargo es necesario considerar la disposición de los rodillos de este tipo de ruedas para obtener un modelo cinemático fidedigno. Las medidas de la plataforma móvil se muestran en la Figura 12-2 y están expresadas en milímetros.

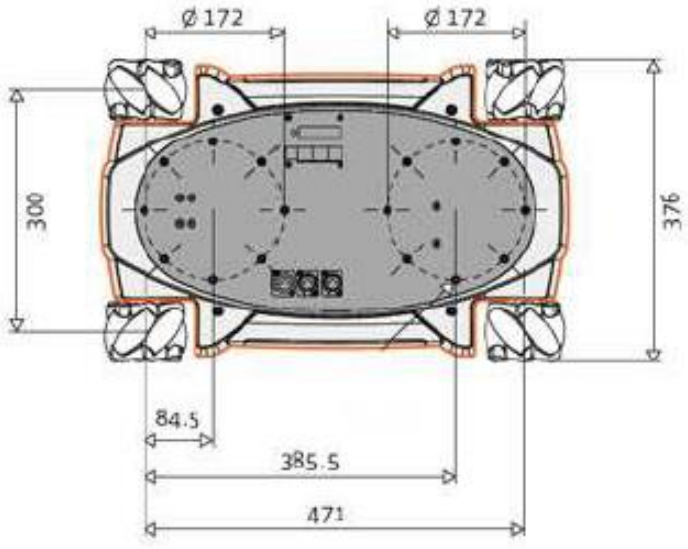

Figura 12-2: Medidas de la plataforma móvil

Fuente: (Mirelez-Delgado et al., 2015)

De acuerdo con (Nagatani, Tachibana, & Tanaka, n.d.), la configuración cinemática de un robot móvil omnidireccional está dada por el conjunto de ecuaciones (1).

$$\dot{x} = v_l = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)$$
$$\dot{y} = v_t = \frac{1}{4}(-d_1 + d_2 + d_3 - d_4)\tan(\alpha_b) \quad (1)$$
$$\dot{\theta}_b = v_a = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)\beta$$

Donde $\dot{x}$ y $\dot{y}$ son la velocidad en los ejes X y Y, o bien las velocidades longitudinal y transversal (v_l) y (v_t), mientras que $\dot{\theta}_b$ es la velocidad angular (v_a) de la plataforma móvil omnidireccional. Por otra parte d_i es la velocidad lineal de cada rueda de la plataforma móvil. Los parámetros α_b y β son calculados experimentalmente debido a que dependen del ángulo en que estén dispuestos los rodillos dentro de las ruedas Mecanum (ver Figura 13-2).

Figura 13-2: Descripción de la rueda tipo Mecanum

Fuente: (Mirelez-Delgado et al., 2015)

El movimiento del robot móvil en el plano dependerá entonces de la combinación de las velocidades lineales de cada rueda de la plataforma, particularmente los movimientos de traslación transversal y rotación se ilustran en la Figura 14-2, donde la plataforma difuminada indica hacia donde se moverá el robot móvil según la combinación de velocidades lineales de las cuatro ruedas.

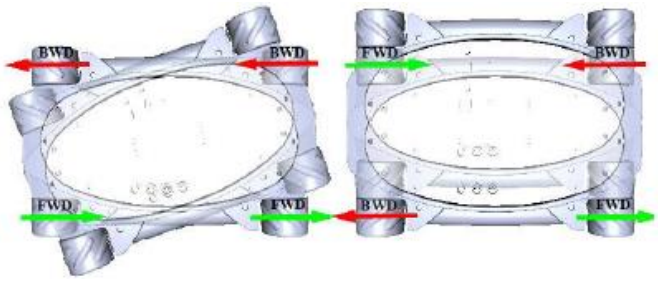

Figura 14-2: Descripción de la rueda tipo Mecanum

Fuente: (Mirelez-Delgado et al., 2015)

2.2.4.2 Modelo cinemático del manipulador de 5 grados de libertad

El brazo manipulador del Kuka youBot es un robot de 5 grados de libertad (todos rotacionales), las medidas de cada eslabón así como sus límites articulares se pueden observar en la Figura 10-2, el efector final es una pinza de dos dedos que le permite sujetar y manipular objetos pequeños. El modelo cinemático del robot manipulador se obtiene mediante el algoritmo de Denavit-Hartenberg (Mirelez-Delgado et al., 2015).

Los marcos de referencia para cada articulación se pueden apreciar en la Figura 15-2 y los parámetros de Denavit-Hartenberg obtenidos para el manipulador se muestran en la Tabla 3-2, mientras que en la Tabla 4-2 se muestran las distancias y ángulos entre los marcos de referencia para el brazo manipulador.

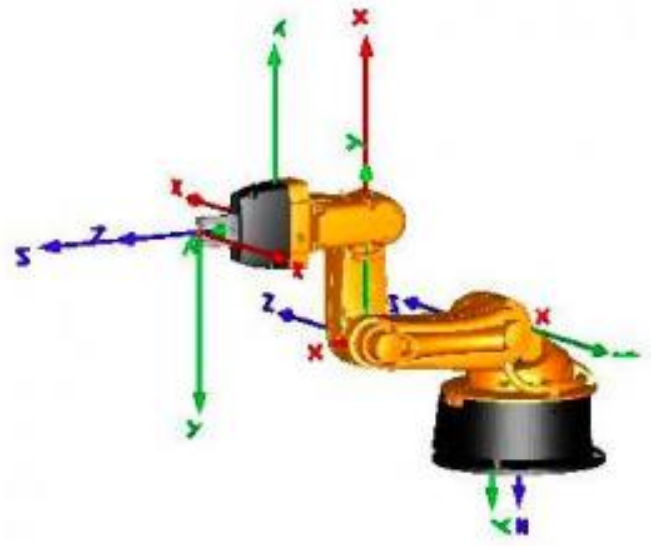

Figura 15-2: Posicionamiento de los marcos de referencia para cada articulación.
Fuente: (Mirelez-Delgado et al., 2015)

Tabla 3-2: Parámetros de Denavit-Hartenberg para el manipulador del youBot

Eslabón	Θ	d	a	α
1	q_1	0.147	0.0330	$\pi/2$
2	q_2	0	0.1550	0
3	q_3	0	0.1350	0
4	$q_{4+\pi/2}$	0	0	$\pi/2$
5	q_5	0.2175	0	0

Fuente: (Mirelez-Delgado et al., 2015)
Realizado por: Mafla, Gabriela, 2019

Tabla 4-2: Cadena cinemática para el manipulador del youBot

	Marco anterior	Traslación (cm)			Rotación (grados)		
		x	y	z	x	y	z
Articulación 1	Base	2.4	0	11.5	180	0	0
Articulación 2	Articulación 1	3.3	0	0	90	0	-90
Articulación 3	Articulación 2	15.5	0	0	0	0	-90
Articulación 4	Articulación 3	0	13.5	0	0	0	0
Articulación 5	Articulación 4	0	11.36	0	-90	0	0
Pinza	Articulación 5	0	0	5.716	90	0	180

Fuente: (Mirelez-Delgado et al., 2015)

Realizado por: Mafla, Gabriela, 2019

Con los parámetros de las Tablas 3-2 y 4-2 es posible obtener las matrices de transformación homogéneas de acuerdo a la ecuación (2).

$$A_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & -S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Donde los términos S y C denotan las funciones sin() y cos() respectivamente. La matriz de transformación total se obtiene de la multiplicación sucesiva de cada una de las matrices homogéneas.

$$T_n^0 = A_1^0 \dots A_n^{n-1}$$

CAPÍTULO III

3 DISEÑO DE LA INVESTIGACIÓN

3.1 Desarrollo de la metodología de investigación

Los métodos en los que se enfrasca la presente investigación esencialmente son:

3.1.1 Métodos Teóricos

- **Documental:** permite recopilar información relacionada al objeto de estudio y cimentar conocimientos específicos referentes al desarrollo de sistemas de control distribuidos bajo la norma IEC-61499.
- **Lógico:** para aplicar la información bibliográfica y experimental a la implementación de un sistema distribuido, desarrollado bajo el concepto de bloques funcionales.

3.1.2 Métodos empíricos

- **Experimental:** para verificar que lo diseñado y programado funcione correctamente.

Para cumplir con los objetivos trazados, se llevarán a cabo las siguientes tareas de investigación:

- Análisis de la composición en hardware y el desplazamiento del robot Kuka youBot.
- Búsqueda de información referente a sistemas de control distribuido bajo la norma IEC-61499.
- Modelado del sistema de control distribuido, a través de elementos conocidos como bloques funcionales, conforme al estándar IEC-61499.
- Aprender el uso de una herramienta de software que permita la simulación del sistema.
- Integración del sistema lógico y físico en el robot KUKA youBot.
- Pruebas de ejecución de movimientos.
- Valoración de resultados.

3.2 Caso de Estudio

El caso de estudio propuesto describe el diseño de un sistema distribuido a través del modelamiento de aplicaciones de control bajo el concepto de bloque funcional (FB) introducido por el estándar IEC-61499, que permite la programación de algoritmos distribuidos de una forma sencilla para el control de manipuladores móviles.

La idea básica de la aplicación es representar un sistema de control robótico como una combinación de subsistemas funcionales desacoplados (Sanfilippo, Hatledal, & Zhang, n.d.); es decir, controlar y manipular de manera independiente la plataforma y el brazo que conforman el robot kuka youBot, como se muestra en la Figura 1-3.

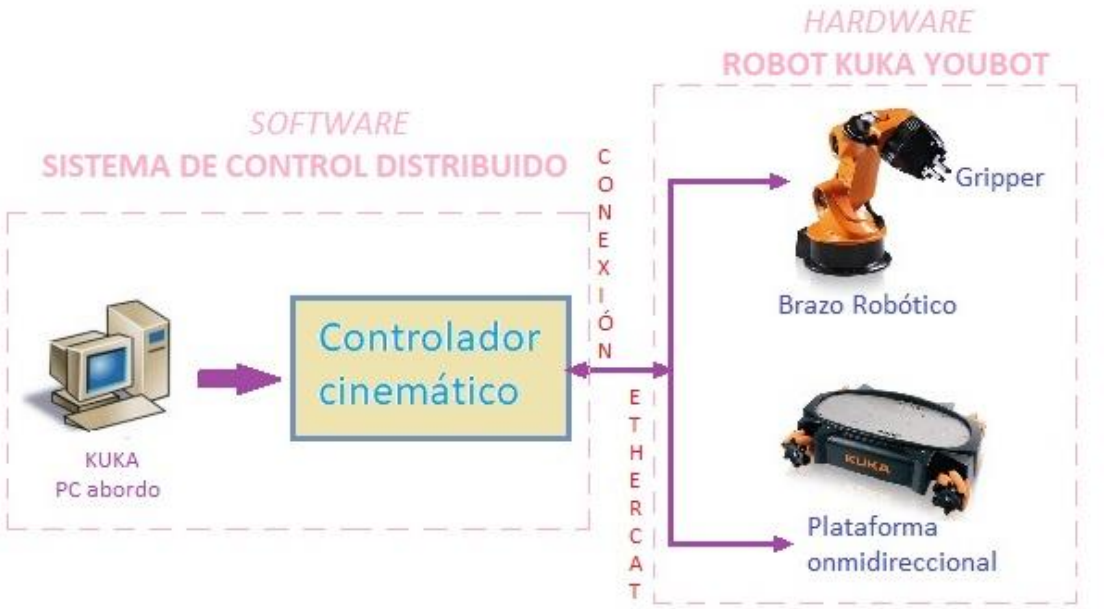

Figura 1-3: Arquitectura de control

Fuente: Mafla, Gabriela, 2019

El sistema de control está formado por una mini *PC ITX*, que se encuentra a bordo de la plataforma del robot KUKA youBot. Éste, recibe a través de conexión EtherCAT la orden enviada por la aplicación de control y establece el comportamiento correspondiente de los sensores y actuadores.

Para realizar la manipulación de la base, se parte del modelo cinemático de la plataforma omnidireccional definida en el apartado 2.2.4.1. En este caso, la plataforma robótica solo se desplazará en el plano xy y solo rotará en el eje z.

Estos robots omnidireccionales, se mueven de una manera distinta a la que se movería un vehículo robótico con unas ruedas normales, dependen de cómo se mueven sus ruedas y a que velocidades para moverse hacia una dirección u otra. El controlador cinemático, como se ha visto y se verá más a detalle en la implementación del FB YouBotBase_WRITE, devuelve 3 valores en un vector, los dos primeros valores serán la velocidad en el eje x y en el eje y respectivamente y el tercer valor será la velocidad angular del robot, para entenderse mejor se puede observar la Figura 2-3.

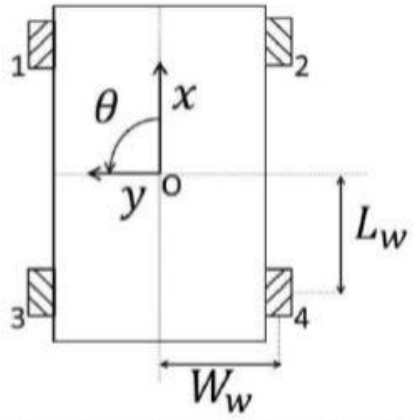

Figura 2-3: Esquema ejes del KUKA youBot
Fuente: (Keiser, 2013)

De la misma manera para la manipulación del brazo robótico, se parte del modelo cinemático del manipulador de cinco grados de libertad detallado en el apartado 2.2.4.2. El controlador cinemático, como se verá más a detalle en la implementación de los FBs YouBotArm_Write, YouBotArm_Read y Controller_Ptp, estos en conjunto consiguen la manipulación del brazo robótico. El FB de lectura devuelve 6 valores en un vector, los cinco primeros valores serán los ángulos correspondientes a cada una de las articulaciones y el sexto valor será el de apertura de la pinza.

3.3 Diagrama de flujo del software

El diagrama de flujo general, muestra de manera gráfica los pasos a seguir en el desarrollo del sistema de control distribuido, ver Figura 3-3.

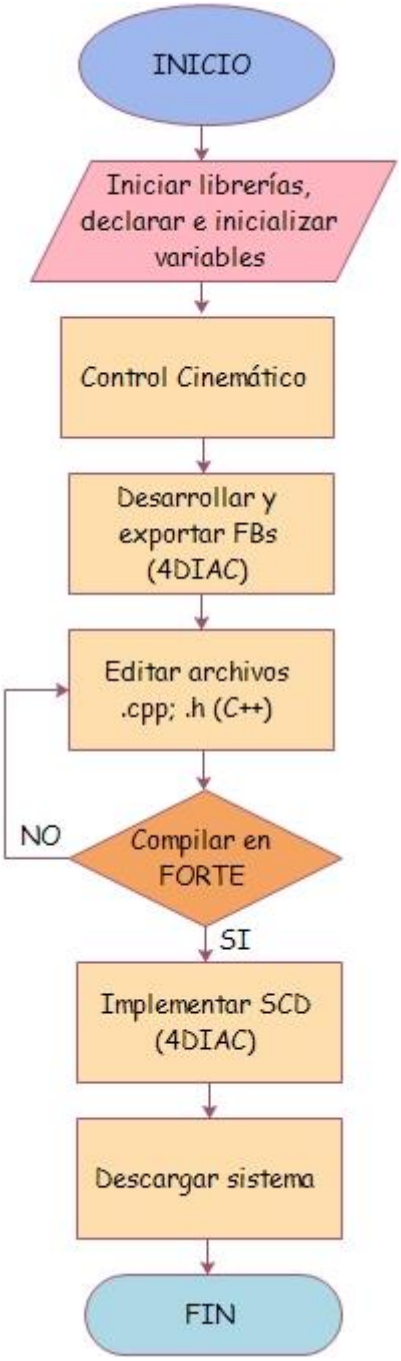

Figura 3-3: Diagrama de flujo del software

Fuente: Mafla, Gabriela, 2019

3.4 Metodología de Diseño de FBs

Como herramienta de desarrollo se ha optado por 4DIAC-IDE, concretamente la versión 1.8.4. Esta aplicación, además de tener buenas prestaciones, es de código libre y compatible con la mayoría de las herramientas actualmente existentes, siendo este uno de los objetivos del estándar IEC-61499.

En la Figura 4-3 se describe de manera gráfica el desarrollo de FBs basados en C++ bajo el sistema operativo Windows y compilados en Linux Ubuntu que dispone el robot Kuka YouBot.

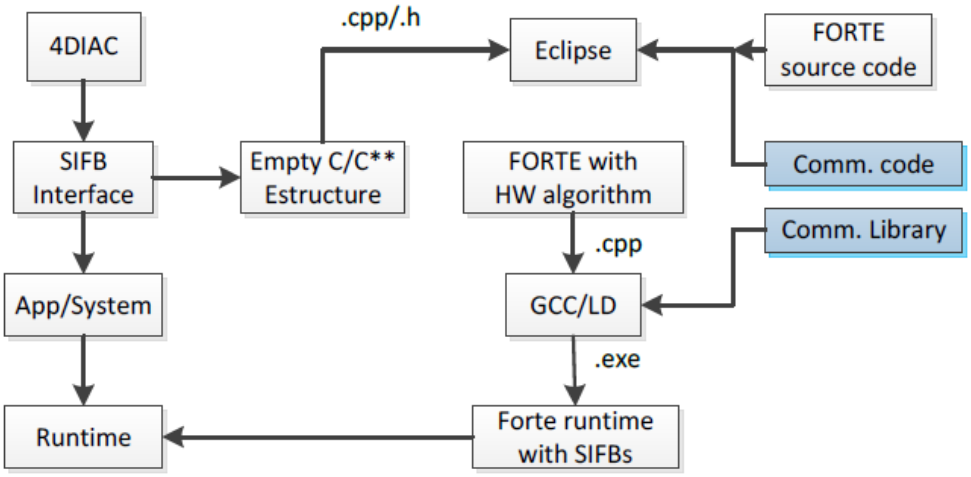

Figura 4-3: Metodología de diseño de FBs

Fuente: (Doctoral, 2018)

3.5 FBs Desarrollados

Siguiendo la metodología antes descrita y utilizando la herramienta de diseño 4DIAC-IDE, se desarrollan algunos FBs que permiten interactuar las aplicaciones implementadas en IEC-61499 con las variables de control del robot Kuka YouBot, es decir controlar sensores y actuadores. En cada FB creado, se identifica las entradas y salidas de datos, se configura el nombre y el tipo de dato de la variable a utilizar y se enlaza con el evento a ejecutar. La creación de un FB se detalla en el Anexo B.

En cuanto al aspecto del control de ejecución, todos reciben un evento de entrada para inicializar el FB, uno para ejecutarlo y otro para devolver los valores internos a cero (hacer un reset) y como eventos de salida todos tienen un evento para indicar que ha finalizado la etapa de inicialización y, como mínimo, otro evento de salida que indica que se ha ejecutado el FB.

Una vez explicadas las características comunes de todos los FBs de control, a continuación se describe las particularidades de cada uno de los FBs desarrollados.

3.5.1 FB YouBotBase_WRITE

El FB desarrollado "*YouBotBase_WRITE"*, como se muestra en la Figura 5-3 es de tipo SIFB, porque nos ofrece un servicio de comunicación entre la aplicación y la plataforma, con lo cual se logra llevar a cabo la manipulación de la base KUKA youBot.

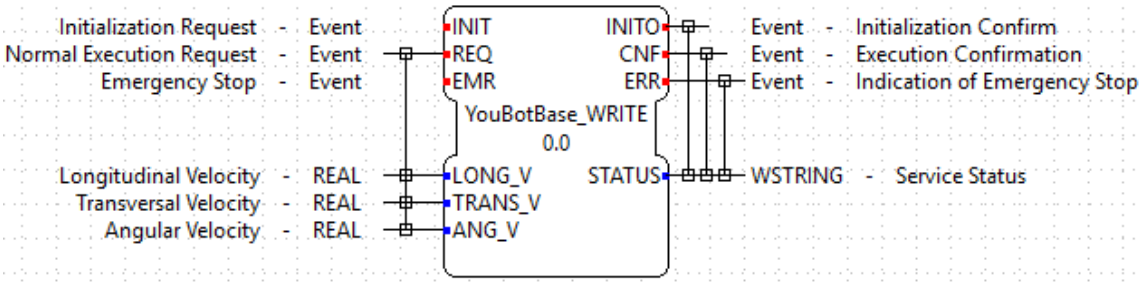

Figura 5-3: FB encargado de la manipulación de la plataforma KUKA youBot

Fuente: Mafla, Gabriela, 2019

Los actuadores que se controlan en este FB son los siguientes:

- *Longitudinal:* desplazamientos hacia adelante y atrás.
- *Transversal:* desplazamientos a la derecha e izquierda.
- *Angular:* rotación alrededor del eje central.

La configuración del bloque de función se describe a continuación en la Tabla 1-3.

Tabla 1-3: Características del FB YouBotBase_WRITE

EVENTOS DE ENTRADA	EVENTOS DE SALIDA
INIT: este evento de entrada sirve para inicializar, testear la comunicación y parámetros de funcionalidad del bloque, esto se realizará una sola vez al arrancar la plataforma del robot.	**INITO:** evento de salida que sirve para la confirmación de la inicialización.
REQ: inicializa la trasmisión de las variables de entrada (LONG_V, TRANS_V, ANG_V).	**CNF:** evento de salida que sirve para indicar el fin de la etapa de ejecución.
EMR: evento desencadenante de la finalización de cualquier operación que se está generando.	**ERR:** evento de salida activado con el paro de emergencia.

DATOS DE ENTRADA	DATOS DE SALIDA
LONG_V (REAL): valores de tipo real de la velocidad longitudinal, utilizados como referencia para manipular los actuadores.	**STATUS (WSTRING):** devuelve como valor de cadena el texto correspondiente al estado o FB, siendo las opciones: inicializado, operativo y parado.
TRANS_V (REAL): valores de tipo real de la velocidad transversal, utilizados como referencia para manipular los actuadores.	
ANG_V (REAL): valores de tipo real de la velocidad angular, utilizados como referencia para manipular los actuadores.	

Fuente: Realizado por: Mafla, Gabriela, 2019

3.5.2 FB YouBotArm_WRITE

Este FB denominado "*YouBotArm_WRITE*", como se muestra en la Figura 6-3, al igual que el anterior es de tipo SIFB, permite escribir los ángulos de las juntas del robot KUKA youBot. El movimiento del brazo se controla de forma individual en cada articulación, posee valores de desplazamiento angular definidos y a su vez sensores que monitorean la posición actual, con estos dos parámetros se realiza el control en lazo cerrado. Bajo la misma metodología se realiza el control de la pinza o gripper, ya que al ser el elemento terminal del brazo su control se incluye en este FB.

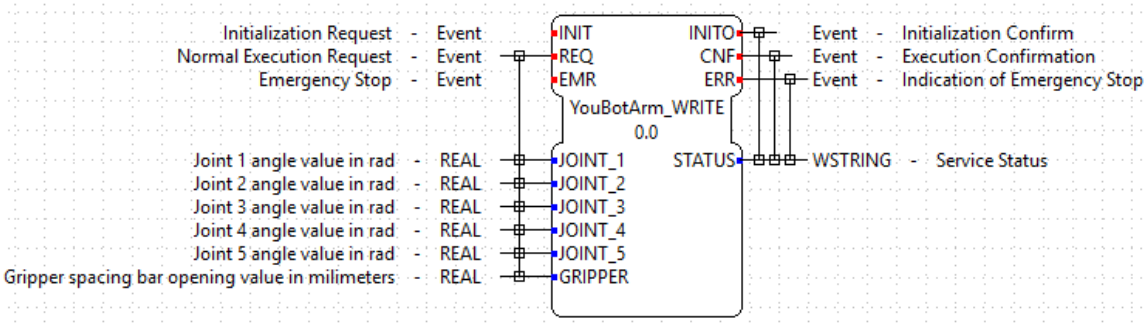

Figura 6-3: FB de escritura encargado de la manipulación del brazo KUKA youBot

Fuente: Mafla, Gabriela, 2019

El bloque de función creado toma los estados de los cinco (5) sensores del brazo, ejecuta el algoritmo programado y activa la respectiva salida para cada actuador, según lo establecido en la etapa de control.

Podemos controlar la secuencia mediante el ingreso manual de los ángulos de las articulaciones, teniendo presente los desplazamientos de cada una para no cometer errores.
La configuración del FB se describe en la Tabla 2-3.

Tabla 2-3: Características del FB YouBotArm_WRITE

EVENTOS DE ENTRADA	**EVENTOS DE SALIDA**
INIT: evento de entrada que sirve para inicializar el bloque.	**INITO:** evento de salida que sirve para la confirmación de la inicialización.
REQ: evento desencadenante de la operación correspondiente a un ciclo normal de trabajo.	**CNF:** evento de salida que sirve para indicar el fin de la etapa de ejecución.
EMR: evento desencadenante de la finalización de cualquier operación que se está generando.	**ERR:** evento de salida activado con el paro de emergencia.
DATOS DE ENTRADA	**DATOS DE SALIDA**
JOINT_1 – JOINT_5 (REAL): valores de tipo real de los ángulos de unión deseados, utilizados como referencia para manipular los actuadores.	**STATUS (WSTRING):** devuelve como valor de cadena el texto correspondiente al estado o FB, siendo las opciones: inicializado, operativo y parado.
GRIPPER (REAL): valores de tipo real de la longitud de apertura deseada de la pinza, utilizada como referencia para manipular el actuador.	

Fuente: (Garcia, Salinas, Perez, Salazar L., & Garcia, 2019)
Realizado por: Mafla, Gabriela, 2019

3.5.3 FB YouBotArm_READ

Este bloque de función como se muestra en la Figura 7-3, tiene una estructura y comportamiento idénticos al FB anterior, salvo por el hecho que permite leer los ángulos de las juntas del robot KUKA youBot.

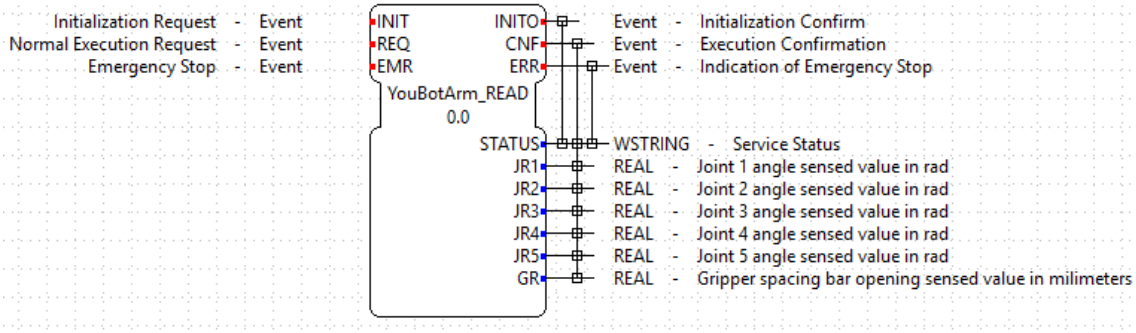

Figura 7-3: FB de lectura encargado de la manipulación del brazo KUKA youBot

Fuente: Mafla, Gabriela, 2019

El bloque de lectura permite monitorear el desplazamiento angular de cada una de las articulaciones, esto se logra mediante la información enviada por los sensores del brazo, y a su vez esto permite realizar el control de posicionamiento de cada una de ellas, permitiendo un control en lazo cerrado.

En cuanto al control de ejecución de este FB, tiene como entradas un evento INIT para inicializar el bloque, un evento REQ para actualizar los parámetros y un evento EMR para detener cualquier operación que se esté generando. Como eventos de salida este FB cuenta con la confirmación de la inicialización INITO y un evento CNF para indicar el fin de la etapa de ejecución. Como parámetros de salida tiene datos de tipo real denominados JR1 – JR5 y un GR, devuelven los valores de ángulo de las uniones y el valor de apertura de la pinza respectivamente, en el momento de la operación de lectura.

3.5.4 FB CONTROLLER_PtP

Subsidiariamente, hay que añadir un FB que aunque no forme parte directamente del control, asegura un correcto funcionamiento de la aplicación. Este FB se centra en la ejecución de un esquema de control basado en el modelo cinemático youBot.

El FB denominado "*CONTROLLER_PtP*" representado en la Figura 8-3, incluye un algoritmo de control basado en el modelo cinemático del brazo robótico, está diseñado para leer un archivo XML con el fin de obtener los puntos de referencia para la ejecución de sus operaciones (Garcia et al., 2019).

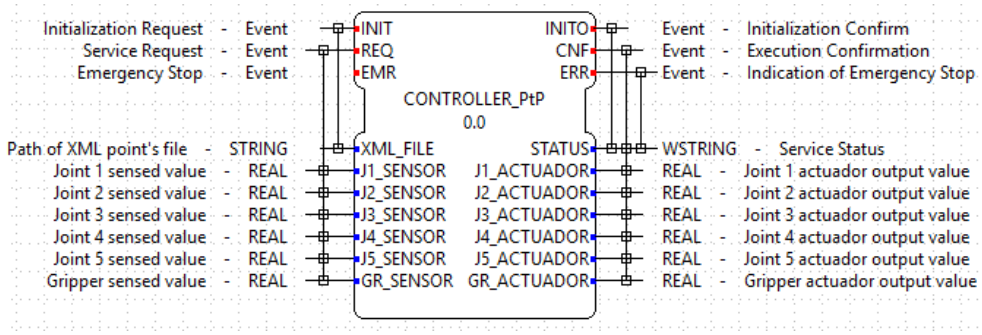

Figura 8-3: FB auxiliar de control

Fuente: Mafla, Gabriela, 2019

Siguiendo la estructura de ejecución de los FBs anteriores, las particularidades del bloque desarrollado se describen en la Tabla 3-3.

Tabla 3-3: Características del FB CONTROLLER_PtP

EVENTOS DE ENTRADA	**EVENTOS DE SALIDA**
INIT: evento de entrada que sirve para inicializar el bloque.	**INITO:** evento de salida que sirve para la confirmación de la inicialización.
REQ: evento desencadenante de la operación correspondiente a un ciclo normal de trabajo.	**CNF:** evento de salida que sirve para indicar el fin de la etapa de ejecución.
EMR: evento desencadenante de la finalización de cualquier operación que se está generando.	**ERR:** evento de salida activado con el paro de emergencia.
DATOS DE ENTRADA	**DATOS DE SALIDA**
XML_FILE (STRING): ruta del archivo XML que contiene los puntos de referencia de control.	**STATUS (WSTRING):** devuelve como valor de cadena el texto correspondiente al estado o FB, siendo las opciones: inicializado, operativo y parado.
J1_SENSOR – J5_SENSOR (REAL): valores de ángulo de las uniones en el momento de la operación de lectura formateada como datos de tipo real.	**J1_ACTUADOR – J5_ACTUADOR (REAL):** resultados de las operaciones del controlador que representan los ángulos de unión deseados del manipulador.
GR_SENSOR (REAL): valor de apertura de la pinza en el momento de la operación de lectura formateada como datos de tipo real.	**GR_ACTUADOR (REAL):** resultado de la operación del controlador que representa la apertura de la pinza deseada del manipulador.

Fuente: (Garcia et al., 2019)

Realizado por: Mafla, Gabriela, 2019

3.6 Incorporación de FORTE en dispositivo de control

Como plataforma de ejecución para la implementación del sistema se ha elegido FORTE debido a la versatilidad que ofrece para elegir el hardware, ya que es independiente de la plataforma en la que se ejecute y funciona sobre varios sistemas operativos como Windows o Linux.

FORTE básicamente es una 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 (Titulación et al., 2017), como se muestra en la Figura 9-3.

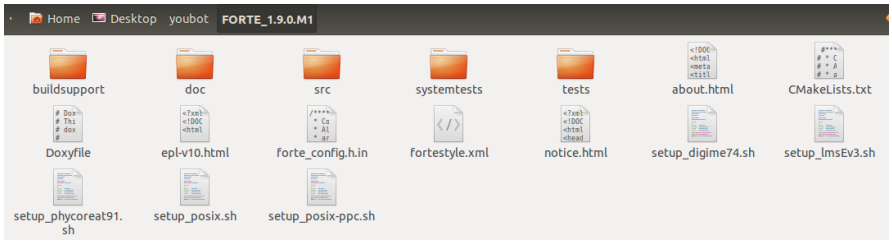

Figura 9-3: Archivos base de FORTE ubicados en escritorio del Kuka YouBot

Fuente: Mafla, Gabriela, 2019

3.6.1 Compilación y generación de archivo ejecutable FORTE

Para poder usar los FBs desarrollados y ejecutarlos en FORTE es preciso crear un nuevo módulo en los archivos base de FORTE, para lo cual se debe ingresar en la carpeta contenedora de FORTE y ubicarse en la dirección *</src/modules>*, seguidamente crear una carpeta en donde se adicionarán sus archivos exportados, en el caso de este proyecto la carpeta se llama *<FncYouBot>* como se muestra en la Figura 10-3.

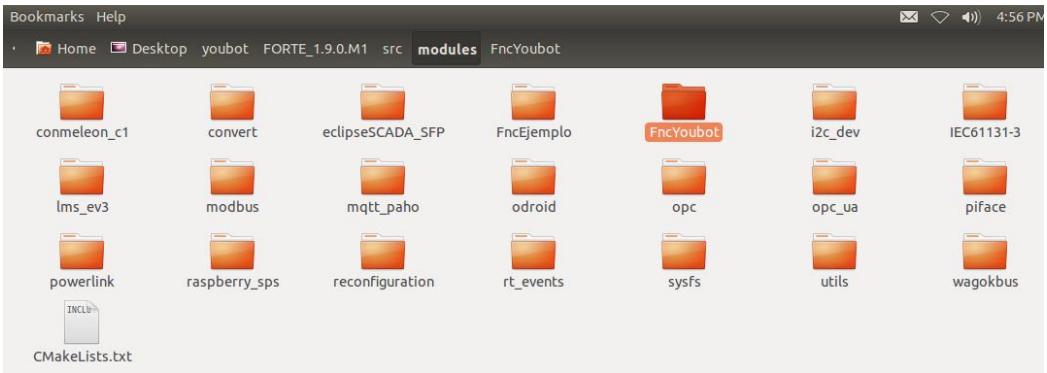

Figura 10-3: Carpeta FncYouBot creada en la localización /src/modules para contener los archivos del nuevo módulo de funciones en desarrollo.

Fuente: Mafla, Gabriela, 2019

En la carpeta de módulo generada se debe añadir los archivos exportados y un archivo de texto con el nombre <*CMakeLists*> como se muestra en la Figura 11-3.

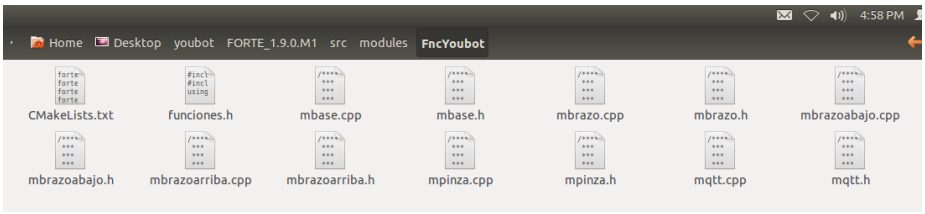

Figura 11-3: Carpeta FncYouBot contenedora de funciones y archivo CMakeLists.txt generados dentro de carpeta de módulo.

Fuente: Mafla, Gabriela, 2019

El archivo <*CMakeLists*> contiene la descripción del módulo desarrollado y todos los recursos de las funciones de los movimientos del robot como se muestra en la Figura 12-3.

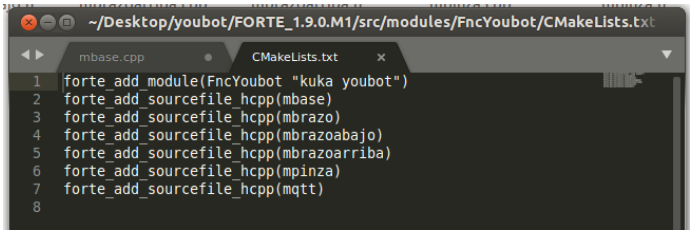

Figura 12-3: Contenido del archivo CMakelists.txt

Fuente: Mafla, Gabriela, 2019

Adquiridos todos los archivos de FORTE y sus módulos se procede a compilarlos con el utilitario <*cmake-gui*>, el cual debe tener acceso en modo de escritura y lectura, ya que se va a generar nuevos archivos de compilación a partir del código fuente de FORTE.

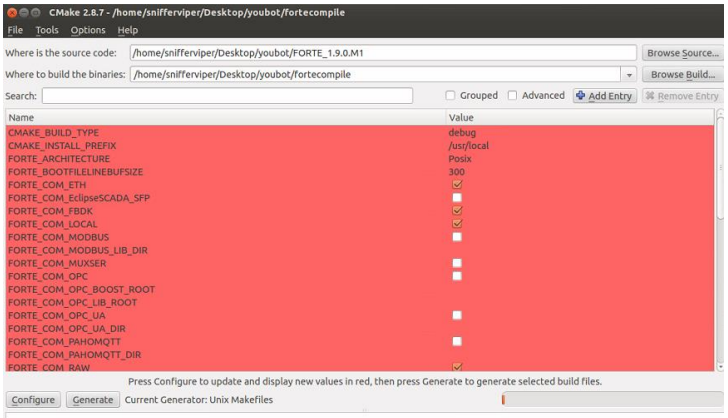

Figura 13-3: Ventana de selección de generador

Fuente: Mafla, Gabriela, 2019

Una vez abierto CMake se visualiza varios parámetros de configuración, entre los cuales se encuentra la dirección del código fuente, la dirección donde se guarda los archivos y los parámetros configurados para la compilación. Debemos poner especial énfasis en el tipo de compilación que será el valor de debug, también la arquitectura deberá ser POSIX que es la recomendada para trabajar con FORTE en el control de manipuladores móviles como se muestra en la Figura 13-3. Con las modificaciones hechas, al presionar el botón <*Generate*> el proceso de generación se completará exitosamente y dentro de la carpeta destinada a contener los binarios de compilación se mostrarán dichos archivos.

Como último paso para generar el archivo ejecutable FORTE como se muestra en la Figura 14-3 es necesario abrir un terminal con ubicación en la carpeta contenedora de binarios de compilación, esta debe tener permisos de superusuario, seguidamente se debe ejecutar el comando <*make*>.

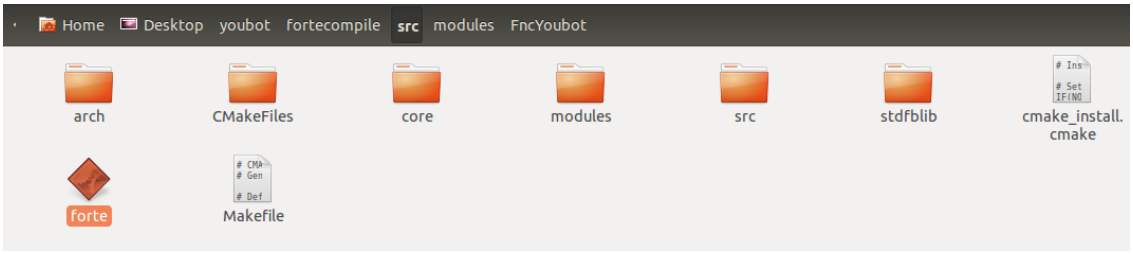

Figura 14-3: Carpeta con los archivos FORTE generados con CMake

Fuente: Mafla, Gabriela, 2019

3.6.2 Prueba de FBs

Una vez que se ha compilado correctamente la aplicación de FORTE en el dispositivo de control, se procede a probar los bloques de funciones (FBs) de los módulos utilizados en la compilación como se detalla a continuación:

1. Abrir una terminal remota hacia el robot con privilegios de superusuario y localizar la carpeta donde se ubica el archivo compilado de FORTE.

2. En la subcarpeta <*src*> mediante la línea de código <*./forte*> ejecutar la aplicación <*forte*>, como se muestra en la Figura 15-3.

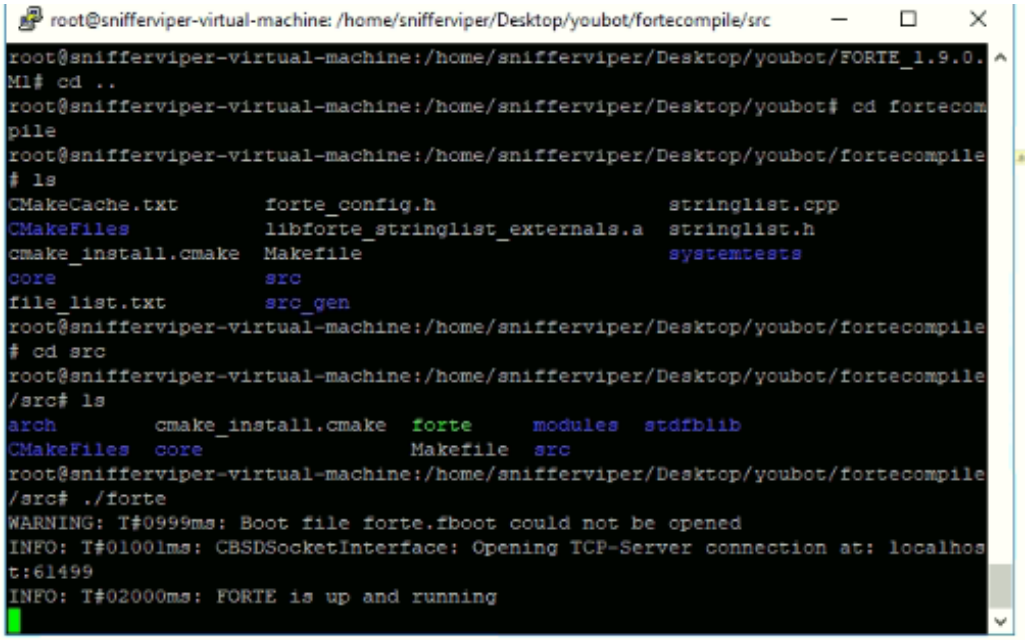

Figura 15-3: Terminal para ejecución de aplicación *forte*

Fuente: Mafla, Gabriela, 2019

3. En el entorno de desarrollo 4DIAC-IDE abrir el bloque funcional (FB) previamente creado. En el área de trabajo ubicarse en la opción <*FBTester*> donde se define la configuración de prueba y se ingresa la dirección IP del robot, finalmente pulsar el botón <*Start Testing FB*> para empezar la prueba del bloque funcional como se muestra en la Figura 16-3.

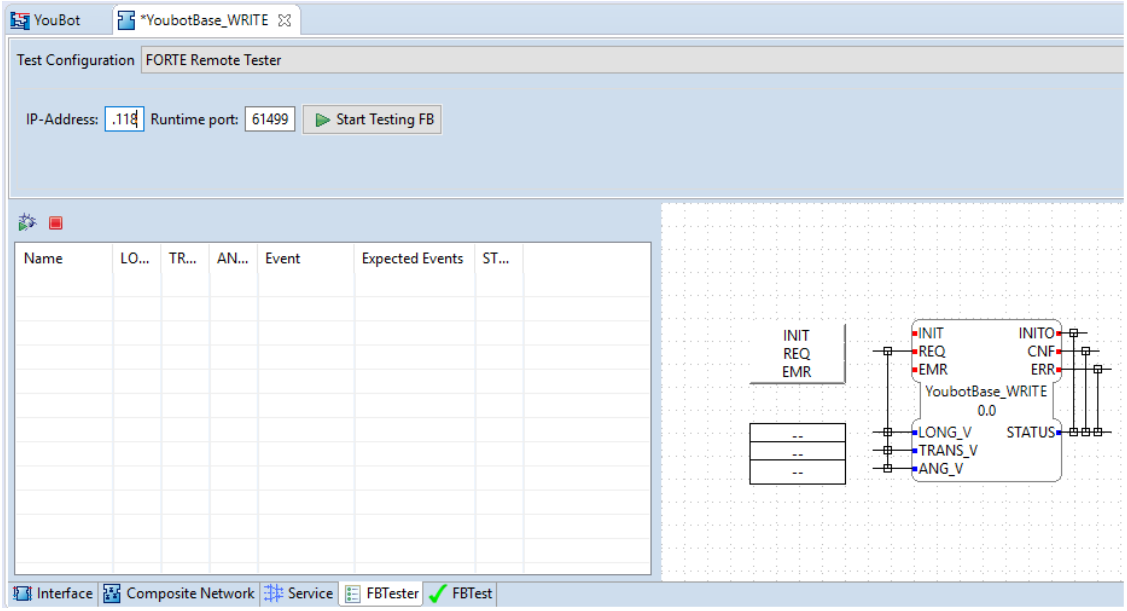

Figura 16-3: Interfaz de prueba de FBs de 4DIAC-IDE

Fuente: Mafla, Gabriela, 2019

3.7 Desarrollo del Sistema Distribuido

3.7.1 Diseño de la aplicación de control

El primer paso en la implementación de un sistema de control distribuido según el estándar IEC-61499 es modelar el sistema con la herramienta 4DIAC-IDE, determinando los dispositivos implicados y los canales de comunicación utilizados entre ellos, seguidamente se realiza la integración de FBs bajo el entorno de ejecución FORTE.

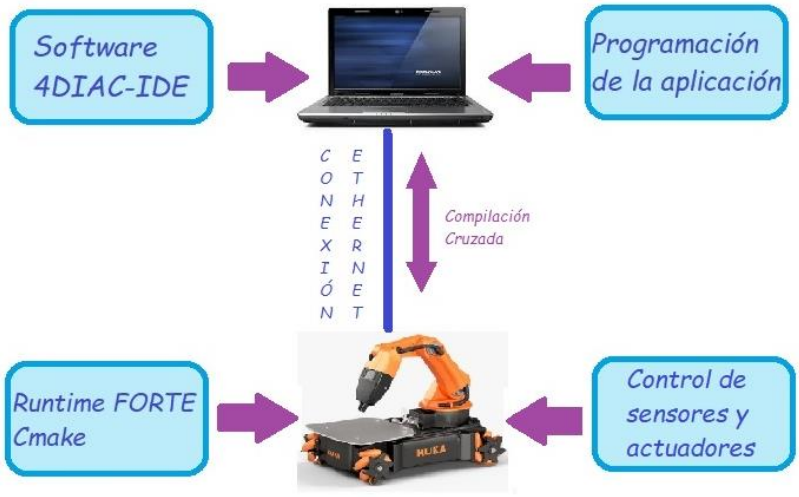

Figura 17-3: Diseño de la aplicación de control

Fuente: Mafla, Gabriela, 2019

La aplicación de control final que se muestra en la Figura 17-3, permite la manipulación de los controladores de youBot, para lo cual se incluye un algoritmo de control basado en el modelo cinemático del brazo robótico.

Adicional a las librerías existentes en 4DIAC-IDE, este entorno de desarrollo permite la creación y vinculación de nuevas librerías con las que se puede implementar prácticamente cualquier aplicación de control. Entre las usadas están todas las librerías del ejemplo “HOLA MUNDO” del software ROS añadidas una vez exportadas a código C++. Para acceder a los diferentes elementos, se utiliza las siguientes clases (C++):

- YouBotBase.hpp
- YouBotBase.cpp
- YouBotManipulator.hpp

3.7.2 Implementación del sistema

La distribución final del SCD desarrollado se muestra en la Figura 18-3, el cual se basa en el uso de una red Ethernet para interconectar, se añade un FORTE_PC y se configura la dirección IP 192.168.10.118 que tiene la PC a bordo del robot KUKA youBot.

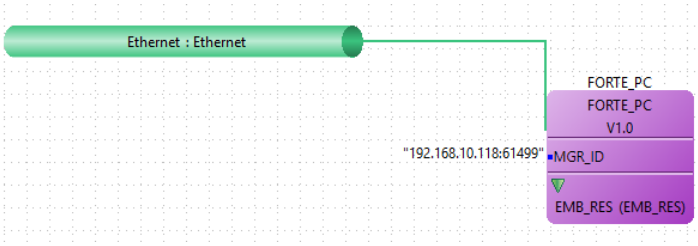

Figura 18-3: Sistema de control distribuido final desarrollado en 4DIAC-IDE, destinado al control del Robot Kuka youBot.

Fuente: Mafla, Gabriela, 2019

Como se describió anteriormente, el FB es la unidad fundamental del estándar IEC-61499, utilizando un conjunto de FB interconectados se construye el sistema de control distribuido, como se muestra en la Figura 19-3. La aplicación de control final se genera con la integración de dos grupos de FB; el primer conjunto permite que los datos se escriban y lean en los recursos físicos del manipulador móvil. El segundo conjunto incluye un algoritmo de control basado en el modelo cinemático del KUKA youBot (Garcia, Naranjo, Zambrano, Ambato, & Lanas, 2018). Las líneas rojas simbolizan los eventos con los que se gestiona la ejecución de los diferentes bloques de función. Con las entradas/salidas de evento INIT/INITO se establece una cadena con la que se da la orden de inicializar los bloques de función que lo requieren. Cuando un bloque termina su inicialización se activa el evento INITO inicializándose así el siguiente. El flujo de datos está representado con líneas azules.

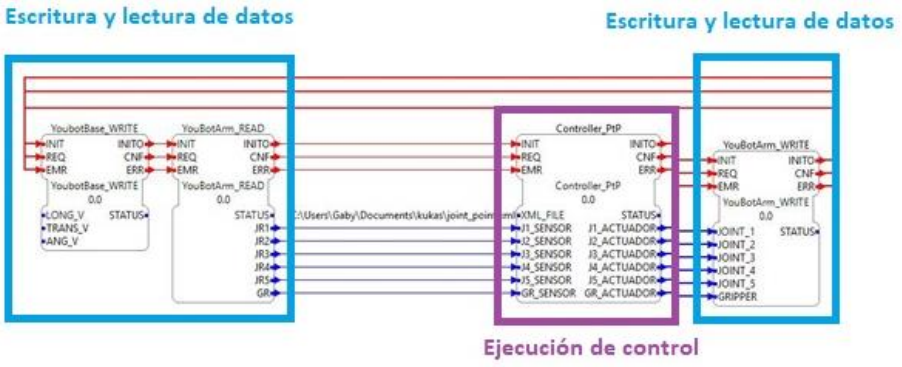

Figura 19-3: Aplicación de Control Distribuido en 4DIAC-IDE

Fuente: Mafla, Gabriela, 2019

Finalmente, el sistema de control distribuido se genera utilizando la misma herramienta de ingeniería para relacionar las aplicaciones creadas con los dispositivos de una red de control especificada y luego descargarlos en ellos. Para que un controlador incorporado pueda ejecutar las operaciones del sistema, se necesita la instalación de un entorno de ejecución como FORTE en su sistema operativo, que puede ejecutar los algoritmos especificados en los FB del sistema según las directrices IEC-61499 (Garcia et al., 2018).

CAPÍTULO IV

4 RESULTADOS Y DISCUSIÓN

4.1 Pruebas de funcionamiento del sistema

Después de implementar la aplicación de control, es necesario realizar varias pruebas de funcionamiento sobre cada FB. Para la ejecución de estas pruebas de carácter técnico es preciso evaluar cada una de ellas con valores booleano (0 o 1), que permiten establecer el funcionamiento de acuerdo a los siguientes parámetros:

- ***Válido (1):*** la prueba se ha realizado con éxito; es decir el movimiento se ejecutó.
- ***Inválido (0):*** no existe funcionamiento en la prueba; es decir el movimiento no se ejecutó.

El tamaño de la muestra en estudios de carácter experimental es aleatoria, esto significa que el número de repeticiones a realizar en cada tratamiento depende de la variabilidad que se espera observar en los datos, a la diferencia mínima que el experimentador considera que es importante detectar y al nivel de confianza que se desea tener en las conclusiones. Normalmente se recomiendan entre 10 y 30 mediciones en cada tratamiento. Para este trabajo de investigación se determina un número de 20 repeticiones por cada prueba, que abarca el fallo general y nos da una prueba confiable.

4.1.1 Prueba de ejecución de movimientos en V-REP

V-REP permite hacer pruebas iniciales del algoritmo implementado, sin correr riesgos debidos a la realización de pruebas con el sistema real. Para este sistema será transparente el que estemos trabajando con un robot real o con un simulador.

Para este caso se emplea el uso de 4DIAC para desarrollar la programación y como se ha mencionado para llevar a cabo las simulaciones se emplea V-REP. Estos dos programas se podrán comunicar entre sí, gracias a las posibilidades que ofrece V-REP de actuar como servidor y mantenerse a la espera de recibir comandos de un cliente, que en este caso será 4DIAC.

Para realizar la simulación se debe trabajar bajo Linux, instalar V-REP y ROS con todos los drivers necesarios para la comunicación externa. Una vez que se tiene instalado lo mencionado anteriormente se procede a virtualizar ROS, con lo cual se logra establecer la comunicación entre ellos. Al actuar o manejarse mediante forma virtualizada, ROS va actuar como un puente de comunicación entre 4DIAC y V-REP. La configuración de la comunicación entre estos dos programas se detalla en el Anexo C.

La finalidad de las pruebas realizadas en el entorno de simulación es determinar si la aplicación de control desarrollada en 4DIAC, es capaz de ejecutar sin errores cada uno de los movimientos determinados para llevar a cabo la manipulación del robot KUKA youBot, con lo cual se asegura que el funcionamiento del SCD al ser ejecutado en el dispositivo real no tendrá problemas y no ocasionará daños en el mismo.

Entre los movimientos que se consideran para la manipulación del youBot están: movimiento longitudinal, movimiento transversal, movimiento angular, movimiento del brazo y el gripper. Para realizar las pruebas se testea cada uno de los FBs desarrollados en el apartado anterior, se define valores de velocidad para obtener movimientos de la plataforma y valores de ángulos para el movimiento del brazo robótico, seguidamente en V-REP se puede apreciar la ejecución de los movimientos, como se observa en las siguientes Figuras.

Movimiento longitudinal: como se observa en la Figura 1-4, el movimiento longitudinal realizado sobre el eje x, se ha ejecutado satisfactoriamente en el entorno virtual.

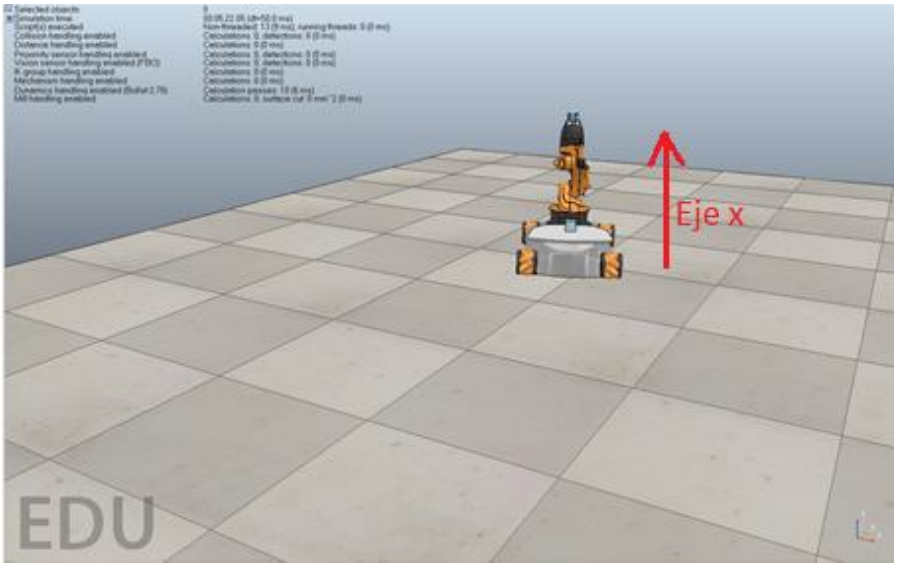

Figura 1-4: Prueba de movimiento longitudinal robot KUKA youBot en V-REP

Fuente: Entorno de simulación V-REP, 2019

Movimiento angular: al igual que en el caso anterior, el movimiento angular realizado sobre el eje z, como se observa en la Figura 2-4, se ha ejecutado satisfactoriamente en el entorno virtual.

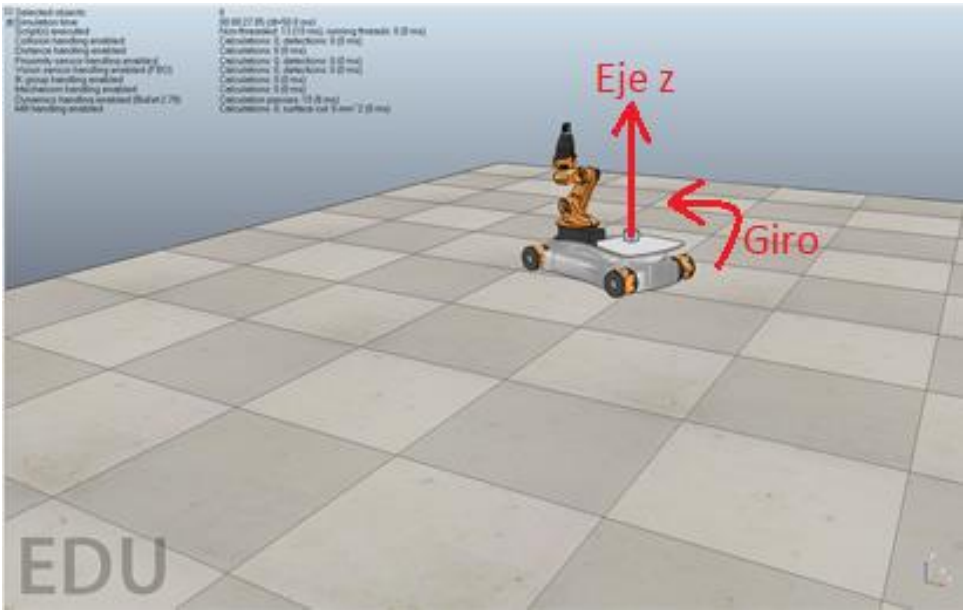

Figura 2-4: Prueba de movimiento angular robot KUKA youBot en V-REP

Fuente: Entorno de simulación V-REP, 2019

Movimiento brazo y gripper: al definir valores de los ángulos en el FB *YouBotArm_WRITE*, se puede evidenciar en la Figura 3-4 que el brazo robótico pasa de la posición home a tomar la posición en base a los valores definidos.

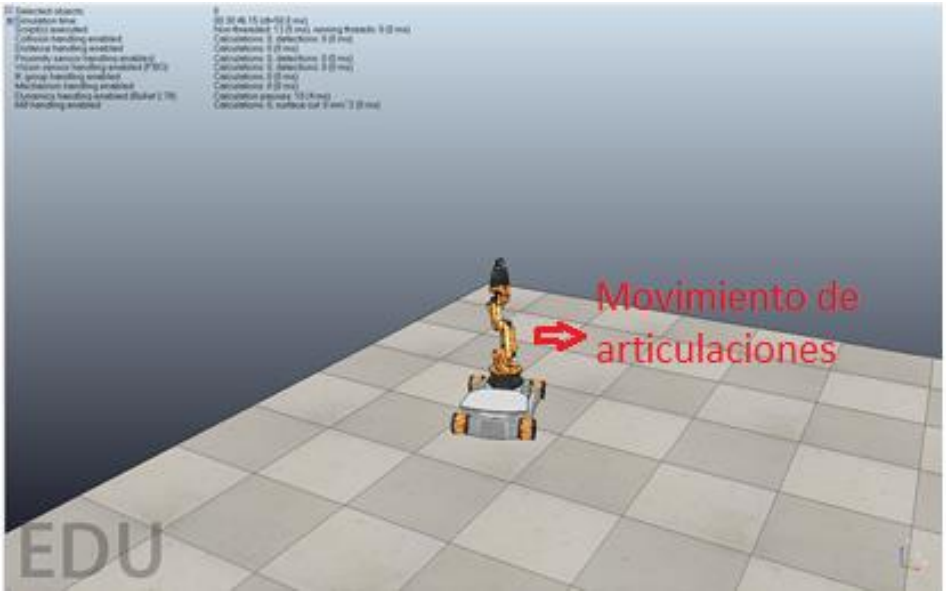

Figura 3-4: Prueba de movimiento brazo robótico robot KUKA youBot en V-REP

Fuente: Entorno de simulación V-REP, 2019

Con los resultados obtenidos en las pruebas anteriores, la simulación de la aplicación de control ayuda a entender mejor la operación de la misma, a detectar las variables más importantes que interactúan en el sistema y a entender mejor las interrelaciones entre estas variables. De esta manera se puede determinar que el sistema de control distribuido implementado en 4DIAC funciona correctamente, con lo cual se procede a realizar las respectivas pruebas en el dispositivo real.

4.1.2 Prueba de ejecución de movimientos de la plataforma

Para las pruebas se procedió a tomar parámetros característicos del desplazamiento de la plataforma. Se cuenta con tres tipos de movimientos: longitudinales, trasversales y angulares, propios de las llantas omnidireccionales que tiene el robot, cada una de ellas posee un motor que las controla.

El objetivo de esta prueba es determinar si la aplicación de control es capaz de ejecutar sin errores cada uno de los movimientos previamente establecidos y probados en V-REP, para la manipulación real de la plataforma youBot.

Movimiento longitudinal: al testear el FB denominado *YouBotBase_WRITE*, se ingresa manualmente valores correspondientes a la velocidad longitudinal, con lo cual se logra que la plataforma se mueva hacia adelante o atrás dependiendo del valor ingresado, tal como se muestra en la Figura 4-4, como se determinó anteriormente esta prueba se repite 20 veces.

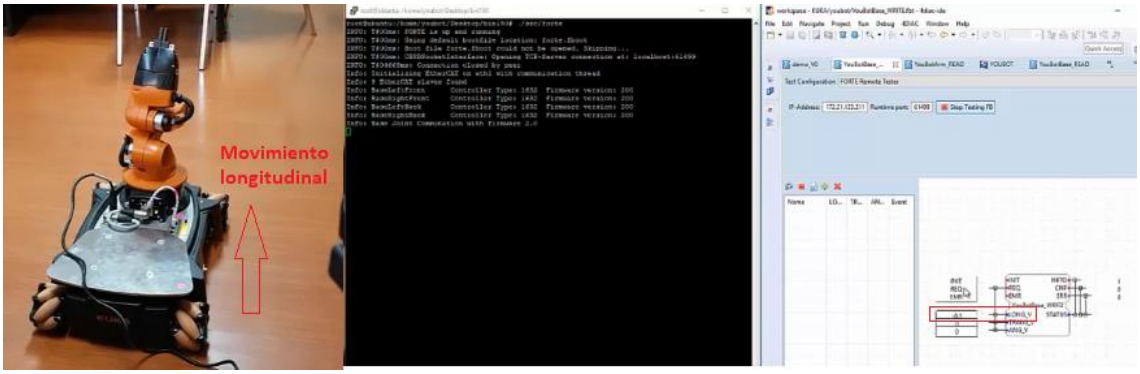

Figura 4-4: Prueba de movimiento longitudinal en robot KUKA youBot

Fuente: Mafla, Gabriela, 2019

Como se muestra en la Tabla 1-4, la ejecución del movimiento longitudinal tiene un porcentaje muy alto, que le da un gran nivel de confiabilidad.

Tabla 1-4: Valores observados en la prueba de ejecución de movimiento longitudinal

Prueba de ejecución movimiento longitudinal				
VALOR	**VÁLIDO**	**INVÁLIDO**	**TOTAL**	**%**
SI	19	1	20	100%
NO	0	0	0	0%
TOTAL	19	1	20	100%
%	95%	5%	100%	

Fuente: Realizado por: Mafla, Gabriela, 2019

Movimiento transversal: muy similar al caso anterior, al testear el FB denominado *YouBotBase_WRITE*, se ingresa manualmente valores correspondientes a la velocidad transversal como se representa en la Figura 5-4, dependiendo del valor ingresado la plataforma se mueva hacia la izquierda o derecha respectivamente. Como se determinó anteriormente esta prueba se repite 20 veces.

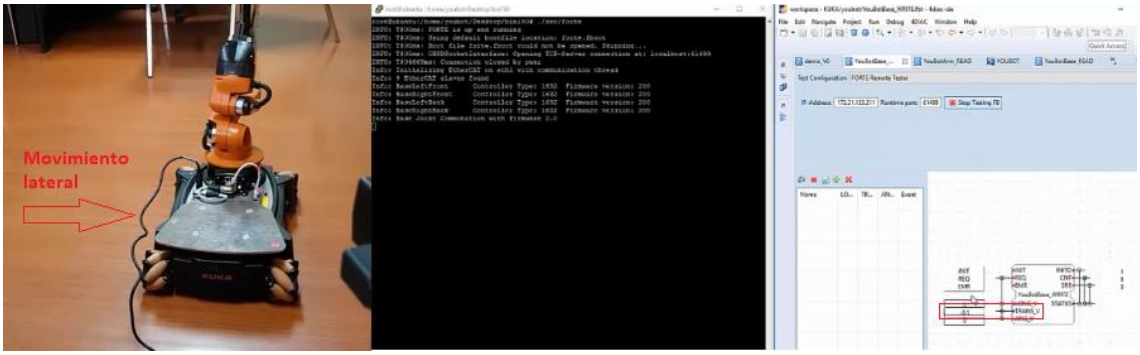

Figura 5-4: Prueba de movimiento transversal en robot KUKA youBot

Fuente: Mafla, Gabriela, 2019

La ejecución del movimiento transversal como se muestra en la Tabla 2-4, tiene un porcentaje muy alto, que le da un gran nivel de confiabilidad.

Tabla 2-4: Valores observados en la prueba de ejecución de movimiento transversal

Prueba de ejecución movimiento transversal				
VALOR	**VÁLIDO**	**INVÁLIDO**	**TOTAL**	**%**
SI	20	0	20	100%
NO	0	0	0	0%
TOTAL	20	0	20	100%
%	100%	0%	100%	

Fuente: Realizado por: Mafla, Gabriela, 2019

Movimiento angular: al testear el FB denominado *YouBotBase_WRITE*, se ingresa manualmente valores correspondientes a la velocidad angular, con lo cual se logra que la plataforma gire alrededor de su eje central, tal como se muestra en la Figura 6-4, como se determinó anteriormente esta prueba se repite 20 veces.

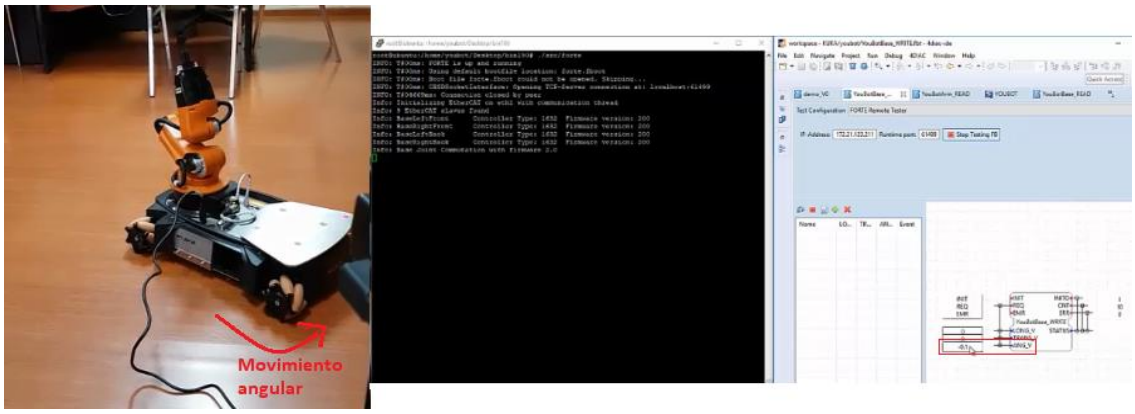

Figura 6-4: Prueba de movimiento angular en robot KUKA youBot

Fuente: Mafla, Gabriela, 2019

Como se muestra en la Tabla 3-4, la ejecución del movimiento angular tiene un porcentaje muy alto, que le da un gran nivel de confiabilidad.

Tabla 3-4: Valores observados en la prueba de ejecución de movimiento angular

Prueba de ejecución movimiento angular				
VALOR	**VÁLIDO**	**INVÁLIDO**	**TOTAL**	**%**
SI	20	0	20	100%
NO	0	0	0	0%
TOTAL	20	0	20	100%
%	100%	0%	100%	

Fuente: Realizado por: Mafla, Gabriela, 2019

4.1.3 Prueba de ejecución de movimientos del brazo y la pinza

Se realizó pruebas del algoritmo de cinemática directa implementado en la aplicación de control. El objetivo de esta prueba es determinar si el SCD controla el movimiento de cada eslabón y la pinza o gripper, que forman parte del brazo robótico.

Al testear el FB *YouBotArm_WRITE* como se muestra en la Figura 7-4, se colocan los valores de los ángulos de cada una de las articulaciones para lograr el movimiento deseado, en el caso del

gripper se define el valor de apertura deseado. Este procedimiento se repite 20 veces para cada parámetro definido.

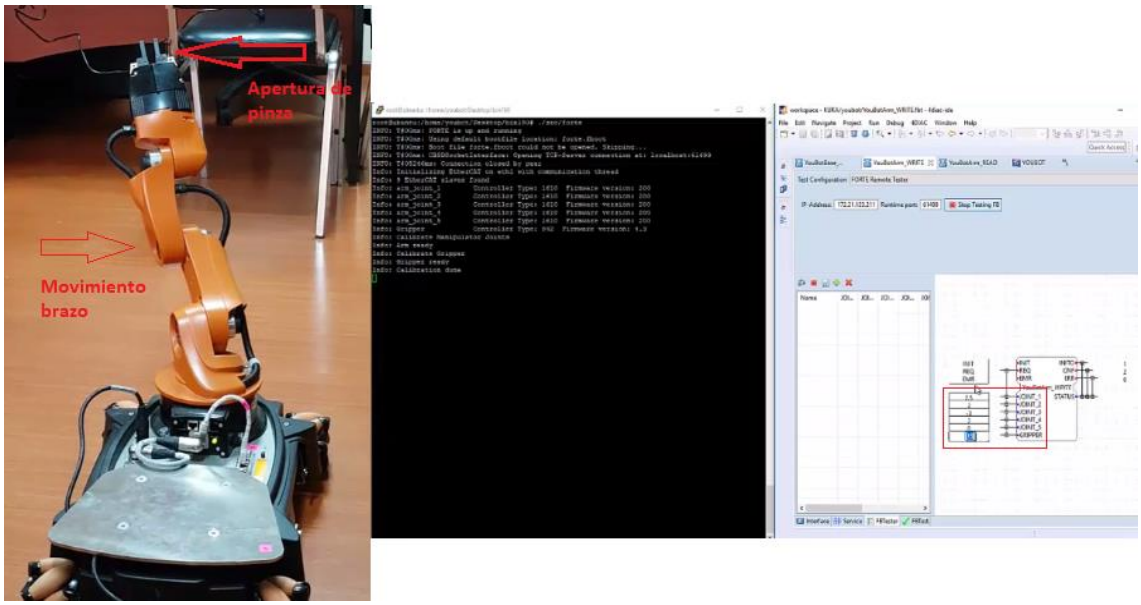

Figura 7-4: Prueba de movimiento angular en robot KUKA youBot

Fuente: Mafla, Gabriela, 2019

Como se muestra en la Tabla 4-4, la ejecución del movimiento del brazo robótico KUKA youBot tiene un porcentaje muy alto, que le da un gran nivel de confiabilidad.

Tabla 4-4: Valores observados en la prueba de ejecución de movimiento del brazo robótico

Prueba de ejecución movimiento brazo robótico					
JOINT	**VALOR**	**VÁLIDO**	**INVÁLIDO**	**TOTAL**	**%**
1	**SI**	19	1	20	100%
	NO	0	0	0	0%
	TOTAL	19	1	20	100%
	%	95%	5%	100%	
2	**SI**	20	0	20	100%
	NO	0	0	0	0%
	TOTAL	20	0	20	100%
	%	100%	0%	100%	
3	**SI**	20	0	20	100%
	NO	0	0	0	0%
	TOTAL	20	0	20	100%

	%	100%	0%	100%	
4	**SI**	20	0	20	100%
	NO	0	0	0	0%
	TOTAL	20	0	20	100%
	%	100%	0%	100%	
5	**SI**	20	0	20	100%
	NO	0	0	0	0%
	TOTAL	20	0	20	100%
	%	100%	0%	100%	
Gripper	**SI**	20	0	20	100%
	NO	0	0	0	0%
	TOTAL	20	0	20	100%
	%	100%	0%	100%	

Fuente: Realizado por: Mafla, Gabriela, 2019

4.2 Pruebas comparativas

Para poder tener un punto de contraste entre el sistema de control original desarrollado bajo ROS 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.

4.2.1 Tiempo de culminación de un ciclo de trabajo de los sistemas

La finalidad de esta prueba consistía en cronometrar los tiempos de 15 ciclos de trabajo del sistema basado en ROS y del sistema basado en IEC-61499, en donde el robot realiza movimientos libres, estos datos fueron obtenidos del entorno de simulación V-REP; los resultados se presentan en la Tabla 5-4:

Tabla 5-4: Tiempos de iteraciones medidos en los sistemas basados en ROS e IEC-61499

		SISTEMA ROS		**SISTEMA IEC-61499**	
No. ITERACIÓN	**MOVIMIENTO**	**ESTADO**	**TIEMPO DE EJECUCIÓN (ms)**	**ESTADO**	**TIEMPO DE EJECUCIÓN (ms)**
1		Ejecutado	300,0	Ejecutado	300,2
2		Ejecutado	300,2	Ejecutado	300,1

3	**PLATAFORMA**	Ejecutado	300,3	Ejecutado	300,3
4		Ejecutado	300,7	Ejecutado	300,5
5		Ejecutado	300,1	Ejecutado	300,0
6	**BRAZO**	Ejecutado	200,0	Ejecutado	200,0
7		Ejecutado	200,2	Ejecutado	200,4
8		Ejecutado	200,6	Ejecutado	200,8
9		Ejecutado	200,1	Ejecutado	200,3
10		Ejecutado	200,8	Ejecutado	200,6
11	**PINZA**	Ejecutado	98,1	Ejecutado	99,5
12		Ejecutado	99,4	Ejecutado	98,4
13		Ejecutado	100,5	Ejecutado	100,2
14		Ejecutado	99,5	Ejecutado	99,5
15		Ejecutado	97,5	Ejecutado	95,5

Fuente: Entorno de simulación V-REP, 2019

Realizado por: Mafla, Gabriela, 2019

Con los datos recopilados se realizaron los cálculos necesarios para obtener los datos presentados en la Tabla 6-4:

Tabla 6-4: Tiempos de iteraciones medidos en los sistemas basados en ROS e IEC-61499

	SISTEMA ROS			**SISTEMA IEC-61499**		
MOVIMIENTO	**TIEMPO MÁXIMO (ms)**	**TIEMPO MÍNIMO (ms)**	**TIEMPO PROMEDIO (ms)**	**TIEMPO MÁXIMO (ms)**	**TIEMPO MÍNIMO (ms)**	**TIEMPO PROMEDIO (ms)**
PLATAFORMA	300,7	300,0	300,26	300,5	300,0	300,22
BRAZO	200,8	200,0	200,34	200,8	200,0	200,42
PINZA	100,5	97,5	99,0	100,2	98,4	98,62

Fuente: Realizado por: Mafla, Gabriela, 2019

4.3 Análisis de resultados

4.3.1 Validación de la hipótesis

Se desea conocer si el sistema distribuido implementado realiza el control del robot KUKA youBot a través de los diferentes movimientos establecidos, tanto para la plataforma como para el brazo. La validez de la aplicación, está definida por el cumplimiento de los objetivos en cada una de las pruebas.

Con la finalidad de comprobar la hipótesis se utiliza el método estadístico de Análisis de la Varianza (ANOVA) de un solo factor, para lo cual es necesario plantear:

Hipótesis Nula (H_0): El desarrollo de un sistema distribuido bajo la norma IEC-61499, no permite manipular el robot Kuka youBot cumpliendo con el concepto de control distribuido.

Hipótesis Alternativa (H_1): El desarrollo de un sistema distribuido bajo la norma IEC-61499, permite manipular el robot Kuka youBot cumpliendo con el concepto de control distribuido.

Mediante el programa SPSS se analiza los datos obtenidos para poder aceptar o rechazar la hipótesis planteada.

Los movimientos definidos para el desarrollo de la presente investigación, se resumen en la Tabla 7-4 que contiene el cuadro estadístico descriptivo.

Tabla 7-4: Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desviación estándar
MOV_LONG	20	0	1	,95	,224
MOV_TRANS	20	1	1	1,00	,000
MOV_ANG	20	1	1	1,00	,000
JOINT_1	20	0	1	,95	,224
JOINT_2	20	1	1	1,00	,000
JOINT_3	20	1	1	1,00	,000
JOINT_4	20	1	1	1,00	,000
JOINT_5	20	1	1	1,00	,000
GRIPPER	20	1	1	1,00	,000
N válido (por lista)	20				

Fuente: SPSS, 2019

Realizado por: Mafla, Gabriela, 2019

Para calcular las medias respectivas a cada movimiento se realiza un análisis descriptivo, el cual permite conocer el número de observaciones, media, desviación estándar, error estándar, intervalo de confianza para la media y valores mínimo y máximo como se muestra en la Tabla 8-4, y obtener un promedio general del funcionamiento del sistema distribuido, medido por el número de aciertos en cada prueba.

Tabla 8-4: Análisis descriptivo

Descriptivos								
PROMEDIO_ACIERTOS								
	N	**Media**	**Desviación estándar**	**Error estándar**	**95% del intervalo de confianza para la media**		**Mínimo**	**Máximo**
					Límite inferior	**Límite superior**		
1	1	95,00	.	.	.	.	95	95
2	1	100,00	.	.	.	.	100	100
3	1	100,00	.	.	.	.	100	100
4	1	95,00	.	.	.	.	95	95
5	1	100,00	.	.	.	.	100	100
6	1	100,00	.	.	.	.	100	100
7	1	100,00	.	.	.	.	100	100
8	1	100,00	.	.	.	.	100	100
9	1	100,00	.	.	.	.	100	100
Total	9	98,89	2,205	,735	97,19	100,58	95	100

Fuente: SPSS, 2019

Realizado por: Mafla, Gabriela, 2019

Para los sujetos de prueba en las secuencias:

1. MOV_LONG
2. MOV_TRANS
3. MOV_ANG
4. JOINT_1
5. JOINT_2
6. JOINT_3
7. JOINT_4
8. JOINT_5
9. GRIPPER

Luego de calcular las medias respectivas para cada movimiento, se realiza el análisis de la varianza como se muestra en la Tabla 9-4. Esta tabla nos permite aceptar o rechazar la hipótesis nula o la hipótesis alternativa.

Tabla 9-4: Análisis de la varianza (ANOVA)

ANOVA					
PROMEDIO_ACIERTOS					
	Suma de cuadrados	**gl**	**Media cuadrática**	**F**	**Sig.**
Entre grupos	38,889	8	4,861	.	0,0002.
Dentro de grupos	,000	0	.		
Total	38,889	8			

Fuente: SPSS, 2019

Realizado por: Mafla, Gabriela, 2019

Se realiza la decisión estadística mediante la comprobación del valor de la columna de significancia.

Se establece el nivel de significancia α=0,05.

Donde:

Si el valor Sig. > 0,05 Se acepta la H_0

Si el valor Sig. < 0,05 Se acepta la H_1

Entonces:

Como 0,0002<0,05 se cumple la hipótesis específica H_1

H_1= El desarrollo de un sistema distribuido bajo la norma IEC-61499, permite manipular el robot Kuka youBot cumpliendo con el concepto de control distribuido.

Con esta comprobación, se puede concluir que el sistema de control distribuido implementado bajo la norma IEC-61499 logra cumplir con su acometido, controlar el robot KUKA youBot a través de la manipulación de la plataforma y el brazo robótico, cumpliendo con el concepto de control distribuido.

4.3.2 Verificación de características distintivas de la norma IEC-61499

Luego de realizar la implementación del sistema se verificó los objetivos que rigen a la norma IEC-61499:

- ***Portabilidad:*** mediante el software 4DIAC-IDE se logró observar el estado de las variables del proceso en tiempo real y la ejecución de los eventos en el bloque de función.
- ***Configurabilidad:*** el empleo de bloques funcionales permitió tener un sistema reconfigurable debido a que se puede cambiar ciertos parámetros en el funcionamiento del proceso.
- ***Interoperabilidad:*** se puede crear sistemas flexibles, debido a que se puede tener componentes con distintos sistemas operativos en un mismo proceso. Además se tiene un solo lenguaje de programación para los dispositivos que soporten el estándar.

4.3.3 Comparación Sistema de Control ROS vs Sistema de Control IEC-61499

Al aplicar la metodología desarrollada en el punto 3.4 para la implementación de aplicaciones de control distribuida en las herramientas software, se llega a la conclusión de que este tipo de herramientas no son utilizadas en la implementación de control cinemático para la manipulación de robots móviles, debido al desconocimiento y gran parte del software aún se encuentra en desarrollo.

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 robot KUKA youBot. Al comparar la media de ciclo del sistema ROS de 300,26 milisegundos (plataforma), 200,34 milisegundos (brazo) y 99,0 milisegundos (pinza), con la media de ciclo del sistema IEC-61499 de 300,22 milisegundos (plataforma), 200,42 milisegundos (brazo) y 98,62 milisegundos (pinza), se puede observar que más allá de obtener una réplica de funcionamiento por parte del sistema distribuido, se obtiene un comportamiento muy similar, ya que no existe retardos en los tiempos de ejecución del sistema de control original con el sistema de control IEC-61499.

CONCLUSIONES

- Mediante la implementación de la norma IEC-61499 se desarrolló un sistema de control distribuido, basado en el concepto de bloques funcionales, logrando así la manipulación del robot KUKA youBot.

- El desarrollo de la aplicación en 4DIAC bajo la norma IEC-61499 se alineó a la cinemática del robot KUKA youBot, debido a que parámetros erróneos llevarían a una fatiga en el mecanismo.

- 4DIAC permitió crear envolturas que están sujetas en la norma IEC-61499, estas se desarrollaron en C++, permitiendo vincular librerías ya existentes y que se ejecuten bajo la misma norma.

- Al realizar la programación mediante bloques de función se puede crear sistemas flexibles y reconfigurables, ya que se puede realizar cambios de manera externa sin ingresar al código del FB.

- Se pudo evidenciar la necesidad de utilizar un estándar especializado en sistemas de control distribuido: IEC-61499. Dicha normativa, está basada en una arquitectura de referencia diseñada para facilitar el desarrollo de aplicaciones con lógica descentralizada, a través de elementos conocidos como bloques funcionales.

- El entorno de desarrollo 4DIAC y su plataforma de ejecución FORTE han demostrado ser ideales para la implementación de sistemas de control. De esta manera, se demuestra la alta compatibilidad entre el estándar IEC-61499 y los robots de nueva generación que trabajan con un software integrado.

- Llevando a cabo este proyecto también se ha demostrado la gran utilidad y buenas prestaciones que ofrecen el estándar IEC-61499 junto con las herramientas de software desarrolladas hasta la fecha para programar algoritmos de control de forma sencilla, rápida y eficaz.

- Mediante las pruebas de funcionamiento de la aplicación de control, se comprobó que los movimientos ejecutados en el robot KUKA youBot responden exitosamente, con una efectividad del 98,89% de aciertos.

- Se puede concluir que la simulación también es una herramienta muy útil en robótica para realizar pruebas iniciales del sistema implementado, ya que en ocasiones es difícil realizar la programación correcta para el control de robots. En este tipo de aplicaciones los errores cometidos en un entorno virtual durante la fase de desarrollo no provocan graves consecuencias.

RECOMENDACIONES

- Realizar pruebas iniciales en un entorno de simulación, para asegurar que los resultados obtenidos sean los adecuados y no ocasionar daños en el dispositivo al momento de ejecutarlo en el sistema real.

- Para implementar los algoritmos desarrollados desde 4DIAC-IDE, este software se debe estar ejecutando en los dispositivos implicados.

- Se debe comprobar la conectividad a internet del robot Kuka YouBot para realizar actualizaciones de software indispensables que requiere el estándar IEC-61499.

- Manejar variables del mismo tipo para entrelazar bloques compuestos creados por el usuario.

- Para obtener el 100% de efectividad en las pruebas de funcionamiento del sistema, detalladas en la sección 4.1, previamente se debe verificar la comunicación para evitar fallas de conexión.

FUTUROS DESARROLLOS

Para dar por terminada la presente memoria del Trabajo de Titulación**,** se proporcionarán pautas e ideas con las que continuar las investigaciones y desarrollos realizados con el proyecto en cuestión. Antes de enumerar nuevos recorridos y plantear la realización de desarrollos más complejos que asienten su base en los conceptos descritos a lo largo de estas páginas, se considera importante mejorar algunos aspectos y características:

- Crear una interfaz gráfica para la manipulación del robot.
- Intentar proporcionar al software de control del gripper comportamientos que permitan el autoaprendizaje para lograr agarres estables.
- Definir una tarea específica para el robot, con el fin de probar su efectividad.

Por otro lado, surgen numerosas ideas que parten de las bases establecidas por este proyecto, de entre las que destacan:

- Generación de FB de comunicación mediante el protocolo MQTT.
- Integración del procesamiento de imágenes con las bibliotecas Open CV.
- Análisis de un entorno y la determinación del mejor camino para evitar obstáculos.
- Desarrollo de un entorno de simulación virtual para la operación del robot Kuka youBot desde la nube.
- Adaptación del trabajo para agarrar objetos dinámicos o en movimiento.

BIBLIOGRAFÍA

Bischoff, R., Huggenberger, U., & Prassler, E. (2011). ICRA Communications KUKA youBot – a mobile manipulator for research and education, 3–6.

Catalan, C., Serna, F., Blesa, A., Rams, J. M., & Colom, J. M. (2011). Communication types for manufacturing systems. A proposal to Distributed Control System based on IEC 61499, 767–772.

Catalán, C., Serna, F., Blesa, A., & Zaragoza, U. De. (2010). IEC 61499 Execution Model Based on Life Cycle of Function Blocks.

Cengic, G. (2010). On Formal Analysis of IEC 61499 Applications , Part A : Modeling, *6*(2), 136–144.

Convention, H. K. (2014). The Application of Service-Oriented Architectures in Distributed Automation Systems, 252–257.

Dai, W., & Vyatkin, V. (2010). Redesign distributed IEC 61131-3 PLC system in IEC 61499 function blocks. *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2010*. https://doi.org/10.1109/ETFA.2010.5641239

Di Napoli, G., Filippeschi, A., Tanzini, M., & Avizzano, C. A. (2016). A novel control strategy for youBot arm. *IECON Proceedings (Industrial Electronics Conference)*, 482–487. https://doi.org/10.1109/IECON.2016.7793658

Doctoral, T. (2018). Sistemas de Control Distribuido, *2018*(c).

Dwiputra, R., Zakharov, A., Chakirov, R., & Prassler, E. (2014). Modelica Model for the youBot Manipulator (pp. 1205–1212). https://doi.org/10.3384/ecp140961205

Engineering, M. (2014). Philosophy Doctoral Thesis in Information Engineering Using Kuka Youbot for Teaching Assistance Supervisors : Professor Luca Iocchi and Professor Massimo Mecella, (November).

Florencia Gutiérrez Keen, Adolfo Chércoles, Walter Daniel Zalazar, Jorge García, & J. S. (2015). *Sistemas Distribuidos*.

Frey, G., & Hussain, T. (2006). Modeling techniques for distributed control systems based on the IEC 61499 standard-current approaches and open problems. *Discrete Event Systems, 2006 8th International Workshop On*, 176–181.

Garcia, C. A., Naranjo, J. E., Zambrano, T. P., Ambato, U. T. De, & Lanas, D. (2018). Low-Cost Cyber-Physical Production Systems for Industrial Control Robots under IEC 61499, 1281–1284.

Garcia, C. A., Salinas, G., Perez, V. M., Salazar L., F., & Garcia, M. V. (2019). Robotic Arm Manipulation Under IEC 61499 and ROS-based Compatible Control Scheme. In M. Botto-Tobar, L. Barba-Maggi, J. González-Huerta, P. Villacrés-Cevallos, O. S. Gómez, & M. I. Uvidia-Fassler (Eds.), *Information and Communication Technologies of Ecuador (TIC.EC)* (pp. 358–371). Cham: Springer International Publishing.

García, M. V, Irisarri, E., & Pérez, F. (2015). Industrial Communications Integration at shop floor level using OPC-UA and IEC-61499. Retrieved from http://ingenieria.ute.edu.ec/inciscos/assets/s2/INCISCOS_2016_paper_63.pdf

Hochgeschwender, N., Gherardi, L., Shakhirmardanov, A., Kraetzschmar, G. K., Brugali, D., & Bruyninckx, H. (2013). A Model-based Approach to Software Deployment in Robotics.

Hussain, T., & Frey, G. (2005). Migration of a PLC Controller to an IEC 61499 Compliant Distributed Control System : Hands-on Experiences, (April), 84–89.

Jakovljevic, Z., Mitrovic, S., & Pajic, M. (2017). Cyber Physical Production Systems—An IEC 61499 Perspective. *Lecture Notes in Mechanical Engineering*. https://doi.org/10.1007/978-3-319-56430-2_3

Kadera, P., Vrba, P., Biffl, S., Andrén, F., & Strasser, T. (2015). Applying Performance Analysis Concepts for Event-Based Industrial Automation Systems.

Karl, C., Gonzales, L., Ing, A., & Carrera, W. (2015). Control e Interfaz para un Brazo

Robótico.

Keiser, B. (2013). Torque Control of a KUKA youBot Arm, (September), 37.

Locomotec. (2011). KUKA youBot User Manual, 1–43.

Miguel, O. (n.d.). Implementación y evaluación de algoritmos de control basados en eventos en el estándar de programación de control distribuidos IEC-61499.

Mirelez-Delgado, F., Morales-Díaz, A., Ríos-Cabrera, R., & Pérez-Villeda, H. (2015). Control Servovisual de un Kuka youBot para la manipulación y traslado de objetos. *Congreso Nacional de Control Automático, AMCA 2015*, (2012), 239–244.

Mizuya, T. (n.d.). Representation and Implementation of Function Blocks by the Event Delegation Model, 2522–2525.

Nagatani, K., Tachibana, S., & Tanaka, M. S. Y. (n.d.). Improvement of Odometry for Omnidirectional Vehicle using Optical Flow Information.

Querol, E., Romero, J. A., Estruch, & A. M. (2014). *Norma iec-61499 para el control distribuido. aplicación al cnc.*

Pierucci, S., Klemeš, J. J., Piazza, L., Bakalis, S., Mikhalevich, S., Krinitsyn, N., & Baydali, S. (2017). Developing of KUKA youBot Software for Education Process. *Chemical Engineering Transactions*, *57*. https://doi.org/10.3303/CET1757263

Sanfilippo, F., Hatledal, L. I., & Zhang, H. (n.d.). JOpenShowVar : an Open-Source Cross-Platform Communication Interface to Kuka Robots ∗.

Sarkar, A. (2015). Distributed Control System Technologies- NODERED, CODESYS, 4DIAC, DOME. https://doi.org/10.13140/RG.2.1.3901.9609

Steinegger, M., Plaschka, N., Melik-merkumians, M., & Schitter, G. (n.d.). A Framework for Modular and Distributable Control of Reconfigurable Robotic Systems.

Stojmenovic, I. (2011). Access Control in Distributed Systems Merging Theory with Practice, 2009–2010. https://doi.org/10.1109/TrustCom.2011.1

Strasser, T., & Froschauer, R. (2012). Autonomous Application Recovery in Distributed Intelligent Automation and Control Systems, *42*(6), 1054–1071.

Suinder, C., Christensen, J. H., Brennan, R. W., Heigths, C., Auinger, F., & Martinez-, J. L. (2006). Usability and Interoperability of IEC 61499 based distributed automation systems Jhchristenseng, *61499*, 31–37.

Sünder, C., Zoitl, A., Christensen, J. H., Colla, M., & Strasser, T. (n.d.). Execution Models for the IEC 61499 elements Composite Function Block and Subapplication, 1169–1175.

Vyatkin, V. (2009a). Different Perspectives, (December), 7–23.

Vyatkin, V. (2009b). The IEC 61499 standard and its semantics - Bridging the Gap Between PLC Programming Languages and Distributed Systems. *Industrial Electronics Magazine, IEEE*, *3*(4), 40–48. https://doi.org/10.1109/MIE.2009.934796

Yoong, L. H., Roop, P. S., Vyatkin, V., Salcic, Z., & Member, S. (2009). A Synchronous Approach for IEC 61499 Function Block Implementation, *58*(12), 1599–1614.

Zhang, L., & Zhou, C. (2013). Kuka youBot arm shortest path planning based on geodesics. *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, (December), 2317–2321. https://doi.org/10.1109/ROBIO.2013.6739815

Zoitl, A., Smodic, R., & Sünder, C. (2006). Enhanced Real-Time Execution of Modular Control Soft- ware based on IEC 61499, (May), 327–332.

ANEXOS

ANEXO A

Programas Utilizados

En este anexo se detallan los programas que se han empleado para el desarrollo del proyecto en su faceta más técnica.

4DIAC: Programa encargado de realizar tanto la programación de aplicaciones como de los elementos que las constituyen, todo ello bajo el paradigma de programación de la norma IEC-61499.

Figura 1. Logotipo 4DIAC

CMake: Programa encargado de generar los proyectos que van a proporcionar la estructura para la ejecución de las diferentes aplicaciones desarrolladas a lo largo de este proyecto.

Figura 2. Logotipo CMake

Eclipse-neon: Programa para el desarrollo de partes específicas de las aplicaciones, así como para el desarrollo de la estructura de ejecución de las mismas.

Figura 3. Logotipo Eclipse

PuTTY: con este programa open-source se ha podido acceder a la PC a bordo del robot KUKA youBot, para lanzar las aplicaciones y obtener datos sobre dicho dispositivo.

Figura 4. Logotipo PuTTY

ANEXO B

Diseño de FBs

Como herramienta de desarrollo se ha optado por 4DIAC-IDE, concretamente la versión 1.8.4. Esta aplicación, además de tener buenas prestaciones, es de código libre y compatible con la mayoría de las herramientas actualmente existentes, siendo este uno de los objetivos del estándar IEC-61499.

A continuación se describe el desarrollo de FBs basados en C++ bajo el sistema operativo Windows.

- En la ventana <*System Explorer*> se crea un nuevo sistema como se muestra en la Figura 1.

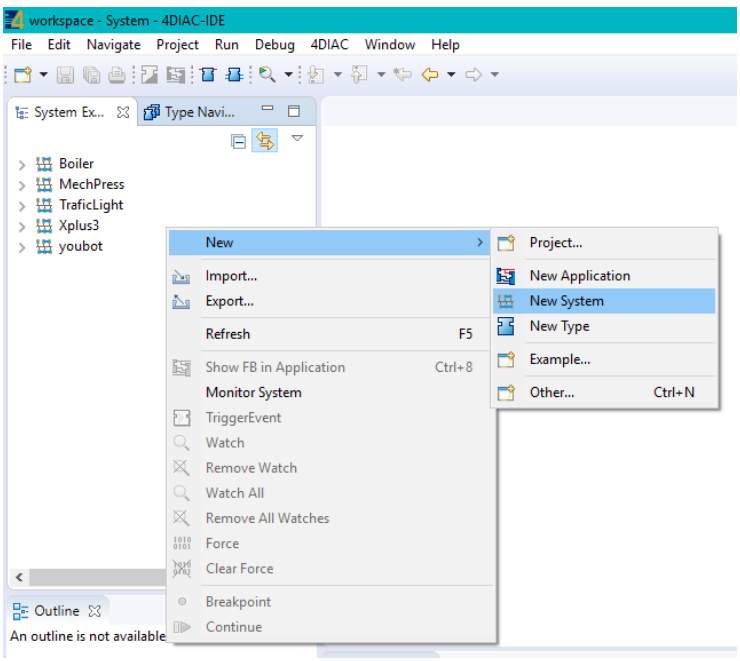

Figura 1: Creación de un nuevo sistema en 4DIAC-IDE

- Dentro del sistema creado en el paso anterior, se genera una nueva aplicación como se muestra en la Figura 2.

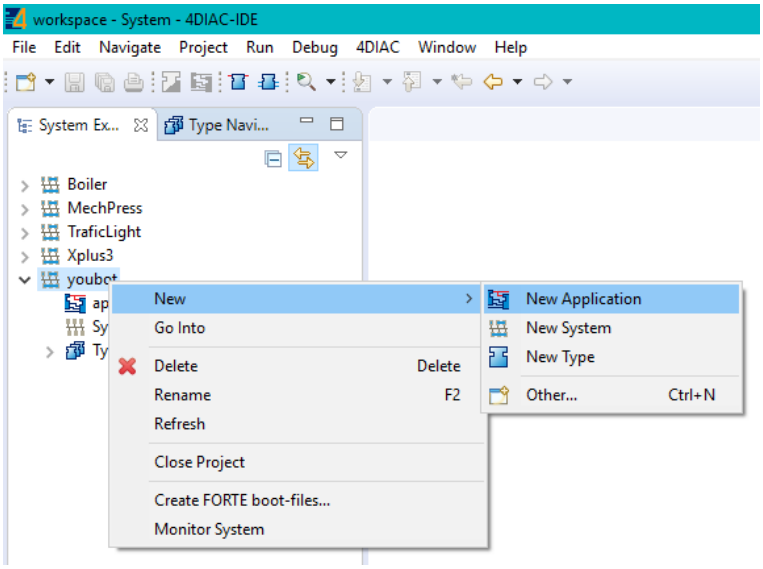

Figura 2: Generación de una nueva aplicación en 4DIAC-IDE

- En la ventana <*Type Navigator*> se ubica el sistema creado anteriormente y se genera una nueva carpeta como se muestra en la Figura 3.

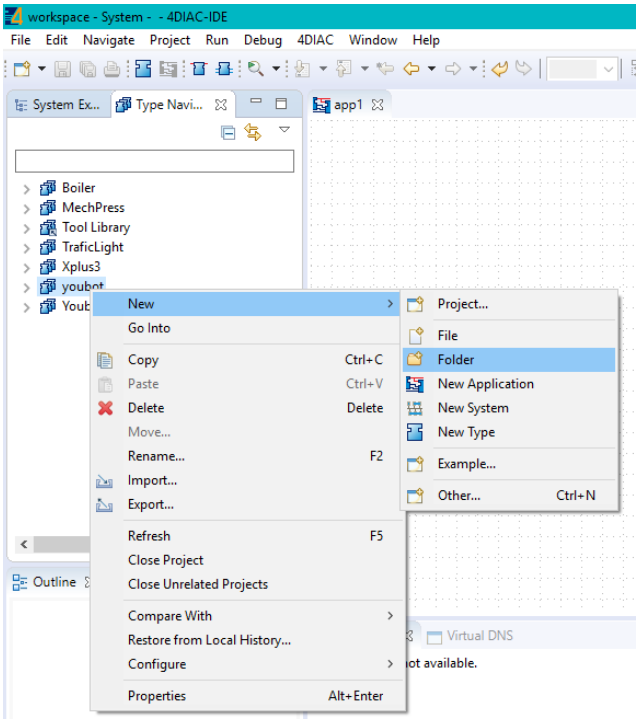

Figura 3: Generación de una nueva carpeta en 4DIAC-IDE

- Sobre la carpeta agregada dirigirse a <*New*>, seleccionar <*Type*> opción <*ServiceInterface.fbt*>, finalizado este proceso automáticamente el FB generado se desplegará en el área de trabajo como se muestra en la en la Figura 4.

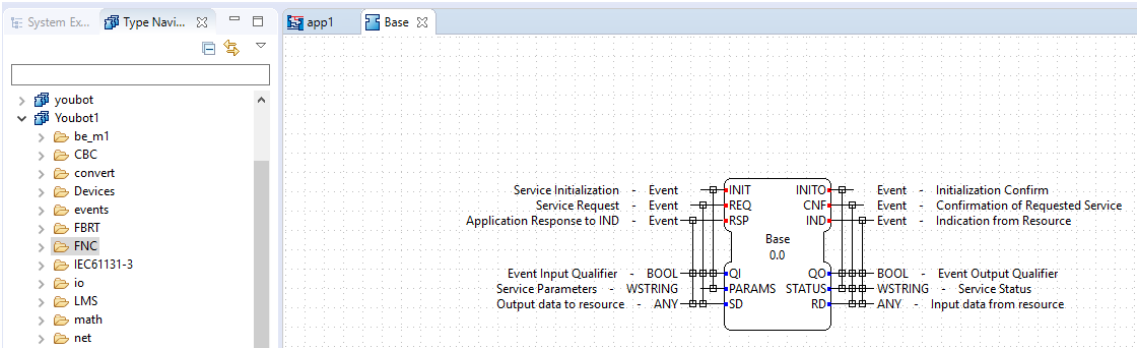

Figura 4: FB preparado para ser manipulado en el área de trabajo tras su creación

Con dicho procedimiento se ha generado una estructura vacía de FB, paso seguido se exporta para generar los archivos fuente como se observa en la Figura 5. Relacionados a dicho bloque de manera automática se generan dos archivos, uno de tipo .cpp en el cual se deben añadir los algoritmos de programación deseados y otro .h que contiene a manera de variables declaradas las entradas y salidas del bloque, estos archivos son compatibles con el lenguaje de programación C++.

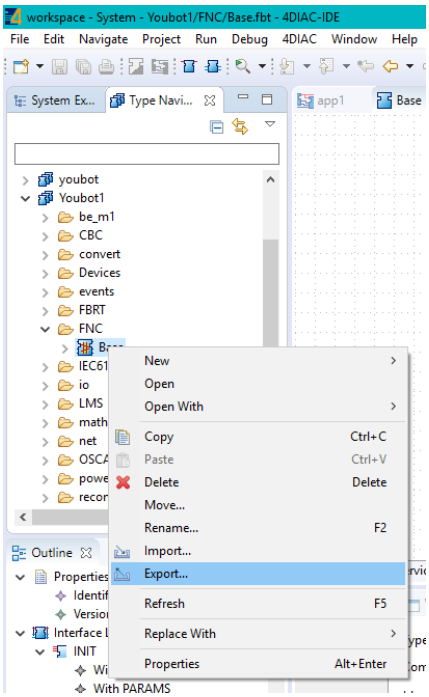

Figura 5: Selección de FBs a exportar

Utilizando un editor de C++, que en nuestro caso es Eclipse CDT, se ha modificado el archivo.cpp que 4DIAC-IDE nos entrega, definiéndose métodos IEC-61499 que enlazan los algoritmos en el hardware para acceso de las entrada y salidas del robot Kuka YouBot, también incluye las funciones para enlazar código C++ con código FORTE.

ANEXO C

Comunicación V-REP con ROS

Para la ejecución del programa de control y simulación es necesaria la instalación de diferentes librerías que utiliza V-REP. En este apartado describiremos el proceso seguido desde terminal para el sistema operativo Ubuntu.

1. El primer paso es descargar el repositorio ROS Hydro versión *"Precise",* de la página http://wiki.ros.org/hydro/Installation/ubuntu. Para ello introduciremos la siguiente línea en el terminal:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise
 main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Actualizar el Sistema Operativo, toma algo de tiempo hasta que se sincronice en los repositorios públicos mencionados en el paso anterior.

3. Obtener los password de ROS para activar el programa. Para ello introduciremos la siguiente línea en el terminal:

```
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.
key -O - | Sudo apt-key add -
```

4. Instalar ROS Hydro:

```
sudo apt-get install ros-hydro-desktop
```

5. Buscar librerías que estén instaladas en la máquina

6. Instalar los drivers del interface de youBot

7. Descargar V-REP de la página http://www.coppeliarobotics.com/downloads.html

8. Instalar V-REP en Linux Ubuntu

9. Una vez instalado el entorno de simulación, abrir una escena creada en youBot de base, para lo cual en el programa V-REP nos dirigimos a <*File/ open scene*>, como se muestra en la Figura 1.

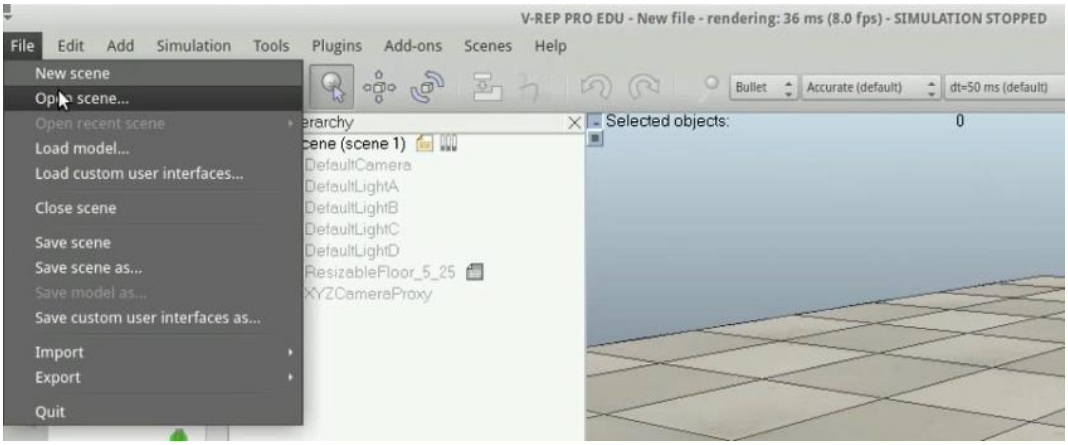

Figura 1

10. Abrir la página https://github.com/mfueller/vrep_youbot_plugin, e instalar los paquetes de youBot ROS, para lo cual introduciremos la siguiente línea en el terminal:

```
sudo apt-get install ros-indigo-youbot-driver-ros-interface
```

11. Clonar el repositorio en el espacio de trabajo de *catkin*

12. Copiar los paquetes vrep_common y vrep_plugin ros en el espacio de trabajo de catkin:

```
Cp –r/opt/v-rep/programming/ros_packages/vrep_plugin~/catkin_ws/
scr/$ cp -r/opt/v-rep/ programming/ros_packages/vrep_common~/cat
kin_ws/scr/
```

13. Compilar

```
$ catkin_make
```

14. Copiar las bibliotecas compiladas libv_repExtyouBot.so y libv_repExtRos.so en el directorio / opt / v-rep

```
sudo cp ~/catkin_ws/devel/lib/libv_repExt*/opt/v-rep
```

15. Empezar ros:

```
$ roscore
```

16. Cargar V-REP, traer el archivo de lanzamiento

```
$ roslaunch vrep_youbot_plugin vrep_youbot.launch
```

17. Iniciar V-REP, para lo cual introduciremos la siguiente línea en el terminal:

```
$/opt/v-rep/v-rep.sh
```

18. Cargue el archivo de escena en V-REP (... / vrep_youbot_plugin / scenes / ...)

19. Iniciar la simulación

20. Dirigirse a *file/open scene/akru/youBot_course_1.tt,* como se muestra en la Figura 2.

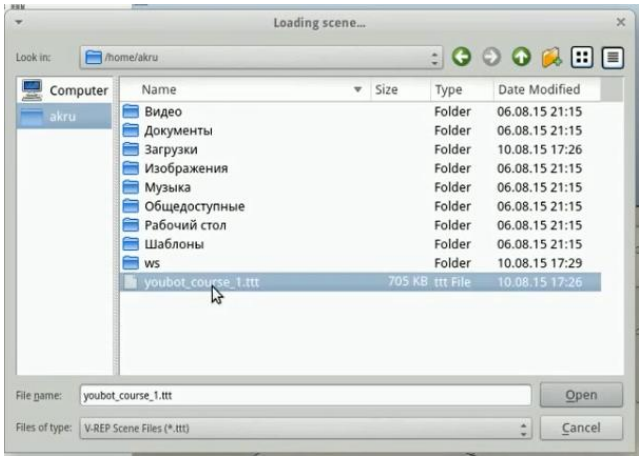

Figura 2

21. En el terminal ingresar los comandos:

```
ros^C
source/opt/ros/hydro/setup.bash
rosrun rviz rviz
```

22. Finalmente al ejecutar los comandos mencionados en el paso anterior queda lista la integración de V-REP para trabajar mediante ROS.