ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

IMPLEMENTACIÓN DE SISTEMAS DISTRIBUIDOS DE BAJO COSTO BAJO NORMA IEC-61499, EN LA ESTACIÓN DE CLASIFICACIÓN DEL MPS 500 DE LA UNIVERSIDAD POLITÉCNICA SALESIANA.

GUSTAVO JAVIER CAIZA GUANOCHANGA

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

Julio 2017

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: IMPLEMENTACIÓN DE SISTEMAS DISTRIBUIDOS DE BAJO COSTO BAJO NORMA IEC-61499, EN LA ESTACIÓN DE CLASIFICACIÓN DEL MPS 500 DE LA UNIVERSIDAD POLITÉCNICA SALESIANA, de responsabilidad del señor Gustavo Javier Caiza Guanochanga, ha sido minuciosamente revisado y se autoriza su presentación.

TRIBUNAL

Ing. Miguel Duque Vaca; M.Sc. …….
PRESIDENTE

Ing. Marcelo Vladimir García; M.Sc. …….
DIRECTOR DE TESIS

Ing. José Alcides Rumipamba; M.Sc. …….
MIEMBRO DEL TRIBUNAL

Ing. Cristian Andrés Yánez; M.Sc. …….
MIEMBRO DEL TRIBUNAL

Riobamba-Ecuador

2017

DERECHOS INTELECTUALES

Yo, Gustavo Javier Caiza Guanochanga soy responsable de las ideas, doctrinas y resultados expuestos en el Trabajo de Titulación modalidad Proyectos de Investigación y desarrollo, y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica de Chimborazo.

GUSTAVO JAVIER CAIZA GUANOCHANGA

No. Cédula: 1721192191

DERECHOS DE AUTOR COPYRIGHT

DECLARACIÓN DE RESPONSABILIDAD

Yo, Gustavo Javier Caiza Guanochanga, 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.

Riobamba, julio de 2017

_____

GUSTAVO JAVIER CAIZA GUANOCHANGA

No. Cédula: 1721192191

DEDICATORIA

Dedico este logro a mis padres Gonzalo y Consuelo por todo el esfuerzo y sacrificio que realizan día a día para salir adelante, por sus consejos y apoyo que me han dado en todas las etapas de mi vida. El tener el apoyo de mis padres hace posible que mis metas se puedan cumplir.

A mis hermanos por estar siempre a mi lado y apoyarme de manera incondicional.

AGRADECIMIENTO

Ante todo agradezco a Dios por brindarme salud y vida, para lograr mis metas propuestas.

Agradezco a la Escuela Superior Politécnica de Chimborazo, en especial al Instituto de Posgrados y Educación Continua, al personal docente quienes fueron parte de la formación profesional y humana.

Mis más sinceros agradecimientos a la Universidad Politécnica Salesiana por permitirme usar sus equipos de laboratorio para el desarrollo del proyecto.

En especial agradezco al Ing. Marcelo García M,Sc. Director de tesis, por guiarme en la elaboración del proyecto, gracias a su apoyo brindado para sacar este proyecto adelante.

CONTENIDO

PORTADA……..i
CERTIFICACIÓN……..ii
DERECHOS INTELECTUALES……..…...iii
DERECHOS DE AUTOR COPYRIGHT……..iv
DECLARACIÓN DE RESPONSABILIDAD……v
DEDICATORIA…….....vi
AGRADECIMIENTO……...……...vii
CONTENIDO……..……viii
ÍNDICE DE FIGURAS xii
ÍNDICE DE TABLAS.....xiv

CAPITULO I

1 INTRODUCCIÓN 1
1.1 PLANTEAMIENTO DEL PROBLEMA 1
1.2 FORMULACIÓN DEL PROBLEMA 1
1.3 SISTEMATIZACIÓN DEL PROBLEMA 1
1.4 JUSTIFICACIÓN DE LA INVESTIGACIÓN 2
1.5 OBJETIVOS 3
1.5.1 Objetivo General 3
1.5.2 Objetivos Específicos 3
1.6 HIPÓTESIS 3

CAPITULO II

2 MARCO TEÓRICO 4

2.1 ESTÁNDAR IEC-61499 4

2.2 MODELOS DE LA NORMA IEC-61499 6

2.2.1 Modelo de Recurso 6

2.2.2 Modelo de Dispositivo 6

2.2.3 Modelo de Sistema 7

2.2.4 Modelo de Aplicación 8

2.2.5 Bloque de Función 8

2.2.5.1 Tipos de Bloque de Función 9

2.2.5.2 Flujo de eventos 10

2.2.5.3 Flujo de datos 11

2.3 RED DE COMUNICACIÓN 11

2.4 ARQUITECTURA DEL CONTROL DISTRIBUIDO 12

2.5 SOFTWARE 12

2.5.1 Entorno de Desarrollo: FBDK 12

2.5.1.1 FBRT (Firmware Platforms) 13

2.5.2 ISaGRAF Workbench 13

2.5.2.1 ISaGRAF Runtime27 (Firmware Plataforms) 14

2.5.3 4DIAC-IDE / FORTE 14

2.5.3.1 FORTE (FIRMWARE PLATFORMS) 16

2.6 COMPARACIÓN DE HERRAMIENTAS Y PLATAFORMAS 17

2.7 CMAKE 18

CAPITULO III

3 METOLOGÍA 20

3.1 MPS 500 20
3.1.1 Estación de Manipulación *21*
3.1.1.1 Actuadores estación de manipulación 22
3.1.1.2 Sensores estación de manipulación 23
3.1.2 Estación de Clasificación 24
3.1.2.1 Actuadores estación de clasificación 25
3.1.2.2 Sensores estación de clasificación 27
3.2 RASPBERRY PI 28
3.2.1 Sistema Operativo y Software 28
3.2.2 Raspberry Pi 3 29
3.2.2.1 Características raspberry Pi 3B 30
3.2.2.2 GPIO raspberry Pi 3 B 30

CAPITULO IV

4 CASO DE ESTUDIO 32
4.1 DIAGRAMA DE FLUJO GENERAL DEL PROCESO 33
4.2 DESARROLLO DE BLOQUES DE FUNCIÓN 34
4.2.1 Diseño de los Bloques de Función 34
4.2.1.1 Bloques de función para los sensores 35
4.2.1.2 Bloques de Función para actuadores 36
4.2.1.3 Bloque de función del brazo 37
4.2.1.4 Bloque de función estación de clasificación 38
4.2.2 Exportación y Programación del Bloque de Función Creado 39
4.2.3 Ejecución del Runtime Forte 39
4.2.4 Configuración del CMake 40
4.3 DISEÑO DE LA APLICACIÓN DE CONTROL 40
4.4 DISEÑO DE LA ETAPA DE ACONDICIONAMIENTO 42
4.4.1 Acondicionamiento de la señal para sensores 42

4.4.2 Acondicionamiento de la señal para actuadores 43
4.4.3 Diseño e implementación de la placa para acondicionamiento. 44
4.5 IMPLEMENTACIÓN DEL SISTEMA 45
4.6 CONFIGURACIÓN Y FUNCIONAMIENTO DEL SISTEMA 45
4.7 ANÁLISIS DE COSTOS 48
CONCLUSIONES 50
RECOMENDACIONES 51
REFERENCIAS 52
ANEXOS ***..... 55***
ARTICULO CIENTIFICO ***..... 55***

ÍNDICE DE FIGURAS

Figura 1.2 Portabilidad, interoperabilidad y configurabilidad 5
Figura 2.2 Modelo de recurso 6
Figura 3.2 Modelo de dispositivo 7
Figura 4.2 Modelo de sistema 7
Figura 5.2 Modelo de aplicación 8
Figura 6.2 Bloque de función 10
Figura 7.2 Flujo de eventos 10
Figura 8.2 Flujo de datos 11
Figura 9.2 IEC-61499 FBDK 13
Figura 10.2 Software ISaGRAF 14
Figura 11.2 Software 4DIAC 15
Figura 12.2 Logo CMake 19
Figura 1.3 MPS 500 21
Figura 2.3 Estación de manipulación 21
Figura 3.3 Pinzas paralelas 22
Figura 4.3 Cilindro plano 23
Figura 5.3 Actuador lineal horizontal 23
Figura 6.3 Sensor óptico 24
Figura 7.3 Sensor de proximidad 24
Figura 8.3 Estación de clasificación 25
Figura 9.3 Electroválvula 26
Figura 10.3 Cilindro de carrera corta 26
Figura 11.3 Cilindro compacto 26
Figura 12.3 Motor DC 27
Figura 13.3 Sensor de Retro Reflexión 27
Figura 14.3 Sensor Inductivo 28
Figura 15.3 Raspberry Pi 3B 29
Figura 16.3 Pines GPIO 31
Figura 1.4 Implementación del proceso 32

Figura 2.4 Diagrama de Flujo 33
Figura 3.4 Desarrollo general 4DIAC 34
Figura 4.4 Bloque de función creado 35
Figura 5.4 Bloque de función del sensor de presencia. 36
Figura 6.4 Bloque de función del actuador 37
Figura 7.4 Bloque de función del brazo 38
Figura 8.4 Bloque de función de la estación de clasificación 39
Figura 9.4 Configuración del sistema 41
Figura 10.4 Aplicación estación de clasificación 41
Figura 11.4 Distribución de pines CNY74-4 42
Figura 12.4 Distribución de pines L293D 43
Figura 13.4 Pistas del circuito de acoplamiento 44
Figura 14.4 Circuito de acoplamiento 45
Figura 15.4 Configuración y descarga de la aplicación 46
Figura 16.4 Diseño de la aplicación 46
Figura 17.4 Visualización de las variables 47

ÍNDICE DE TABLAS

Tabla 1.2 Portabilidad de Librerías entre Software……..……...17

Tabla 2.2 Interoperabilidad de Elementos……...17

Tabla 1.3 Raspberry Pi 2 vs Raspberry Pi 3……...……..……...29

Tabla 1.4 Análisis de costos……..……...……...48

RESUMEN

El presente trabajo tiene como propósito la implementación de sistemas distribuidos de bajo costo bajo norma IEC-61499, en la estación de clasificación de la Universidad Politécnica Salesiana. El desarrollo de nuevas tecnologías, requerimientos de nuevas prestaciones y limitantes dadas por el fabricante de software y hardware para la automatización ha hecho que nuevos estándares sean desarrollados. La norma IEC-61499 es una de las últimas tendencias de automatización que estandariza un entorno de programación para sistemas distribuidos, tiene un alto nivel de versatilidad para el diseño de sistemas, ya que permite combinar software independiente del hardware utilizado. Para realizar la implementación del proceso se utiliza el software 4DIAC y se crean nuevos bloques de función para el control de sensores y actuadores, posteriormente se programó el runtime FORTE el cual fue ejecutado en una tarjeta raspberry Pi. Se evidencio en los resultados que realizar la programación mediante bloques de función se crea sistemas flexibles y reconfigurables. 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 dar prioridades en el orden de ejecución de los bloques de función. Además al realizar la implementación se logró evidenciar la reducción de costos en un 78% respecto al sistema implementado con PLC. Se recomienda en las futuras investigaciones realizar un circuito de acoplamiento en la tarjeta para acoplar la señal de sensores y actuadores.

PALABRAS CLAVES: <ESTANDAR (IEC-61499)>, <TARJETA RASPBERRY (PI)>, <RUNTIME FORTE>, <SISTEMA DISTRIBUIDO>, <SOFTWARE (4 DIAC)>.

SUMMARY

The purpose of this work is to implement low-cost distributed systems under IEC-61499, at the classification station of the Universidad Politécnica Salesiana. The development of new technologies, new performance requirements and limitations given by the software and hardware manufacturer for automation has made new standards developed. The IEC-61499 standard is one of the latest automation trends that standardizes a programming environment for distributed systems. It has a high level of versatility for system design, since it allows to combine software independent of the hardware used. To implement the process, the 4DIAC software is used and new function blocks are created for the control of sensors and actuators, later FORTE Runtime was then programmed and executed on a raspberry Pi card. It is evident from the results that the programming through function blocks creates flexible and reconfigurable systems. It is concluded that the standard allows to control the algorithm through events, since it has a connection between the execution of the event flow and data flow, in this way priorities can be given in the order of execution of the function blocks. In addition the implementation was possible to show the reduction of costs by 78% compared to the system implemented with PLC. It is recommended in future research to perform a coupling circuit in the card to couple the signal of sensors and actuators.

KEYWORDS: <DISTRIBUTED SYSTEM>, <STANDARD (IEC-61499)>, <SOFTWARE (4DIAC)>, < RASPBERRY CARD (PI)>, <RUNTIME FORTE>

CAPITULO I

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

La tendencia de utilizar software propietario, además del alto costo y limitación que tiene cada fabricante con sus productos (PLC), ha limitado a las industrias la compra y uso de una misma marca y como consecuencia a las características dadas por el fabricante, al trabajar con estos equipos se tiene la dificultad de crear sistemas reconfigurables que se puedan integrar y expandir según el requerimiento y sin realizar grandes cambios en el proceso.

1.2 FORMULACIÓN DEL PROBLEMA

¿Un sistema distribuido implementado con el estándar IEC-61499, puede reducir los costos de implementación y crear sistemas reconfigurables reemplazando los métodos actualmente utilizados?

1.3 SISTEMATIZACIÓN DEL PROBLEMA

- ¿Cuáles son los métodos actuales para realizar la automatización y control de un proceso?
- ¿Cómo se puede dejar de depender de un solo software de automatización?
- ¿Qué limitantes pueden producirse al usar hardware y software licenciado?
- ¿Qué tareas implicaría una posible ampliación o cambio de una característica en el proceso ya implementado?

1.4 JUSTIFICACIÓN DE LA INVESTIGACIÓN

En la actualidad la automatización de procesos industriales a nivel mundial está cambiando, ya que debido a la demanda se requieren mayores prestaciones en los procesos. La mayor parte de procesos industriales utilizan los controladores lógicos programables (PLC), y en estos sistemas las aplicaciones ya implementadas son difíciles de integrar y expandir. (Ivanova, Frey, & Batchkova, 2008, pp. 444-446)

Esto ha llevado al desarrollo de nueva tecnología y creación de nuevos estándares, como es el IEC-61499 el cual estandariza un entorno de programación para los sistemas de control distribuido.(Roberto, 2010, pp. 9-10)

En la actualidad existen varios métodos para la implementación de sistemas distribuidos, pero tienen un alto costo y están limitados por el software y hardware, por ejemplo el controlador de automatización programable (PAC) que utilizan software propio del fabricante.

Por lo que en el presente proyecto se va implementar un sistema distribuido utilizando un sistema embebido de bajo costo, se utilizara la tarjeta raspberry Pi 3B, y el sistema se implementara bajo la norma IEC-61499.

Mediante la norma IEC-61499 se tiene un alto nivel para el diseño de sistemas, ya que permite combinar componentes de software independientemente de hardware utilizado. La industria puede tener sus sistemas más flexibles, ya que el software no será limitado o bloqueado por el proveedor por lo cual se reducirá los costos. (Peltola, Christensen, Sierla, & Koskinen, 2007, pp. 482-485)

1.5 OBJETIVOS

1.5.1 Objetivo General

Implementación de sistemas distribuidos de bajo costo bajo norma IEC-61499, en la estación de clasificación del MPS 500 de la Universidad Politécnica Salesiana.

1.5.2 Objetivos Específicos

- Implementar un sistema distribuido de bajo costo bajo con la norma IEC-61499 utilizando el runtime 4DIAC_FORTE en la estación de clasificación del MPS 500.
- Diseñar nuevos bloques de funciones utilizando el software 4DIAC-IDE para manipular las entradas y salidas digitales de la tarjeta Raspberry Pi.
- Implementar una aplicación de control distribuido en la estación de clasificación del MPS 500 utilizando la tarjeta Raspberry Pi.

1.6 HIPÓTESIS

Es factible implementar un sistema distribuido de bajo costo con la norma IEC-61499, el cual reduzca el costo de implementación y además cree un sistema flexible y reconfigurable.

CAPITULO II

2 MARCO TEÓRICO

2.1 ESTÁNDAR IEC-61499

Su primera versión surgió en el año 2005, luego se fueron haciendo mejoras y la segunda versión salió en el año 2012, el estándar define una arquitectura y unos modelos para desarrollar aplicaciones reconfigurables de control distribuido. (Catalán, 2016, pp. 9-10)

El estándar IEC 61499 se basa, o es una mejora del estándar IEC 61131, sin embargo la semántica de los elementos de lenguaje no eran bien definidos en el apartado IEC 61131-3, y como consecuencia de esto, las diferentes herramientas de software interpretan de distinta el código ingresado.(Ing, Vladimir, Sánchez, González, & Iec-, 2013, pp. 10-11)

El estándar IEC 61499 define una metodología para el desarrollo de sistemas de control distribuidos, la arquitectura está organizada de manera jerárquica, está destinado a la portabilidad, interoperabilidad, reutilización y reconfiguración de aplicaciones distribuidas, además proporciona un modelo genérico, este modelo incluye procesos y redes de comunicación como un medio para dispositivos embebidos, recursos y aplicaciones. (Peltola et al., 2007, pp. 811-813)

El estándar IEC-61499 tiene un alto nivel para el diseño de sistemas, ya que permite combinar componentes de software independientemente de hardware utilizado, la unidad central de este estándar es el bloque de función, el cual permite encapsular algoritmos de control o comunicación, y estos puede ser escritos en varios tipos de lenguaje de programación por ejemplo, Java, C++, Phyton, etc. (Gárcia, Pérez, Calvo, López, & Morán, 2015,pp. 230-233)

Se puede tener varios dispositivos compatibles con la norma IEC-61499, los cuales puedan interactuar entre si y no dependan del fabricante, se tiene un entorno de ejecución de código abierto

como es el 4DIAC-RTE (Linde & Elec, 2016). En esta aplicación se puede desarrollar bloques de función de acuerdo a la aplicación y pueden ser ejecutados en cualquier dispositivo compatible para el trabajo de campo.

Con el desarrollo de nuevos estándares pretende que la industria pueda tener nuevas opciones para automatizar un proceso y así crear sistemas más flexibles, ya que el software no será limitado o bloqueado por el proveedor reduciendo en gran medida los costos de implementación (Sierla, Christensen, Koskinen, & Peltola, 2007, pp. 482-485)

En la Figura 1.2 se muestra las herramientas de software, plataformas y librerías que son indispensables para la adopción de la norma IEC-61499.(Christensen, Strasser, Vyatkin, & Zoitl, 2012, pp. 482-486.)

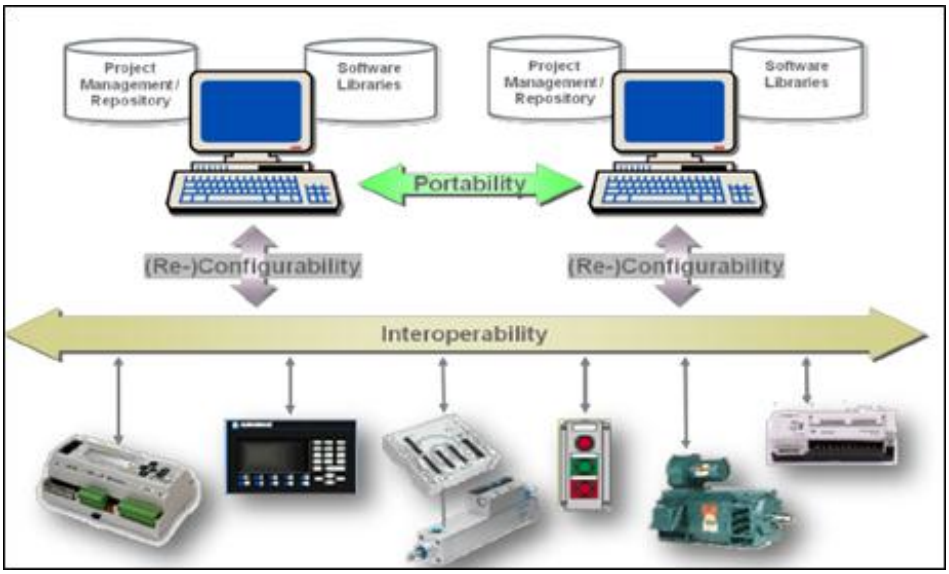

Figura 1.2: Portabilidad, interoperabilidad y configurabilidad
Fuente:(Christensen, Strasser, Vyatkin, & Zoitl, 2012.)

2.2 MODELOS DE LA NORMA IEC-61499

2.2.1 Modelo de Recurso

Representa un entorno para la ejecución independiente de las aplicaciones, es decir una parte del dispositivo es designada a ejecutar unos FBs determinados. La ejecución puede ser de recursos pertenecientes a la aplicación o también pueden ser de otras aplicaciones. Los recursos tiene acceso a las interfaces de comunicación y proceso del dispositivo.(Catalán, 2016, pp. 11-12).

En la Figura 2.2 se muestra los procesos que cumple un modelo de recurso.

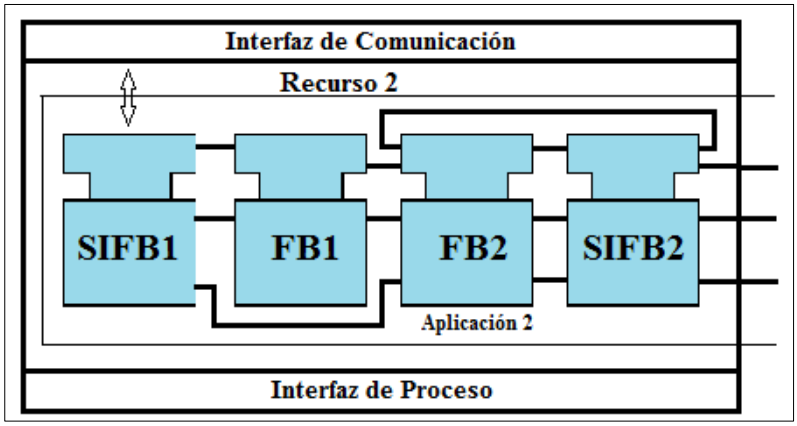

Figura 2.1: Modelo de recurso
Realizado por: "CAIZA GUSTAVO", 2017
Fuente:(Catalán, 2016)

2.2.2 Modelo de Dispositivo

Es la parte física en la cual se va ejecutar el programa, este dispositivo presenta sus propias características (interfaces de comunicación, periféricos de entrada y salida, etc.) de acuerdo al fabricante. Estos dispositivos pueden ser conectados a más de un segmento ya que posee dos interfaces:

Interfaz de proceso: La cual permite la comunicación entre dispositivos es decir tener acceso a los sensores y actuadores.

Interfaz de comunicación: Suministra servicios de comunicación entre los dispositivos y aplicaciones de procesos. (Garcia M., 2013, pp. 13-15)

El modelo de dispositivo se muestra en la Figura 3.2.

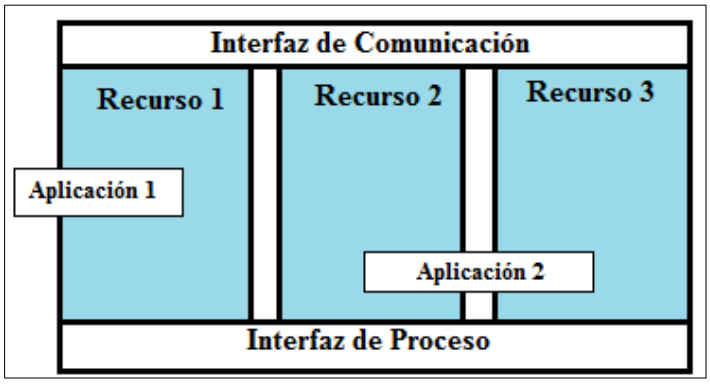

Figura 3.2: Modelo de dispositivo
Realizado por: "CAIZA GUSTAVO", 2017
Fuente:(Catalán, 2016)

2.2.3 Modelo de Sistema

El sistema es el elemento de mayor nivel el cual engloba todos los dispositivos y aplicaciones y su respectiva relación entre estos mediante alguna red de comunicación. Es decir describe los dispositivos interconectados y sus enlaces para las aplicaciones. En la Figura 4.2 se muestra los dispositivos conectados y los enlaces que tienen.

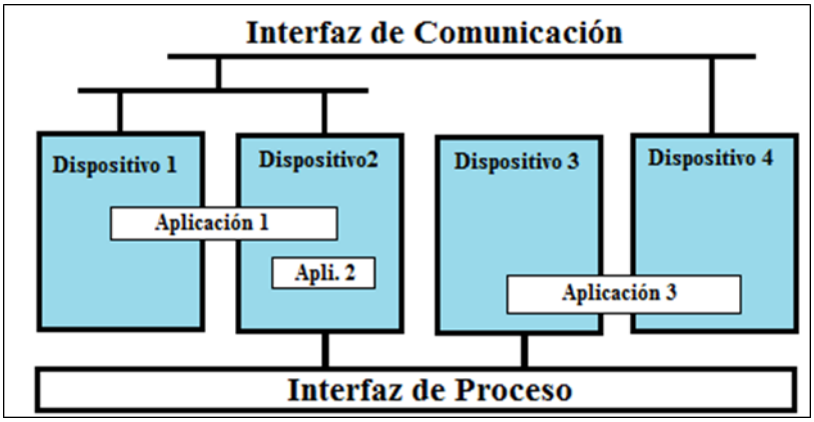

Figura 4.2: Modelo de sistema
Realizado por: "CAIZA GUSTAVO", 2017
Fuente:(Catalán, 2016)

2.2.4 Modelo de Aplicación

Es un conjunto de bloques de función interconectados para una aplicación. Las aplicaciones se organizan mediante redes de FBs las cuales son distribuidas en diferentes recursos, siendo los FBs elementos atómicos respecto a la distribución. (Catalan, 2016, pp. 11-12)

El modelo de aplicación y su red de FBs se muestra en la Figura 5.2.

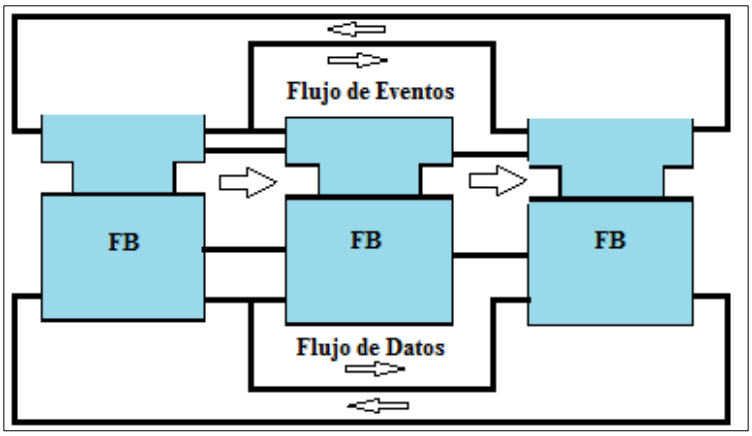

Figura 5.2: Modelo de aplicación
Realizado por: "CAIZA GUSTAVO", 2017
Fuente: (García., 2013)

2.2.5 Bloque de Función

El bloque de función proporciona una interfaz gráfica para eventos de entrada y salida, estos bloques pueden ser interconectados entre sí para de esta manera crear una aplicación, además tiene funciones internas las cuales procesan las entradas y de acuerdo al código interno genera las salidas.

Los bloques de funciones básicas incluyen gráficos de control de ejecución basados en eventos ECC (Execution Control Chart). Los elementos de la ECC son los estados, transiciones y acciones activadas por eventos, que controlan la ejecución de los algoritmos que se encuentran asociados al bloque de función. Una ECC puede desencadenar la ejecución de algoritmos dependiendo del evento dado, estas características permite en la automatización la fabricación de sistemas flexibles y reconfigurables. (Ivanova et al., 2008)

Los bloques de función se crean definiendo su ECC y programando su algoritmo, el ECC es una máquina de estado que procesa los eventos y algoritmos de programación, definiendo así el comportamiento del bloque de función.

2.2.5.1 *Tipos de Bloque de Función*

- **Bloque de Función Básico**

  Contiene una máquina de estado que controla la ejecución interna llamada ejecución Control Grafico (ECC), esta consta de tres partes; los estados, las acciones y las transiciones.

- **Bloque de Función Compuesto**

  Un bloque de función compuesta puede contener otros bloques de función, por lo tanto, los bloques de función de compuesta permiten metodologías de diseño modular.(Thramboulidis, Sierla, Papakonstantinou, & Koskinen, 2007, pp.177.179)

- **Bloque de Función de Interfaz de Servicio**

  Son la interfaz entre las aplicaciones y el hardware de los sistemas embebidos, es decir vincula los recursos para los subsistemas de procesos y comunicaciones.(Calvo, Pérez, Etxeberria, & Morán, 2010)

  Los bloques de función SIFBS proporcionan servicios a una aplicación como interacción entre aplicación y recurso. (García, 2013, pp. 13-15)

El bloque de función se observa en la figura 6.2 el cual permite la separación entre el flujo de eventos y flujo de datos, de esta manera se puede tener una mayor flexibilidad y organización al momento del desarrollo de aplicaciones.

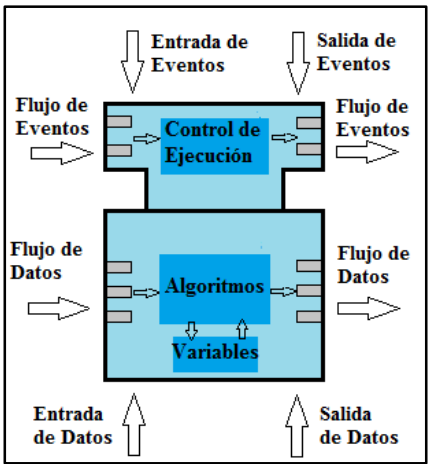

Figura 6.2: Bloque de función
Realizado por: "CAIZA GUSTAVO", 2017
Fuente: (García., 2013)

2.2.5.2 *Flujo de eventos*

Es una información instantánea que activa la ejecución de un FB, permitiendo configurar la entrada y salida de eventos, los cuales sincronizan la ejecución de los bloques de función de acuerdo a la aplicación, de esta manera permite una mayor optimización en el diseño del algoritmo.

El diseño en el bloque de función permite tener varios eventos los cuales se ejecutan de acuerdo a la programación, así los eventos solo se distribuirán a los lugares asignados. La aplicación utiliza una pequeña cantidad de procesamiento del controlador, a su vez acelera el tiempo de respuesta del mismo (Linde & Elec, 2016, pp.1-6). En la Figura 7.2 se muestra el flujo de eventos en color rojo.

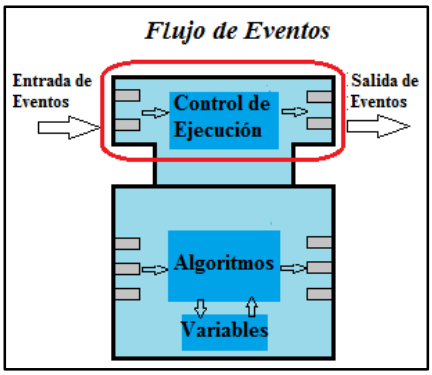

Figura 2.2 Flujo de eventos
Realizado por: "CAIZA GUSTAVO", 2017

2.2.5.3 *Flujo de datos*

Los bloques de función permiten enviar flujos de datos, estos datos de entrada están disponibles de manera directa en el bloque de función y se pueden ejecutar algoritmos con estos datos, luego estos datos son enviados de acuerdo a la confirmación de los eventos, el flujo de datos en un FB se puede observar de color azul en la Figura 8.2.

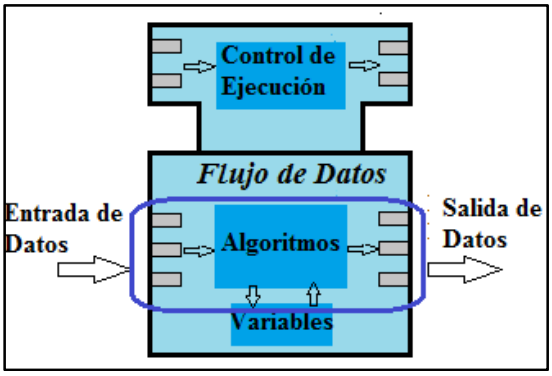

Figura 8.2: Flujo de datos
Realizado por: "CAIZA GUSTAVO", 2017

2.3 RED DE COMUNICACIÓN

El estándar define un servicio de comunicación, que puede comunicarse con una parte de la aplicación dentro del mismo dispositivo o con una aplicación externa ubicada en un dispositivo remoto. La interface de los FBs permite administrar recursos de esta manera es posible cambiar el estado del recurso y a su vez configurar el mismo recurso.

Las dos formas principales de comunicación son PUBLISH/SUBSCRIBE y CLIENT/SERVER. El PUBLISH publica la información en la red mientras que el SUBSCRIBE recibe esta información. El CLIENT/SERVER en cambio es u protocolo punto a punto (Roberto, 2010)

2.4 ARQUITECTURA DEL CONTROL DISTRIBUIDO

En un sistema distribuido es muy importante el tipo de comunicación que se utilice. En el caso de los sistemas de control distribuido debe ser una comunicación en tiempo real. En la norma IEC-61499 ha sido propuesto el uso de redes y protocolos específicos: RT-CORBA, MODBUS-TCP, RTPS y CIP. Pero hay una convergencia al uso de los estándares de comunicación de propósito general Ethernet e IP.(Catalán, 2016, pp. 15-16)

Se puede decir que una estación de trabajo es un conjunto de equipos que funcionan coordinadamente para lograr un objetivo común, dada espacio de trabajo tiene sus entradas y salidas, sistemas de transporte, máquinas de procesamiento, robots y almacenamiento. (Trejo-hernández & López-mellado, 2013, pp. 180-181)

2.5 SOFTWARE

A continuación se describe varias herramientas de Software actualmente utilizadas para la implementación de la norma IEC-61499.

2.5.1 Entorno de Desarrollo: FBDK

Eta herramienta es original de la norma IEC-61499, fue desarrollada inicialmente como un applet Java8 simple para dibujar bloques de función y redes de FBs, luego los creadores modificaron la herramienta para probar el modelo de gráficos y el formato de intercambio de archivos XML definidos en el estándar IEC-61499-2 y al mismo tiempo se utilizó en el desarrollo de la primera demostración de viabilidad del IEC-61499 por el consorcio Holonic Manufacturing System. (Christensen et al., 2012.)

En la Figura 9.2 se muestra el entorno de desarrollo FBDK.

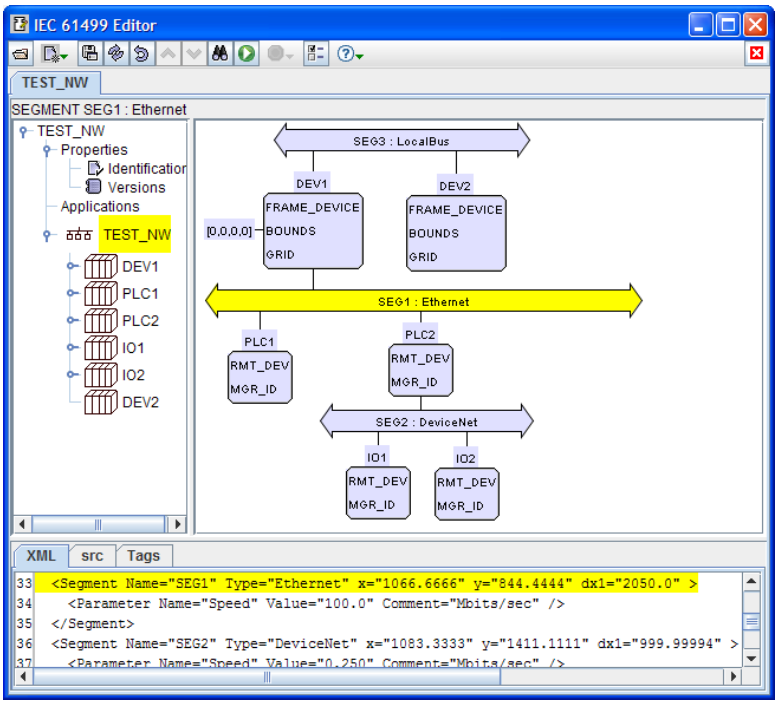

Figura 9.2: IEC-61499 FBDK
Fuente: (Christensen, Strasser, Vyatkin, & Zoitl, 2012.)

2.5.1.1 *FBRT (Firmware Platforms)*

Es un runtime que fue desarrollado en lenguaje JAVA y utilizado para pruebas de factibilidad y demostración en la norma IEC-61499 utilizando la tecnología Java incorporada de lmsys Technologies 22. Elementos de librería y aplicaciones basadas en la plataforma Java SE, la próxima versión del FBDK incluirá la generación de páginas web que se incorporen en este Applet.(Christensen et al., 2012.)

2.5.2 ISaGRAF Workbench

En diciembre de 2005, ICS Triplex ISaGRAF, una empresa de Rockwell Automation, lanzo la versión de ISaGRAF Workbench para el estándar IEC 61131-3, convirtiéndose así en la primera herramienta comercial de software que soporta el estándar IEC-61499. (Christensen et al., 2012.)

En la Figura 10.2 se muestra la versión 6 de ISaGRAF Workbench.

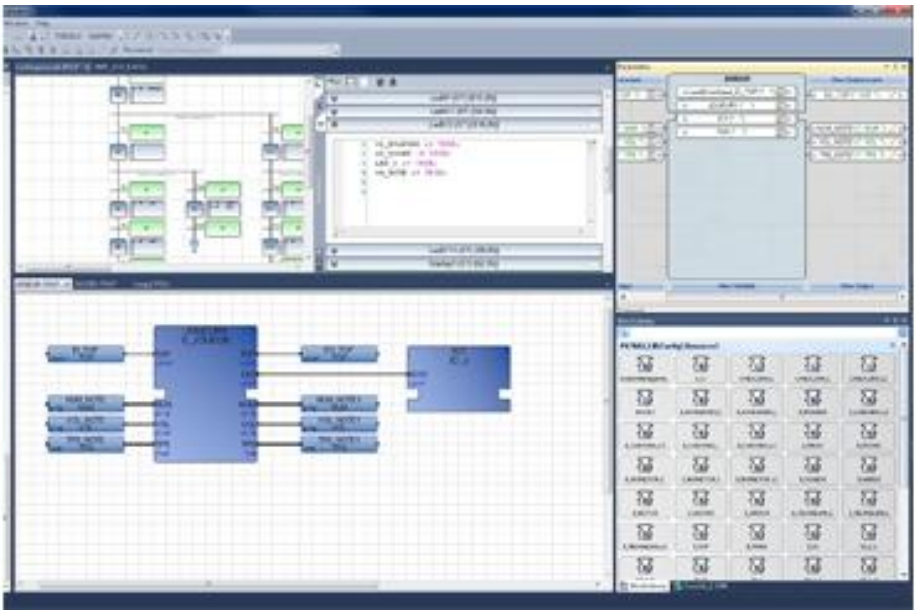

Figura 10.2: Software ISaGRAF
Fuente:(Christensen, Strasser, Vyatkin, & Zoitl, 2012.)

2.5.2.1 *ISaGRAF Runtime27 (Firmware Plataforms)*

Es un runtime basado en máquinas virtuales para el estándar IEC 61131-3, luego ha añadido soporte para la norma IEC-61499. Está certificada por TÜV Süd28 para ser configurable con ISaGRAF Workbench e interoperable entre dispositivos que utilicen la mismo runtime como plataforma. No se ha demostrado la interoperabilidad con otras plataformas de runtime y la configurabilidad mediante otras herramientas de software.(Christensen et al., 2012.)

2.5.3 4DIAC-IDE / FORTE

4DIAC (For Distributed Industrial Automation and Control). Se basa en el estándar IEC 61499 - Bloques de Función, el cual es un modelo de referencia para la automatización y control distribuido bajo licencia pública Eclipse. El objetivo de la iniciativa 4DIAC es proporcionar un estándar abierto y gratuito para la automatización y también fomentar la investigación.(Zoitl, Strasser, & Ebenhofer, 2013, pp. 358-360)

Es un entorno de ingeniería extensible en base a la norma IEC-61499 para aplicaciones de control distribuido, estas aplicaciones pueden ser descargadas en dispositivos campo que soporten esta norma, además permite una fácil integración de otros plug-ins.

Es una herramienta de software que se distribuye como un conjunto de complementos para el entorno de desarrollo integrado (IDE) de eclipse. El 4DIAC-IDE soporta la especificación de bloques de función así como el desarrollo de configuraciones de dispositivos así como la aplicación de sistemas distribuidos. (Christensen et al., 2012.pp. 482,485)

En la Figura 11.2 se muestra el IDE 4DIAC.

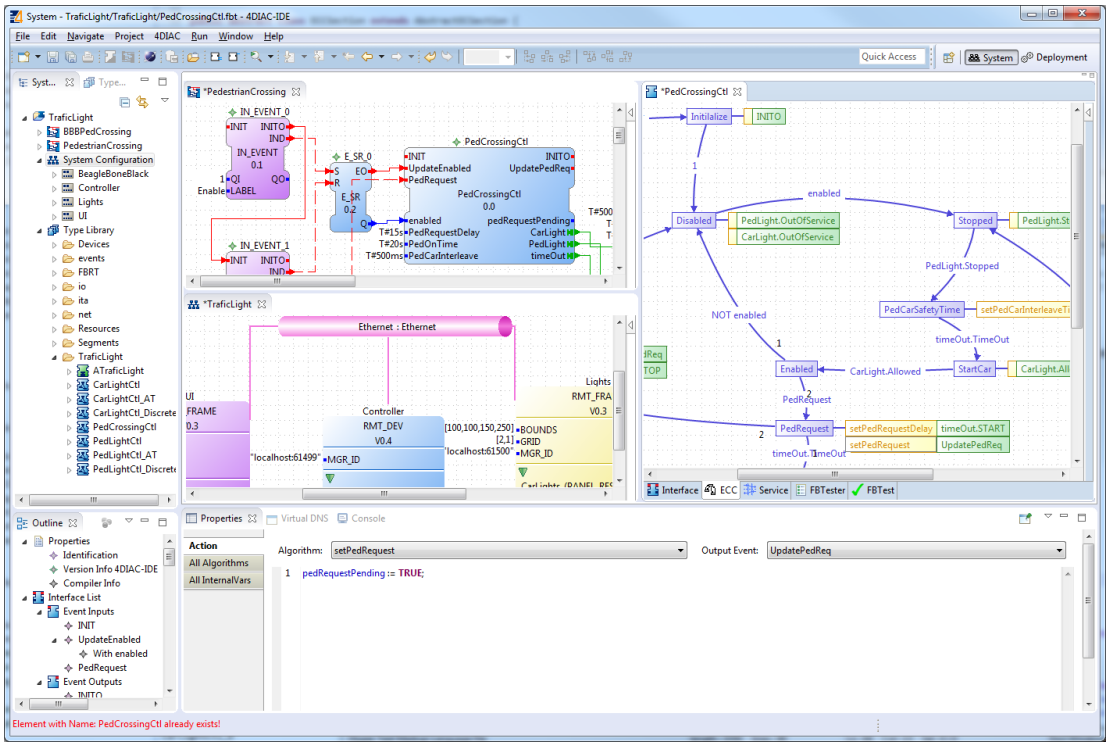

Figura 11.2: Software 4DIAC
Fuente: (Eclipse, 4DIAC, 2017)

Características:

- **Explorador del sistema:** Gestiona la aplicación de proyectos bajo la norma IEC-61499, una configuración de sistema y proyectos específicos tipo librería.

- **Editor de aplicaciones:** Modelando redes de bloque de función IEC-61499 en los elementos de la librería.
- **Editor de hardware:** Especificación y parametrización de hardware de automatización para dispositivos bajo norma IEC-61499.
- **Tipo Editores:** Creación de bloques de función básicos (BFB), FBs compuestos (CFB), FB de interfaz de servicio (SIFB) y adaptadores.
- **Editor de distribución:** Mapeo de FBNs o FBs individuales a los recursos especificados.
- **Implementación**: Descarga selectiva de aplicaciones a sus recursos correspondientes.
- **Funcionalidad de monitoreo y depuración:** permite ver y forzar los valores de los elementos de la interfaz, así como los eventos de disparo.
- **Funcionalidades de prueba:** Realiza el test de un FB, ya sea mediante la configuración manual de valores de datos y desencadenando eventos dentro del editor FBTest. (Eclipse, 2017)

2.5.3.1 *FORTE (FIRMWARE PLATFORMS)*

El entorno de ejecución de 4DIAC-RTE, FORTE es una pequeña implementación portable de un entorno de ejecución IEC-61499 el cual está enfocado a pequeños dispositivos (16/32 bits) implementado en C++.

FORTE proporciona una infraestructura de comunicación flexible a través de las llamadas capas de comunicación. Además, proporciona una arquitectura escalable que permite a FORTE adaptarse al diseño de la aplicación. (Eclipse, 4DIAC, 2017)

- **Capas de comunicación compatibles:**
  - Eclipse SCADA SFP
  - Ethernet (TCP/UDP)
  - Ethernet PowerLink
  - FBDK ASN.1
  - Modbus TCP client
  - MQTT

- OPC UA
- OPC DA

- **Tarjetas Soportadas:**
  - BeagleBone Black
  - Digi Connect ME
  - KIPR's CBC v2
  - Lego Mindstorms EV3
  - Raspberry PI
  - EmBRICK
  - NXH 51-ETM

- **Sistemas Operativos Compatibles**
  - Ecos
  - NET + OS 7
  - Poxis: Cygwin, Linux (i386,PPC,ARM)
  - Windows

- **Controladores Lógicos Programables compatibles**
  - Bachmann electronic M1 PLC
  - CONMELEON C1
  - Raspberry-SPS
  - WAGO PFC200

2.6 COMPARACIÓN DE HERRAMIENTAS Y PLATAFORMAS

A continuación se realiza una comparación entre los diferentes Software utilizados en la norma antes descrita, las herramientas y atributos que ofrecen al usuario. En la tabla 1.2 se muestra la portabilidad de elementos de librería entre las herramientas de software.

Tabla 1.2: Portabilidad de Librerías entre Software

	FBDK	**4DIAC**	**nxtSTUDIO**	**ISaGRAF**
FBDK	x	X	X	
4DIAC	x	X	X	
nxtSTUDIO	x	X	X	
ISaGRAF				X

Fuente: (Christensen et al., 2012.)
Realizado por: "CAIZA GUSTAVO", 2017

En la tabla 2.2 se muestra la comparación de interoperabilidad entre elementos.

Tabla 2.1: Interoperabilidad de Elementos

	FBDK	**4DIAC**	**nxtSTUDIO**	**ISaGRAF**
FBDK	x	X	x	
4DIAC	x	X	x	
nxtSTUDIO	x	X	x	
ISaGRAF				X

Fuente: (Christensen et al., 2012.)
Realizado por: "CAIZA GUSTAVO", 2017

2.7 CMAKE

Es una herramienta de código abierto y multiplataforma diseñada para construir, probar y empaquetar software, soporta entornos complejos que requieren configuración de sistema. Se utiliza para controlar el proceso de compilación del software utilizando archivos de configuración independientes de plataforma, generar makefiles y se puede utilizar el entorno de compilación de

escoja el usuario. El conjunto de herramientas fueron creadas por KitWare en respuesta a la necesidad de un potente entorno de compilación multiplataforma. (Kitware, 2017)

Figura 12.2: Logo CMake
Fuente: (Kitware, 2017)

Es una herramienta de compilación de código abierto que ha estado en desarrollo desde el 2000. CMake pretende simplificar la construcción de plataforma cruzada, es decir en lugar de tener archivos de compilación separados para cada plataforma Cmake solo necesita un solo conjunto de archivos de entrada para controlar la compilación para todas las plataformas.(Hoffman, Cole, Park, & Vines, 2010, pp. 378-381).

CAPITULO III

3 METOLOGÍA

3.1 MPS 500

MPS 500-FMS (Modular Production System) es una estación de producción flexible-compatible, modular y versátil de marca FESTO. Forma la base para la formación técnica en general, utilizando problemas prácticos de aplicaciones reales, Permite la interacción de la mecánica, neumática, ingeniería eléctrica, tecnología de control y las interfaces de comunicación. (Festo, MPS 500-FMS, 2017)

El sistema de transporte es el componente central del sistema ya que es el que se encarga de conectar las estaciones, y tiene las siguientes características.

- Seis posiciones de trabajo
- Identificación de pallet
- Tecnología de terminal de válvulas
- Bus AS-i
- PLC S7-300
- Variador de frecuencia
- Motores AC
- Paro de emergencia
- Puertos E/S por estación (Festo, MPS 500-FMS, 2017)

El modulo consta de seis estaciones, cada estación se controla con dos PLCs S7-300 y están unidas por un sistema de trasporte (Conveyor) las estaciones son: Distribución, Verificación, Procesamiento, Montaje, Almacenamiento y Clasificación. En la Figura 1.3 se muestra todas las estaciones que componen el MPS 500.

Figura 1.3: MPS 500
Fuente: (Festo, MPS 500-FMS, 2017)

3.1.1 Estación de Manipulación

La estación es de control electro neumático y está equipada con un manipulador flexible de dos ejes ("x", "y"). Este manipulador cuenta con sensores los cuales se ubican de manera estratégica para saber la posición del brazo. Las piezas que llegan a la pinza neumática son detectadas por un sensor reflectivo y son transportadas mediante el brazo hacia una posición asignada. En la Figura 2.3 se muestra la estación de manipulación.

Figura 2.3: Estación de manipulación
Fuente: (Festo, MPS 500-FMS, 2017)

A continuación se describen los sensores y actuadores utilizados en el proyecto para la Estación de Manipulación.

3.1.1.1 *Actuadores estación de manipulación*

- **Pinzas paralelas**

Es una pinza de doble efecto la cual tiene adaptado un dispositivo mecánico para convertir el movimiento longitudinal en movimiento transversal y así pueda sujetar las piezas.

En la Figura 3.3 se muestra las pinzas paralelas utilizadas en la estación.

Figura 3.1: Pinzas paralelas
Fuente: (Festo, MPS 500-FMS, 2017)

- **Cilindro plano EZH**

Es un cilindro de simple efecto compacto de forma extremadamente plana para tener un mejor montaje en bloque y se utiliza para bajar y subir el brazo, la imagen del cilindro se muestra en la Figura 4.3.

Figura 4.3: Cilindro plano
Fuente: (Festo, MPS 500-FMS, 2017)

- **Actuador lineal horizontal**

Se encarga de llevar las piezas de la estación de transporte hacia la estación de clasificación, las cargas y dispositivos pueden ser montados en la corredera y los sensores pueden ser colocados de acuerdo a la aplicación sobre el actuador. La imagen de actuador se muestra en la Figura 5.3.

Figura 5.3: Actuador lineal horizontal
Fuente: (Festo, MPS 500-FMS, 2017)

3.1.1.2 *Sensores estación de manipulación*

- **Sensor óptico de reflexión directa**

El sensor está ubicado para detectar cuando existe un objeto en la pinza, presenta las siguientes características: distancia nominal de conmutación es de 30mm, salida de conmutación pnp y tiene un conductor de fibra óptica como lo muestra la Figura 6.3.

Figura 6.3: Sensor óptico
Fuente: (Festo, MPS 500-FMS, 2017)

- **Sensor de proximidad Magnético**

Están diseñados para adaptarlos en los actuadores de marca Festo, la distancia de conmutación es de 4mm. Detectan el campo magnético y está insertado en la ranura longitudinal del actuador lineal horizontal, para saber la posición del brazo. La Figura 7.3 muestra la forma del sensor de proximidad.

Figura 7.3: Sensor de proximidad
Fuente: (Festo, MPS 500-FMS, 2017)

3.1.2 Estación de Clasificación

La estación tiene tres rampas para clasificar las piezas y una rampa para desechar. Cuando las piezas llegan a la banda transportadora son detectadas por un sensor de reflexión directa, luego de eso se tiene otros sensores para detectar el color de la pieza (negra, roja, plateada). Cuando el color es detectado se tiene varios actuadores para realizar la clasificación según el algoritmo de programación. Los desviadores clasificadores, accionados por cilindros de carrera corta realizan la distribución de piezas a cada rampa, y finalmente se tiene un sensor de retro reflexión para detectar

cuando la rampa está llena. (Festo, Estacion de Clasificación:Final, 2017). En la Figura 8.3 se muestra la estación de clasificación.

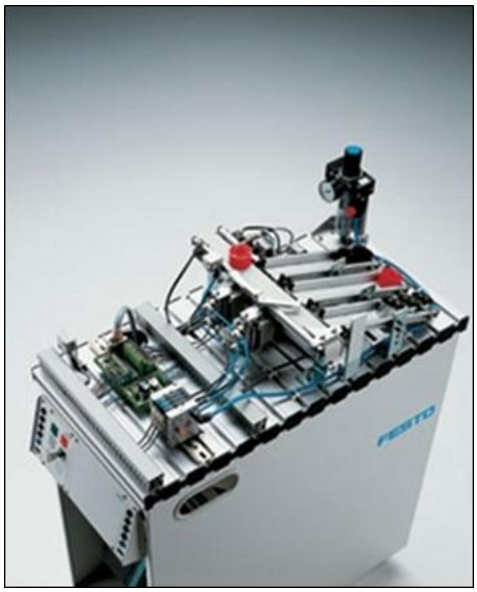

Figura 8.3: Estación de clasificación
Fuente: (Festo, MPS 500-FMS, 2017)

A continuación se describe los sensores y actuadores utilizados en el proyecto en la estación de clasificación.

3.1.2.1 *Actuadores estación de clasificación*

- **Electroválvula Neumática**

Tiene terminales de válvulas para funciones múltiples, permiten la distribución de aire comprimido para el control de los actuadores como lo muestra la Figura 9.3.

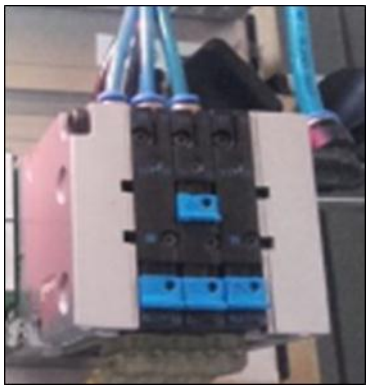

Figura 9.3: Electroválvula
Realizado por: "CAIZA GUSTAVO", 2017

- **Cilindro de carrera corta**

Consta de un cilindro simple efecto el cual es activado mediante la electroválvula, es utilizado para detener las piezas mientras los sensores detectan el color, para luego hacer la selección. Su imagen se muestra en la Figura 10.3.

Figura 10.3: Cilindro de carrera corta
Fuente: (Festo, MPS 500-FMS, 2017)

- **Cilindro compacto**

Tiene un cilindro doble efecto de carrera corta el cual tiene guías para realizar la clasificación a las diferentes rampas, además está provisto con sensores para detectar la posición en la que se encuentra en actuador como se puede ver en la Figura 11.3.

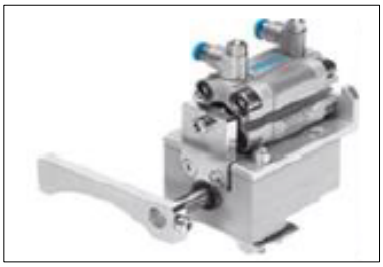

Figura 11.3: Cilindro compacto
Fuente: (Festo, MPS 500-FMS, 2017)

- **Motor DC**

El motor DC funciona con un voltaje de 24v y una corriente de 1.5 A, está provisto de acoplamientos y accesorios para el montaje en la banda trasportadora del módulo, en la Figura 12.3 se puede ver su estructura externa.

Figura 12.3: Motor DC
Fuente: (Festo, MPS 500-FMS, 2017)

3.1.2.2 *Sensores estación de clasificación*

- **Sensor de Retro Reflexión**

Es un sensor difusor de fibra óptica el cual se encarga dar una señal cuando pase cualquier pieza sin importar el color. Su imagen se muestra en la Figura 13.3.

Figura 10.2: Sensor de Retro Reflexión
Fuente: (Festo, MPS 500-FMS, 2017)

- **Sensor inductivo**

Detecta una amplia gama de metales, la distancia de conmutaciones de 4mm de salida PNP y es más rápido que el detector de proximidad. La imagen y forma del sensor se muestra en la Figura 14.3.

Figura 14.3: Sensor Inductivo
Fuente: (Festo, MPS 500-FMS, 2017)

3.2 RASPBERRY PI

Es un pequeño ordenador de placa reducida fabricado en Reino Unido por la Fundación Raspberry Pi, la primera versión fue lanzada al mercado en el año 2012 con el propósito de inspirar la enseñanza de ciencias de computación en los institutos de educación básica.(FOUNDATION, 2017). La tarjeta trabaja bajo el sistema operativo Debian/Linux, el mismo que permite trabajar con una librería de cogido abierto OpenCV (Open Source Computer Vision Library).

Actualmente la tarjeta debido a sus altas prestaciones, tamaño reducido, bajo consumo de energía y bajo costo es muy utilizada en la educación y también para la implementación de diferentes proyectos en diferentes áreas, por ejemplo: investigación, domótica, procesamiento de imágenes, comunicaciones inalámbricas, etc.

En el presente proyecto a implementar se va utilizar la tarjeta Raspberry Pi 3B debido a sus altas y mejoradas prestaciones respecto a los modelos anteriores, además se debe controlar una gran cantidad por lo que los modelos anteriores no cuenta con la cantidad de pines suficientes. La tarjeta dispone de 40 pines GPIO, para el control de sensores y actuadores. Se realizara la programación y se utiliza las entradas y salidas digitales GPIO (General Purpose Input/Output) para el control de un proceso de automatización.

3.2.1 Sistema Operativo y Software

El procesador permite que se ejecute sobre LINUX, snappy UBUNTU y Windows 10, el procesador Pi requiere su propio sistema operativo (SO) por ejemplo: Raspbian o NOOBS que son las versiones diferentes existentes que pueden ser implementadas.(Sahitya, Lokesha, & Sudha, 2016)

Para poder descargar el sistema operativo a utilizar se debe tener una tarjeta micro SD, la cual se coloca en la raspberry Pi y luego se realiza la instalación y configuración deseada según la aplicación.

El análisis de software ocupa la plataforma Open CV que soporta diferentes lenguajes de programación como: C, C++, Phyton, Java, Android, etc. (Sahitya et al., 2016)

3.2.2 Raspberry Pi 3

Es la tercera generación de Raspberry Pi y lanzada al mercado en febrero de 2016 por el fabricante. Tiene completa compatibilidad con los modelos anteriores 1 y 2. (FOUNDATION, 2017). En la Figura 3.15 se muestra la tarjeta.

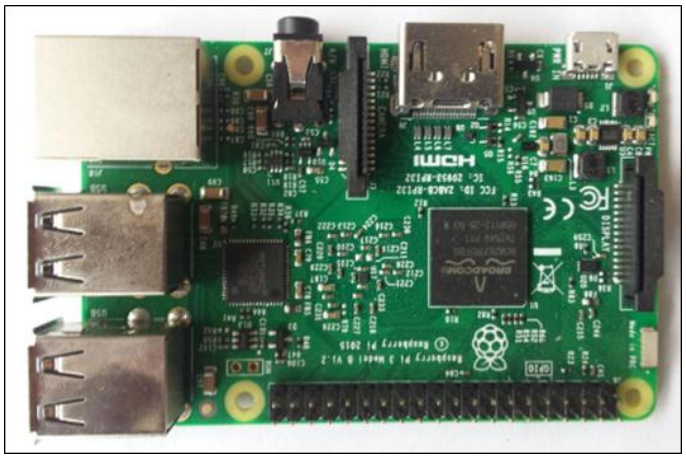

Figura 15.3 Raspberry Pi 3B
Realizado por: "CAIZA GUSTAVO", 2017

En la Tabla 1.3 se observa las principales características que tiene la tarjeta en comparación con el modelo anterior.

Tabla 1.3: Raspberry Pi 2 vs Raspberry Pi 3

	RASPBERRY PI 2	**RASPBERRY PI 3**
Procesador	Quad-Core 900MHz	Quad-Core 1.2GHz
Arquitectura	Cortex-A7 (32 bits)	Cortex-A53 (64 bits)

RAM	450 MHz	900 MHz
Periféricos	-	Wi-Fi (802.11n)
Periféricos	-	Bluetooth 4.1

Fuente: (FOUNDATION, RASPBERRY PI FOUNDATION, 2017)
Realizado por: "CAIZA GUSTAVO", 2017

3.2.2.1 *Características raspberry Pi 3B*

- 1 GB RAM
- Procesador de 1.2 GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth
- 4 puertos USB
- Puerto HDMI
- Puerto Ethernet
- Interfaz Cámara
- Interfaz de Pantalla
- 40 pines
- Tarjeta Micro SD (FOUNDATION, 2017)

3.2.2.2 *GPIO raspberry Pi 3 B*

Los pines GPIO son pines de entrada y salida de propósito general, mediante estos pines se tiene una interfaz con el mundo exterior, ya que se puede enviar y recibir datos según el requerimiento, para nuestra aplicación se utilizará para adquirir datos de sensores y controlar actuadores.

Estos pines son una interfaz física, se puede pensar que son como interruptores que pueden activar o desactivar entradas y salidas, de los 40 pines 26 son pines GPIO y los otros son pines de alimentación de 3.3v, 5v y tierra, además dos pines de ID EEPROM. (FOUNDATION, GPIO, 2017). Dentro de los pines GPIO se tiene algunos pines que a su vez pueden cumplir otras funciones por ejemplo para comunicación UART, SPI, PWM, etc.

En la Figura 16.3 se muestra la distribución de pines (GPIO) de la tarjeta Raspberry Pi 3

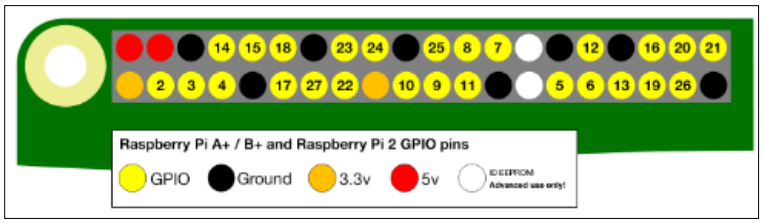

Figura 16.3 Pines GPIO
Fuente: (FOUNDATION, GPIO, 2017)

Los pines digitales de salida cuando están en HIGH tiene un voltaje de 3.3v y cuanto están en estado LOW tiene un voltaje de 0v, se debe tener en cuenta si son salidas de Pull Up o Pull Down.

Los pines digitales de entrada de igual manera se deben tener un voltaje de 3.3v para que la tarjeta lea como 1 lógico y de 0v para 0 lógico.

Si se tiene voltajes superiores a los indicados se debe recordar que los pines no tienen protección contra voltajes y corrientes, por lo cual se debe realizar el acondicionamiento de las señales de entrada y salida mediante circuitos electrónicos por ejemplo el uso de opto acopladores o relés.

3.2.2.2.1 Programación de los GPIO

Para realizar la programación de los pines digitales se puede utilizar diferentes lenguajes de programación por ejemplo: C++, Python, Java, Basic, etc. Además se debe descargar e instalar las librerías existentes para los GPIO como son: wiringPi, Sysfs, estas librerías deben ser enlazadas con el software usado para la programación según el requerimiento del usuario.

CAPITULO IV

4 CASO DE ESTUDIO

Para realizar la aplicación de control del proceso bajo norma IEC-61499 se utiliza el software 4Diac 1.7.3 en el cual se diseñan los bloques de función a utilizar para el control de sensores, actuadores y proceso. La programación se realizó en un ordenador con Windows 7 y el runtime Forte 1.7.3 que se ejecuta en el sistema operativo de la tarjeta en este caso Debian/Linux. El control se realizó desde tarjeta Raspberry Pi 3B y fue implementado en la estación de clasificación del MPS-500. Se implementó el control del brazo para el abastecimiento de piezas y luego que lleguen a la estación de clasificación se ordena por colores (negro, rojo, plateado) en diferentes rampas. En la figura 1.4 se muestra de manera general la implementación del proceso.

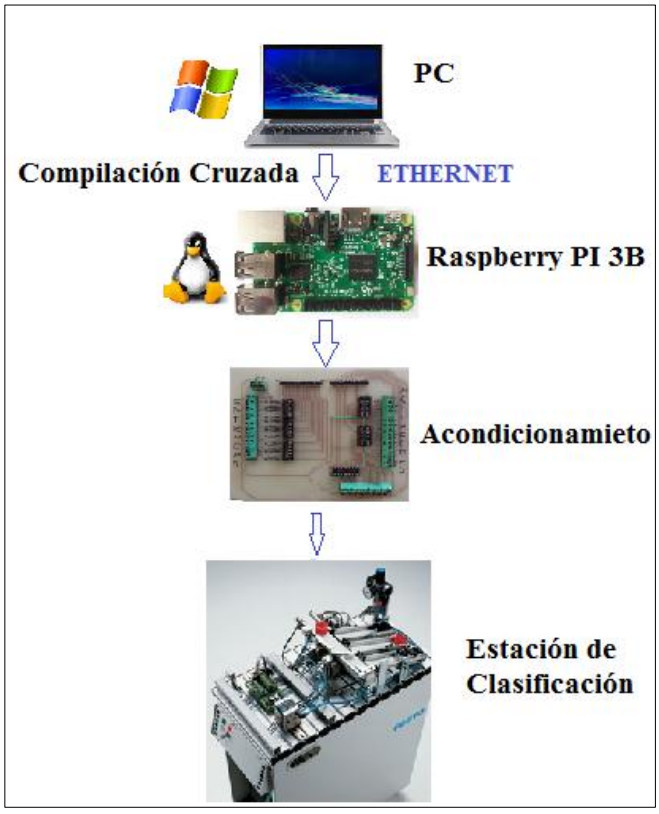

Figura 1.4: Implementación del proceso
Realizado por: "CAIZA GUSTAVO", 2017

4.1 DIAGRAMA DE FLUJO GENERAL DEL PROCESO

El diagrama de flujo general muestra de manera gráfica los pasos a seguir en el desarrollo del proceso y se muestra en la Figura 2.4.

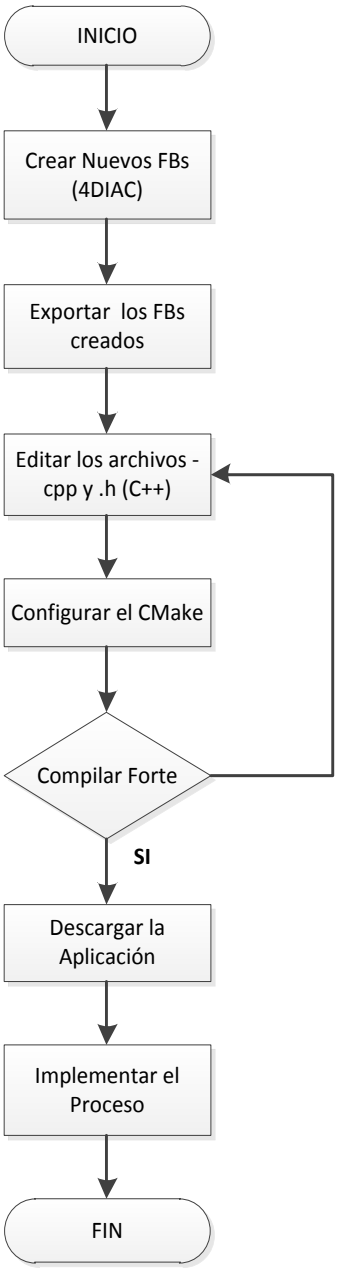

Figura 2.4: Diagrama de Flujo
Realizado por: "CAIZA GUSTAVO", 2017

4.2 DESARROLLO DE BLOQUES DE FUNCIÓN

Para desarrollar los FBs se utiliza el software 4Diac en el mismo se diseñan nuevos bloques de Función, se editan los nuevos FBs y finalmente se crea la respectiva aplicación. En la figura 3.4 se muestra el diagrama de bloques en el cual están los pasos a seguir para desarrollar los FBs.

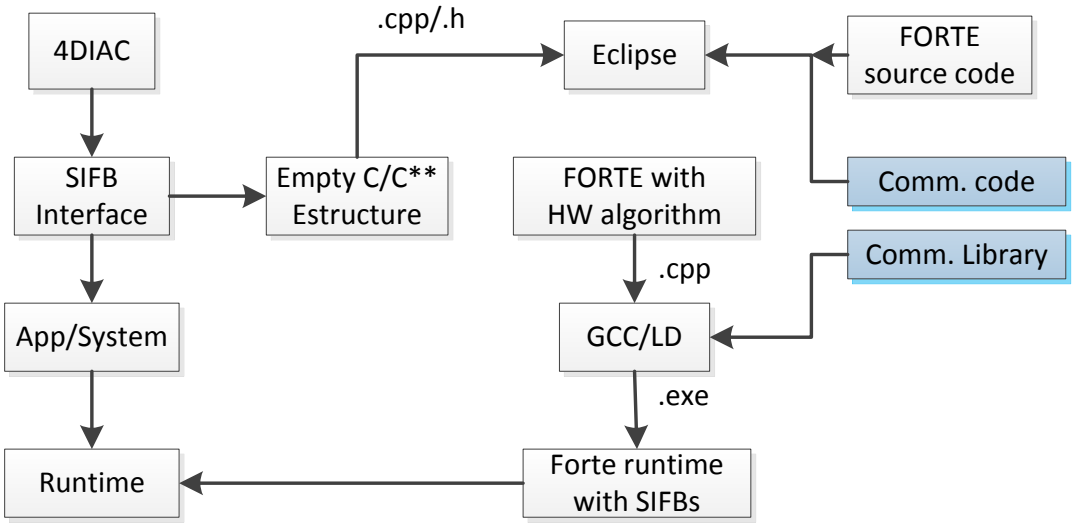

Figura 3.4: Desarrollo general 4DIAC
Realizado por: "CAIZA GUSTAVO", 2017
Fuente: (Calvo et al., 2010)

4.2.1 Diseño de los Bloques de Función

Se realiza el diseño de nuevos bloques de función para asignar las características según el requerimiento del proceso, de esta manera no se tendrá la limitación de usar los FBs que vienen en las librerías del software 4DIAC.

Primero se crea nuevos Bloques de Función para manipular las entradas y salidas digitales de la tarjeta, es decir controlar sensores y actuadores, luego en estos bloques se añade los datos y eventos a utilizar. Se debe crear las entradas y salidas de datos en el FB luego se configura el nombre y el tipo de dato de la variable a utilizar y se enlaza con el evento a ejecutar como lo muestra la Figura 4.4.

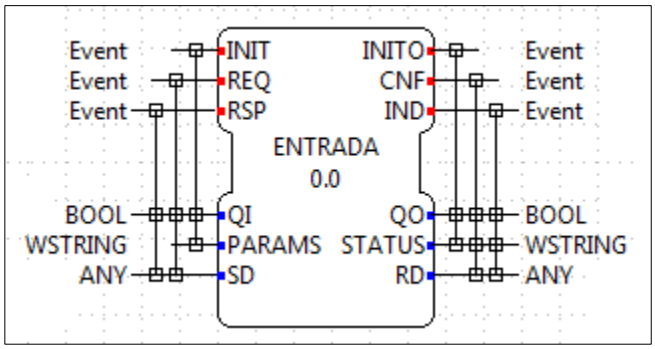

Figura 0.1: Bloque de función creado
Realizado por: "CAIZA GUSTAVO", 2017

4.2.1.1 *Bloques de función para los sensores*

Se diseñan nuevos bloques de función para tomar los datos de cada uno de los sensores de las estaciones y se asigna el número de pin a utilizar en la tarjeta raspberry Pi usando los GPIO. Se realizó el diseño nueve FBs para los sensores de la estación de manipulación y la estación de clasificación, de esta manera el usuario puede asignar cualquier pin de la tarjeta a cada sensor.

Los sensores del brazo que se van a controlar son los siguientes: sensor izquierda, sensor derecha, sensor abajo, sensor arriba, sensor pinza.

Los sensores para la clasificación son los siguientes: sensor inductivo, sensor presencia, sensor color y sensor reflectivo.

La configuración de los bloques de función para los sensores se describe a continuación:

- PIN (INT). Sirve para asignar el número de pin de manera externa al que se conecta cada sensor en la tarjeta.
- VALUE1 (BOOL). Sirve para enviar el estado del sensor al FB que controla el proceso.
- INIT (Event). Envía eventos de manera continua para que el FB envié los datos del estado del sensor cada 100ms.
- REQ (Event). Envía un solo evento al bloque de función, que está programado para que inicialice las librerías a utilizar en el algoritmo.

Dentro de la programación se encuentra que en sensor envié un TRUE si está activado y un FALSE si esta desactivado, estos datos son enviados por la salida VALUE que es de tipo BOOL.

En la Figura 5.4 se muestra un FB creado para el sensor de presencia, y se puede visualizar las entradas y salidas añadidas y el tipo de dato asignado.

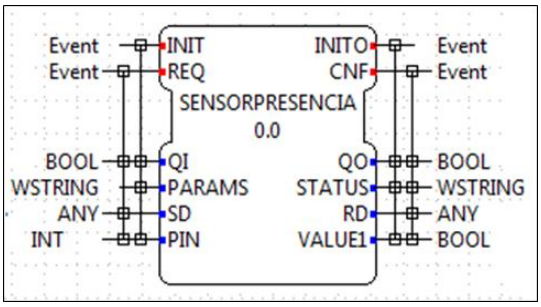

Figura 5.4: Bloque de función del sensor de presencia.
Realizado por: "CAIZA GUSTAVO", 2017

4.2.1.2 *Bloques de Función para actuadores*

Posteriormente se desarrollaron bloques de función para el control de los actuadores. Se creó un FB para cada actuador de la estación de manipulación y clasificación, por lo que se tiene 8 FBs para actuadores.

Los actuadores del brazo que se van a controlar son los siguientes: brazo izquierda, brazo derecha, baja brazo, abre pinza.

Los actuadores para la clasificación son los siguientes: motor, pistón de espera, pistón rampa uno, pistón rampa dos.

La configuración de los FBs se describe a continuación:

- PIN (INT). Sirve para asignar el número de pin al que se conecta el actuador de manera externa.
- VALUE (BOOL). Recibe el dato del bloque de función del proceso, y permite activar o desactivar el actuador seleccionado según el algoritmo programado.

Dentro de la programación se encuentra que el actuador se active según el algoritmo suministrados por el proceso, si el proceso envía un TRUE el actuador se activa y si envía un FALSE el actuador se desactiva, estos datos son adquiridos por la entrada VALUE que es de tipo BOOL.

En la Figura 6.4 se muestra un FB creado para el control del motor, en el cual se puede visualizar las entradas añadidas y el tipo de dato asignado

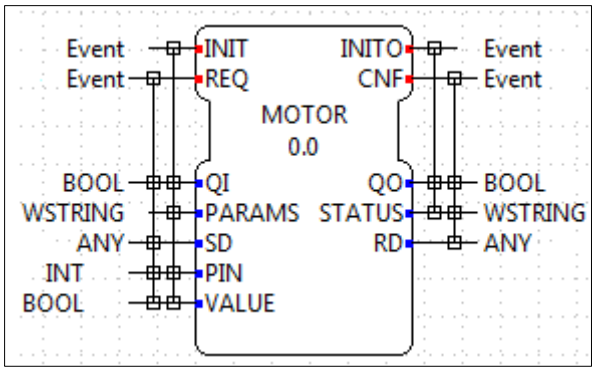

Figura 6.4: Bloque de función del actuador
Realizado por: "CAIZA GUSTAVO", 2017

4.2.1.3 *Bloque de función del brazo*

El siguiente paso fue desarrollar un FB para configurar el algoritmo que controla el brazo, y tiene entradas de tipo BOOL para el ingreso de estado de los sensores y también datos de salida tipo BOOL para el control de actuadores.

El FB creado debe tomar los estados de los 5 sensores del brazo, realizar el algoritmo programado y activar las cuatro salidas de los actuadores según el algoritmo implementado.

Los datos de los sensores ingresan a cualquiera de las variables VALUE y para los actuadores son enviados por cualquiera de las variables ACT.

La configuración del FB se describe a continuación:

- VALUE1 (BOOL). Recibe el estado del sensor asignado al pin, para luego programar el algoritmo de control.
- ACT1 (BOOL). Envía el dato para activar o desactivar el actuador asignado.

La programación está realizada para que el brazo espere una pieza, una vez que llega la pieza de cualquier color el sensor de la pinza se activa y hace que el actuador cierre la pinza y transporte la pieza hacia la estación de clasificación, una vez que llega a la estación de clasificación el brazo deja la pieza y regresa a transportar otra pieza y así continua el proceso de manera cíclica. Si el brazo no detecta la pieza durante el trayecto, regresa a buscar una pieza y no sigue en el proceso sin llevar nada de esta manera se optimiza el proceso.

En la Figura 7.4 se muestra un FB creado para el control del brazo, en el cual se visualiza las entradas, salidas y el tipo de dato asignado.

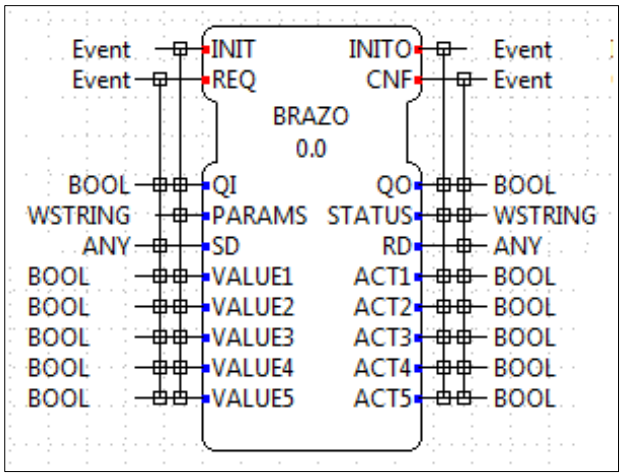

Figura 7.4: Bloque de función del brazo
Realizado por: "CAIZA GUSTAVO", 2017

4.2.1.4 *Bloque de función estación de clasificación*

Se procede a desarrollar un FB para el configurar el algoritmo que controla la estación de clasificación, y tiene entradas de tipo BOOL para el ingreso de datos de los sensores y también datos de salida tipo BOOL para el control de actuadores de la estación.

El FB creado debe tomar los estados de los cuatro sensores del brazo, realizar el algoritmo programado y activar las cuatro salidas de los actuadores según el estado de los sensores. La configuración de pines se describe a continuación.

- VALUE1 (BOOL). Recibe el estado del sensor asignado al pin, para luego programar el algoritmo de control para la estación de clasificación.
- ACT1 (BOOL). Envía el dato para activar o desactivar el actuador de la estación asignado

La programación está diseñada para seleccionar las piezas por colores, una vez que el brazo suministra la pieza se activa el sensor de presencia y este activa el motor de la banda transportadora y el pistón que detiene la pieza, posteriormente los otros sensores detectan el color de la pieza y según el caso activan los pistones de carrera corta para la distribución en las diferentes rampas, una vez que la pieza cae a la rampa se activa el sensor reflectivo con el cual se sabe que la pieza fue seleccionada según el color y detiene el proceso hasta que otra pieza llegue a la estación.

En la Figura 8.4 se muestra un FB creado para realizar la clasificación, en el cual se visualiza las entradas añadidas y el tipo de dato asignado.

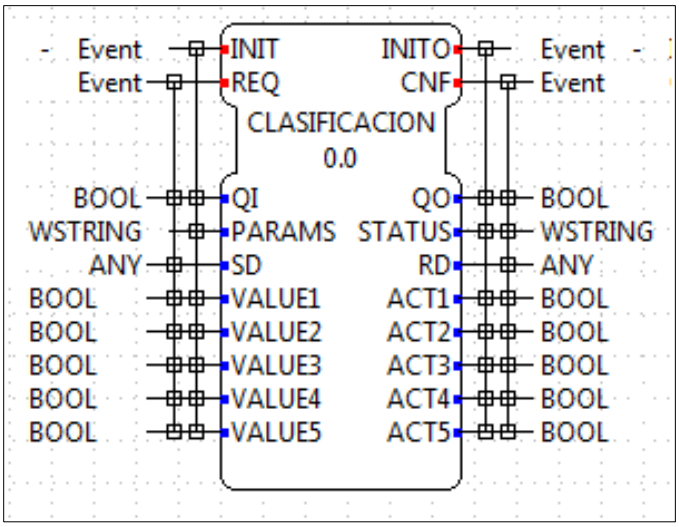

Figura 8.4 Bloque de función de la estación de clasificación
Realizado por: "CAIZA GUSTAVO", 2017

4.2.2 Exportación y Programación del Bloque de Función Creado

Una vez creado y editado el bloque de función se procede a exportar y se obtiene los archivos con extensión .h y .cpp. Estos archivos deben ser editados en cualquier programa que utilice uno de los lenguajes aceptados en la norma IEC-61499, en el proyecto se utilizara eclipse y el lenguaje C++.

En el archivo .h se añaden las librerías a utilizar, se utilizó la librería wiringPi para el control de los GPIO de la tarjeta.

En el archivo .cpp se programa el algoritmo de control para los datos de entrada y salida según la aplicación, además se debe establecer el flujo de eventos para la ejecución de los datos.

4.2.3 Ejecución del Runtime Forte

Luego de tener editados los archivos .h y .cpp de los bloques de función diseñados se guardan en una carpeta dentro de Forte en la raspberry y se genera el archivo CMakeLists.txt y se añaden: las librerías, ficheros a compilar, nombres de los bloques de función y parámetros externos que se desean ejecutar.

Se puede tener un único fichero por proyecto o si se desea se puede dividir de manera jerárquica en varias carpetas y luego mandarles a llamar de las subcarpetas mediante el comando ADD_SUBDIRECTORY.

4.2.4 Configuración del CMake

El proyecto fue diseñado para realizar la programación en una laptop y la tarjeta se utiliza para ejecutar el runtime FORTE, por ello se debe hacer una compilación cruzada. Esto se realiza por que un computador es más rápido que la raspberry Pi, de esta manera no se consume muchos recursos de la tarjeta. Para compilar Forte en la tarjeta Raspberry Pi se debe instalar el Cmake en la tarjeta y realizar la configuración del software, además se debe descargar las librerías gcc y g++, esto es necesario ya que se va a realizar compilación cruzada.

Al configurar el Cmake se señala la variable Poxis, que es para la Raspberry pi y luego se procede a seleccionar las carpetas creadas para la aplicación ya que son las que se van a ejecutar en el runtime Forte.

4.3 DISEÑO DE LA APLICACIÓN DE CONTROL

La aplicación de control se realiza en el software 4DIAC, una vez creados y programados los bloques de función se procede a unir para generar la aplicación.

Para diseñar la aplicación se configura el sistema y la aplicación en el 4DIAC para lo cual se comunica a través de una red Ethernet luego se añade un RMT_DEV y un EMB_RES el cual es para el sistema embebido, posteriormente se configura la dirección IP 169.254.168.33 que se tiene en la tarjeta como lo muestra la Figura 9.4.

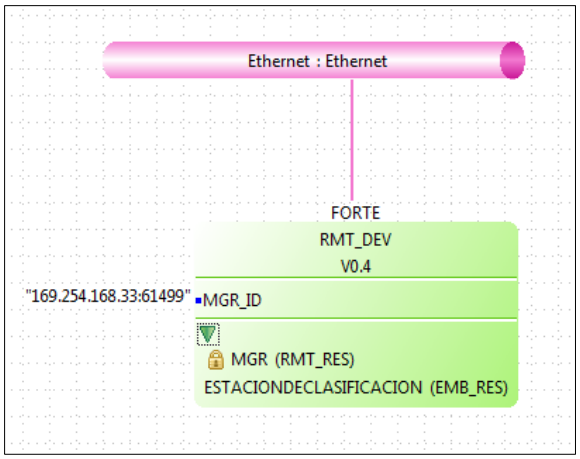

Figura 9.4: Configuración del sistema
Realizado por: "CAIZA GUSTAVO", 2017

Para el control de la aplicación final del proceso en la estación de clasificación se enlaza los FBs creados tanto para los sensores y actuadores como lo muestra la Figura 10.4.

También se debe añadir el bloque E_CYCLE el cual permite que se ejecuten los eventos creados en el bloque de función y puedan enviar los datos entre los bloques de función luego de ejecutar la acción programada.

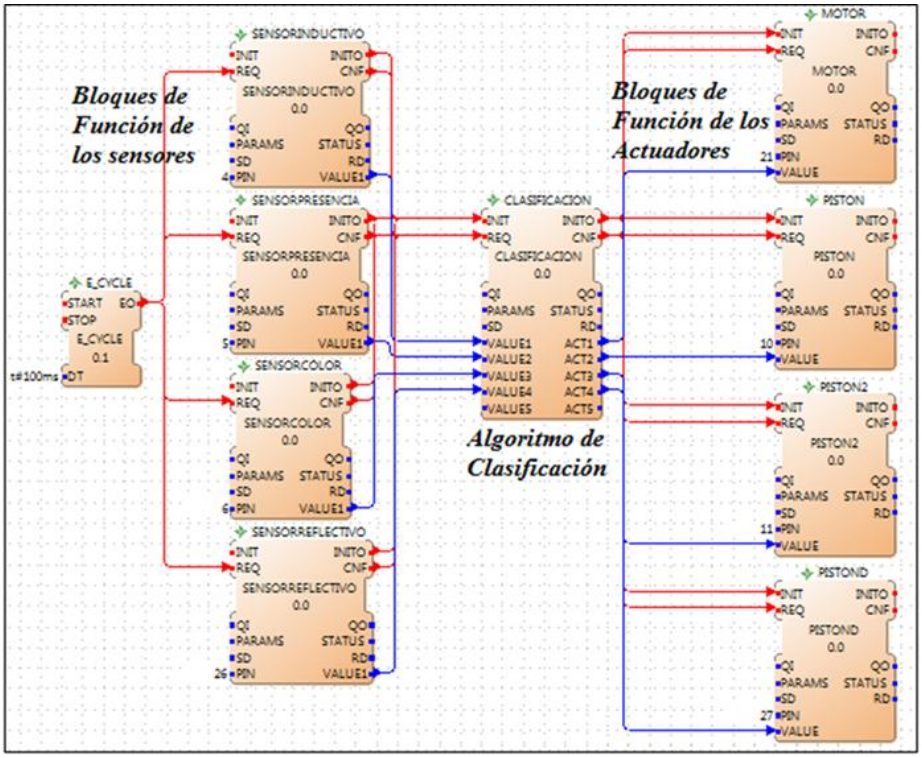

Figura 10.4: Aplicación estación de clasificación
Realizado por: "CAIZA GUSTAVO", 2017

4.4 DISEÑO DE LA ETAPA DE ACONDICIONAMIENTO

Para realizar la etapa de acondicionamiento se debe revisar los voltajes y corrientes con los que trabajan los sensores, actuadores y también en este caso los pines GPIO de la tarjeta utilizada.

4.4.1 Acondicionamiento de la señal para sensores

Se tiene que los sensores del MPS 500 trabajan con un voltaje de 24v y los pines de entrada de la raspberry Pi aceptan un voltaje de 3.3v, por lo cual se debe hacer un acondicionamiento de la señal.

Para acondicionar la señal se utilizara el driver CNY74-4 que es un opto acoplador que consta de un led infrarrojo y un fototransistor NPN de silicio. En integrado tiene cuatro canales aislados y permite eliminar problemas de ruido y acople de tierras, la distribución de pines se muestra en la Figura 11.4.

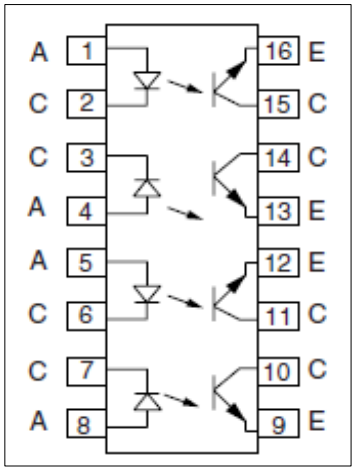

Figura 11.4: Distribución de pines CNY74-4
Fuente: (VISHAY Semiconductors, 2016)

Los sensores de las estaciones del MPS se conectan al ánodo de led con una resistencia para limitar la corriente y los cátodos se conectan a tierra, en el fototransistor se conecta al colector el voltaje de referencia que se desea obtener.

4.4.2 Acondicionamiento de la señal para actuadores

Los actuadores del MPS 500 que se deben controlar son los siguientes: cilindro simple, doble efecto, pistones y motores los cuales trabajan con un voltaje de 24v y en el caso del motor con una corriente de 1.5 A.

Los pines de salida de la tarjeta raspberry Pi tienen un voltaje de 3.3v en estado HIGH y 0v en LOW, por lo que se acondiciona la señal para que pueda activar los actuadores y se acopla las tierras para no quemar la tarjeta y evitar problemas al momento de implementar el proceso.

La señal para los actuadores se acondiciona con el driver L293D el cual es un puente H y se puede hacer el control de cuatro actuadores de manera independiente, tiene entradas mediante las cuales habilito o deshabilito las salidas para los actuadores de manera independiente, estas salidas son controladas por una señal TTL (Lógica Transistor a Transistor), además tiene un pin para alimentar los actuadores con un voltaje diferente al de alimentación del integrado, la distribución de pines se observa en la Figura 12.4.

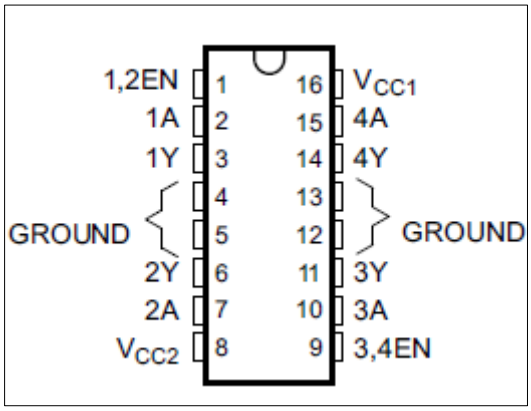

Figura 12.4: Distribución de pines L293D
Fuente: (Texas Instruments, 2016)

Los pines del circuito 1 y 9 van conectados a Vcc para habilitar la salida de los canales del driver, el pin 16 se conecta la alimentación para el driver en este caso 5v, el pin 8 se conecta la alimentación con la cual trabajan los actuadores en el caso de los actuadores del MPS 500 funcionan a 24v, los pines 3, 6, 11,14 son las salidas para los actuadores y se controlan con una señal TTL que es enviada desde la tarjeta a los pines 2, 7, 10, 15 respectivamente.

4.4.3 Diseño e implementación de la placa para acondicionamiento.

Una vez que se tiene los circuitos de acondicionamiento tanto para los sensores y actuadores se procede a realizar el diseño de la placa electrónica, para lo cual se usa el software Proteus.

El diseño se realiza para conectar los pines GPIO los cuales van a los drives para el acondicionamiento de la señal según sea el caso y luego las salidas van a las borneras en las cuales se deben conectar los sensores en las entradas y los actuadores en las salidas.

En el diseño de la placa se realiza con tierras diferentes es decir la tierra de la fuente de la raspberry Pi es diferente a la tierra de los sensores y actuadores, para de esta manera proteger contra daños de voltaje y corriente a la tarjeta. El diseño de las pistas de la placa se muestra en la Figura 13.4.

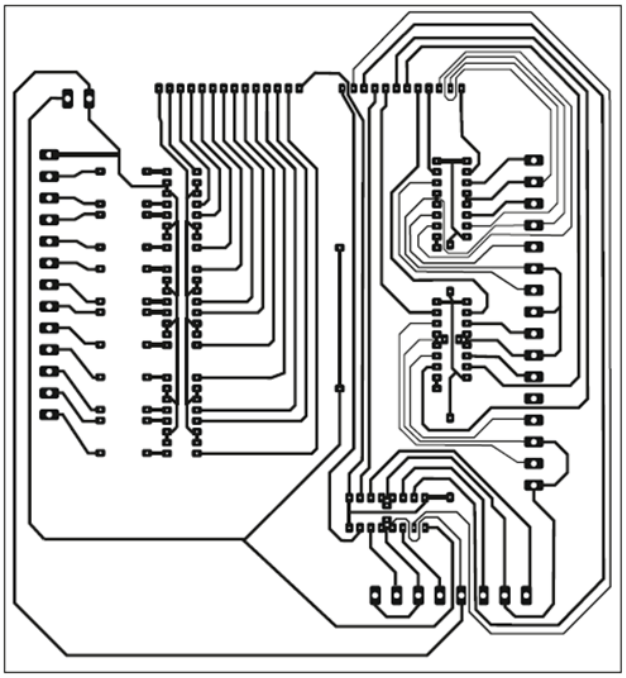

Figura 13.4: Pistas del circuito de acoplamiento
Realizado por: "CAIZA GUSTAVO", 2017

En la Figura 14.4 se muestra el diseño e implementación de la placa finalizada para el acondicionamiento de las señales.

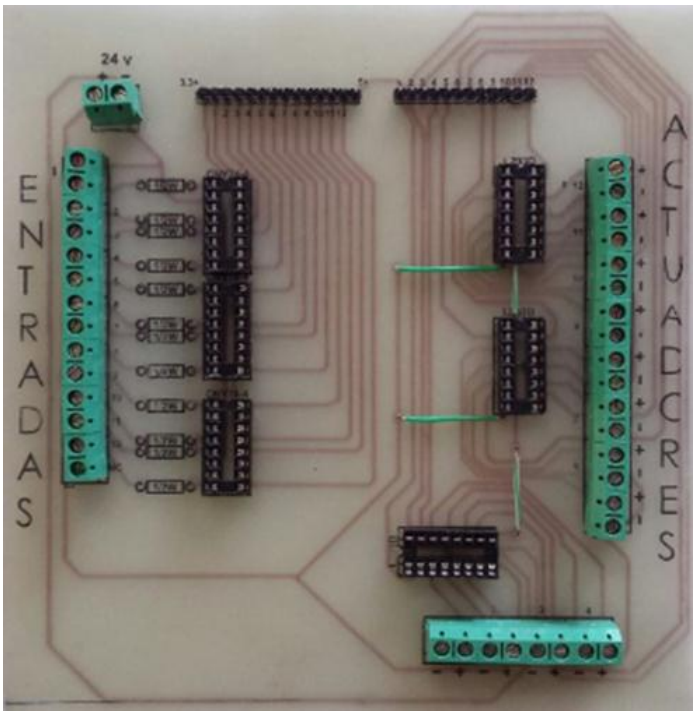

Figura 14.0.2: Circuito de acoplamiento
Realizado por: "CAIZA GUSTAVO", 2017

4.5 IMPLEMENTACIÓN DEL SISTEMA

Los resultados experimentales en este caso son la implementación y funcionamiento de la aplicación diseñada. En el proyecto se muestra en la implementación y control del sistema bajo el estándar IEC-61499, toda la programación fue realizada en el software 4DIAC el cual es propio del estándar. Se creó la aplicación para que el runtime FORTE se ejecute con los nuevos bloques creados en el 4DIAC y de esta manera lograr descargar la aplicación creada para el control del proceso y ver su funcionamiento.

4.6 CONFIGURACIÓN Y FUNCIONAMIENTO DEL SISTEMA

Luego de realizar la aplicación en el software 4DIAC y compilar FORTE en la raspberry Pi con los nuevos bloques de función diseñados se procede a configurar y descargar la aplicación en la tarjeta como lo muestra la Figura 15.4.

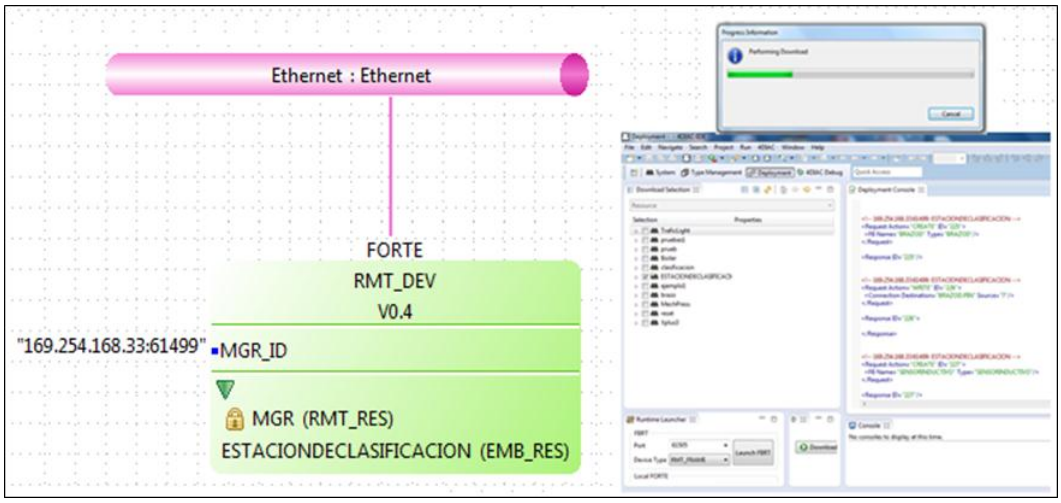

Figura 15.4 Configuración y descarga de la aplicación
Realizado por: "CAIZA GUSTAVO", 2017

Una vez creada la interfaz se procede a conectar los sensores y actuadores de las estaciones de clasificación y manipulación a la placa de acondicionamiento y luego a la raspberry Pi.

El siguiente paso es ejecutar el runtime FORTE que tiene las librerías a utilizar y los bloques de función creados. La aplicación fue creada en dos sistemas operativos diferentes como lo muestra la Figura 16.4.

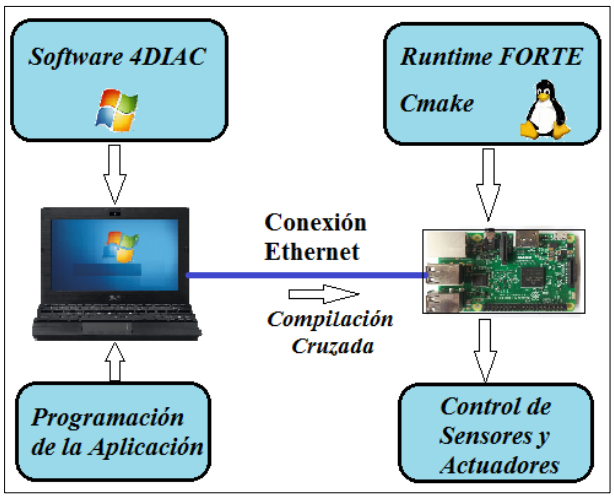

Figura 16.4: Diseño de la aplicación
Realizado por: "CAIZA GUSTAVO", 2017

Durante la ejecución del proceso se puede visualizar el estado de las variables, esto indica que el estado los sensores está cambiando y de acuerdo a la programación se activan los actuadores, estos datos se pueden observar en tiempo real gracias a la opción que ofrece el software 4DIAC.

Se puede modificar ciertos parámetros de la planta, sin necesidad de cambiar el código interno de los bloques de función. Esto hace que tengamos un sistema más flexible al momento de realizar cambios en el proceso.

En la Figura 17.4 se muestra el bloque de función que realiza el proceso del brazo, el cual cambian de estado de TRUE a FALSE dependiendo el estado de los sensores y actuadores de dicha estación.

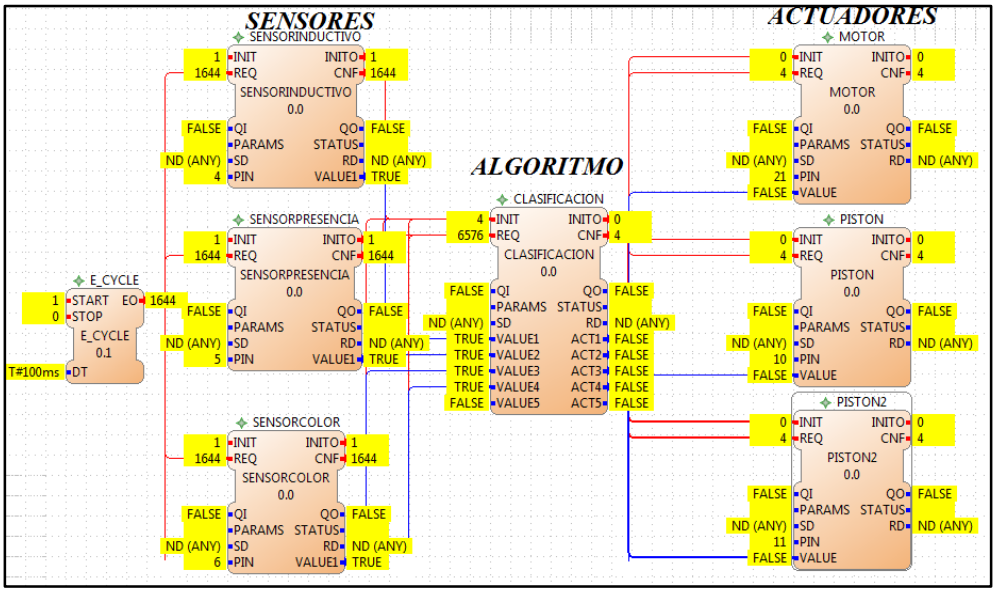

Figura 0.3.4: Visualización de las variables
Realizado por: "CAIZA GUSTAVO", 2017

Luego de realizar la implementación del sistema se observó los alcances y ventajas que se tiene al implementar el proceso bajo la norma IEC-61499 los cuales son detallados a continuación:

- Mediante el software 4DIAC se logra observar el estado de las variables del proceso en tiempo real y la ejecución de los eventos en el bloque de función.
- Se tiene un sistema reconfigurable debido a que se puede cambiar ciertos parámetros en el funcionamiento del proceso, dichos parámetros son configurados en el bloque de función sin necesidad de ingresar al código interno. Se puede modificar la asignación de pines para

sensores y actuadores dependiendo del requerimiento o la ejecución de los eventos para la activación de los bloques de función.

- Se puede crear sistemas flexibles y escalables, debido a que se puede tener componentes de distintas marcas en un mismo proceso. Además se tiene un solo lenguaje de programación para los dispositivos que soporten el estándar.

4.7 ANÁLISIS DE COSTOS

Se realiza el análisis de costos reales debido a que en la hipótesis del proyecto se dice que el sistema reducirá los costos de implementación.

En la tabla 1.4 se muestra los costos de automatización y control que se tiene actualmente en la estación de clasificación del MPS 500 al ser controlado mediante un PLC. Se realiza una comparación de precios al controlar el mismo sistema, con la tarjeta raspberry Pi 3B que es un sistema de bajo costo y utilizar el estándar IEC-61499.

Tabla 1.4: Análisis de costos

Ítem	***Automatización y Control PLC*** Costos ($)	***Automatización y Control IEC-61499*** Costos ($)
PLC S7-300	600	-
Raspberry Pi 3B	-	110
Módulo de expansión	200	-
Terminal de Entradas y Salidas	50	-
Fuente DC	150	150

Licencia De Software	646	-
Etapa de Acondicionamiento	-	90
Total $	1646,00	350,00

Realizado por: "CAIZA GUSTAVO", 2017

El análisis de costo de automatización y control bajo el estándar IEC-61499 es referenciado para el diseño de un prototipo. Los precios pueden disminuir de manera considerable si se llegara a producir en mayor cantidad para nivel industrial.

CONCLUSIONES

- Mediante la implementación del estándar IEC-61499 para el desarrollo de aplicaciones distribuidas y desarrollo de nuevos bloques de función para el control del proceso, se verifico 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 dar prioridades en el orden de ejecución de los FBs.

- Al ejecutar el runtime FORTE en la raspberry Pi y tener los otros programas en un ordenador diferente, se evita el consumo de recursos sobre la tarjeta, mejorando así el tiempo de respuesta en el proceso. Esto se logra haciendo compilación cruzada para ejecutar los archivos creados de Windows en una plataforma Linux.

- 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. Estos cambios pueden ser la ejecución del FB que se controla mediante eventos o modificar algún parámetro de la planta para recibir o enviar las señales de sensores y actuadores.

- El estándar IEC-61499 fue implementado en un sistema donde se contaba con un PLC S7-300 para su control, el cual fue desconectado y fue remplazado por una tarjeta raspberry Pi previo a una etapa de acondicionamiento de los sensores y actuadores. Por lo cual la robustez y diseño del PLC es una ventaja en este tipo de procesos.

- Al realizar la implementación de un control distribuido de bajo costo mediante la norma IEC-61499, se puede evidenciar la reducción de costos en un 78% respecto al sistema anterior que era controlado con un PLC como lo muestra la tabla 1.4.

RECOMENDACIONES

- Se debe realizar un circuito de acoplamiento para trabajar con los sensores y actuadores del MPS, ya que estos trabajan a voltajes y corrientes superiores a los que soporta la tarjeta, además se debe acoplar las tierras para no tener problemas el momento de la ejecución.

- Se recomienda añadir las librerías de los GPIO, para no tener problemas de compilación.

REFERENCIAS

CALVO, I., PÉREZ, F., ETXEBERRIA, I., & MORÁN, G. (2010). Control Communications with DDS using IEC61499 Service Interface Function Blocks, IEEE, Madrid-España, pp. 3–6. Consultado el 15 de 11 de 2016.

CATALÁN, C. (2016). Modelos y plataforma IEC61499 adaptados al control distribuido de máquinas herramientas en sistemas de fabricación ágil, Tesis Doctoral, Universidad de Zaragoza, Zaragoza - España.
Consultado el 03 de 02 de 2017.

CHRISTENSEN, J. H., STRASSER, T., VYATKIN, V., & ZOITL, A. (2012). The IEC 61499 Function Block Standard : Software Tools and Runtime Platforms, AIT Austrian Institute of Technology , Vienna AT.
Consultado el 11 de 11 de 2016.

GÁRCIA, M. V., PÉREZ, F., CALVO, I., LÓPEZ, F., & MORÁN, G. (2015). Desarrollo de CPPS sobre IEC-61499 Basado en Dispositivos de Bajo Coste. XXXVI Jornadas de Automática, pp. 230–237, Bilbao-España.
Consultado el 15 de 11 de 2016.
http://www.ehu.eus/es/web/ja2015/sistemas-de-tiempo-real

HOFFMAN, B., COLE, D., PARK, C., & VINES, J. (2010). Software Process for Rapid Development of HPC Software Using CMake, pp. 378–382.
Consultado el 15 de 02 de 2017.
https://doi.org/10.1109/HPCMP-UGC.2009.62

GARCÍA, SÁNCHEZ, G., GONZÁLEZ, (2013), Implementación de Sistemas Empotrados de Control Distribuido Bajo el Estandar IEC-61499. Trabajo de Fin de Master, Universidad del País Vasco, Bilbao-España.
Consultado el 15 de 11 de 2016.

IVANOVA, D., FREY, G., & BATCHKOVA, I. (2008). Intelligent component based batch control using IEC61499 and ANSI/ISA S88. 2008 4th International IEEE Conference Intelligent Systems, IS 2008, pp. 444–449.
Consultado el 15 de 11 de 2016.
https://doi.org/10.1109/IS.2008.4670424

LINDE, M. VAN DER, & ELEC, B. E. (2016). Using IEC61499 to Achieve Smart Grid

Automation through Interconnected Distribution Reclosers, Australasian Universities Power, Brisbane-Australia.
Consultado el 25 de 02 de 2017.

PELTOLA, J., CHRISTENSEN, J., SIERLA, S., & KOSKINEN, K. (2007). A migration path to IEC 61499 for the batch process industry. IEEE International Conference on Industrial Informatics (INDIN), pp. 811–816.
Consultado el 15 de 11 de 2016.
https://doi.org/10.1109/INDIN.2007.4384878

ROBERTO, R.SOUSA, (2010). Analyzing the Compatibility Between ISA 88 and IEC 61499 Mário de Sousa, Porto - Portugal.
Consultado el 14 de 12 de 2016.

SAHITYA, S., LOKESHA, H., & SUDHA, L. K. (2016). Real Time Application of Raspberry Pi in Compression of Images, pp. 1047–1050, India.
Consultado el 10 de 01 de 2017.

SIERLA, S., CHRISTENSEN, J., KOSKINEN, K., & PELTOLA, J. (2007). Educational approaches for the industrial acceptance of IEC 61499. IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, pp. 482–489, Finland. Consultado el 04 de 10 de 2016 .
https://doi.org/10.1109/EFTA.2007.4416807

THRAMBOULIDIS, K., SIERLA, S., PAPAKONSTANTINOU, N., & KOSKINEN, K. (2007). An IEC 61499 Based Approach for Distributed Batch Process Control. … Informatics, 2007 5th, pp. 177–182, Grecia.
Consultado el 04 de 10 de 2016. .
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4384752

TREJO-HERNÁNDEZ, M., & LÓPEZ-MELLADO, E. (2013). Specification of Manufacturing Systems Controllers Using the Standard IEC61499, IEEE Int. Conference, 179–184, Zapopan - Mexico.
Consultado el 12 de 12 de 2016.

ZOITL, A., STRASSER, T., & EBENHOFER, G. (2013). Developing Modular Reusable IEC 61499 Control Applications with 4DIAC,Conference on Industrial Informatics, 358–363. Consultado el 10 de 11 de 2016

ECLIPSE. 4DIAC.

Consultado el 26 de 02 de 2017.

https://eclipse.org/4diac/en_ide.php

FESTO. Estacion de Clasificación:Final.
Consultado el 27 de 02 de 2017.
http://www.festo-didactic.com/es-es/productos/mps-sistema-de-produccion-modular/estaciones/estacion-de-clasificacionn final.htm?fbid

FESTO. MPS 500-FMS.
Consultado el 17 de 02 de 2017.
http://www.festo-didactic.com/es-es/productos/fabrica-didactica,cim-fms-sistemas/mps-500-fms/mps-500-fms-produccion-flexible-compatible,modular-y-versatil.htm?fbid=ZXMuZXMuNTQ3LjE0LjE4LjYzOS40MjYx

FESTO. MPS 500-FMS.
Consultado el 27 de 02 de 2017.
Obtenido de http://www.festo-didactic.com/es-es/productos/mps-sistema-de-produccion-modular/mps-500-fms/mps-500-fms-produccion-flexible-compatible,modular-y-versatil.htm?fbid=ZXMuZXMuNTQ3LjE0LjE4Ljg1NS40MjYx

FOUNDATION, R. P. GPIO.
Consultado el 26 de 02 de 2017.
https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/

FOUNDATION, R. P.. RASPBERRY PI FOUNDATION. RASPBERRY PI 3 MODEL B
Consultado el 23 de 02 de 2017
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

KITWARE. CMake.
Consultado el 25 de 02 de 2017
https://cmake.org/

ANEXOS

ARTICULO CIENTIFICO