INSTITUTO TECNOLÓGICO DE CHIHUAHUA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“SISTEMA FLEXIBLE DE CONTROL BASADO EN CONTROLADOR LÓGICO PROGRAMABLE”

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRA EN INGENIERÍA MECATRÓNICA

PRESENTA:

ING. SILVIA ALEXANDRA HERMOSILLO RODRÍGUEZ

DIRECTOR DE LA TESIS:
DR. JOSÉ EDUARDO ACOSTA CANO DE LOS RÍOS

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

CHIHUAHUA, CHIH. MÉXICO, ENERO DE 2019

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MEXICO

Instituto Tecnológico de Chihuahua

Chihuahua, Chih. 08/enero/2019
OFICIO No. DEPI-001/2019

M.F. LUIS CARDONA CHACÓN
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

Por medio de la presente notificamos a usted que en cumplimiento de los requerimientos para la obtención de grado de Maestra, el documento de tesis de la **C. SILVIA ALEXANDRA HERMOSILLO RODRÍGUEZ**, ha sido aprobado y aceptado para su impresión. El título de la tesis es:

"SISTEMA FLEXIBLE DE CONTROL BASADO EN CONTROLADOR LÓGICO PROGRAMABLE"

Por lo que proponemos, le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

A T E N T A M E N T E
Excelencia en Educación Tecnológica.

DR. JOSÉ EDUARDO ACOSTA CANO DE LOS RÍOS
MIEMBRO DEL JURADO DE EXAMEN

DR. PEDRO RAFAEL ACOSTA CANO DE LOS RÍOS
MIEMBRO DEL JURADO DE EXAMEN

M.C. PEDRO RAFAEL MÁRQUEZ GUTIÉRREZ
MIEMBRO DEL JURADO DE EXAMEN

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE CHIHUAHUA

Isidro Robledo Vega
DR. ISIDRO ROBLEDO VEGA.
MIEMBRO DEL JURADO DE EXAMEN

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MEXICO

Instituto Tecnológico de Chihuahua

Chihuahua, Chih. 08/enero/2019

OFICIO No. DEPI-002/2019

C. SILVIA ALEXANDRA HERMOSILLO RODRÍGUEZ
PRESENTE

Por este conducto le comunico que, a propuesta del Jurado de Examen, la División de Estudios Posgrado e Investigación le ha concedido la autorización para la impresión de tesis para obtener el grado de Maestra, cuyo título es:

"SISTEMA FLEXIBLE DE CONTROL BASADO EN CONTROLADOR LÓGICO PROGRAMABLE"

Con el siguiente contenido de capítulos:

I. Introducción.
II. Marco teórico.
III. Planteamiento del problema.
IV. Desarrollo de la solución propuesta.
V. Validación y resultados de la solución propuesta.
VI. Conclusiones.

A T E N T A M E N T E
Excelencia en Educación Tecnológica.

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE CHIHUAHUA

M. EN F. LUIS CARDONA CHACÓN
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

LCC/reba.*

Chihuahua, Chih., a 17 de enero de 2019.

Dr. Enrique Cabrero Mendoza.
Director de CONACYT.

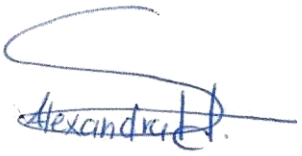
At´n. **Luis Gill Cisneros**
Dirección de Formación de Científicos y Tecnólogos.

P r e s e n t e.

Por este conducto aprovecho la ocasión para saludarlo e informarle que a la fecha he obtenido el Grado de Maestría en Ingeniería Mecatrónica en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua. Motivo por el cual agradezco todo el apoyo brindado por esta Institución que Usted representa, el otorgamiento de esta beca-crédito permitió dedicarme de tiempo completo a la realización de mis estudios de Posgrado y de esta manera lograr el cumplimiento del objetivo principal del convenio establecido.

Sin otro particular por el momento, me es grato quedar de Usted como su seguro servidor, no sin antes reiterar mi agradecimiento. Muchas Gracias!

A t e n t a m e n t e



SILVIA ALEXANDRA HERMOSILLO RODRÍGUEZ
EX BECARIA CONACYT No. 764358

c.c.p M.F. Luis Cardona Chacón
Jefe de la división de Posgrado e Investigación

RESUMEN

"SISTEMA FLEXIBLE DE CONTROL BASADO EN CONTROLADOR LÓGICO PROGRAMABLE"

Ing, Silvia Alexandra Hermosillo Rodríguez
Maestro en Ingeniería Mecatrónica
División de Estudios de Posgrado e Investigación del
Instituto Tecnológico de Chihuahua
Chihuahua, Chih. 2019
Director de Tesis: Doctor José Eduardo Acosta Cano de los Ríos

Actualmente en la industria el PLC ha sido un elemento clave en las células robotizadas, la reutilización de código entre las diferentes células es una característica deseable con la cual se obtendría una flexibilidad considerable en el entorno de programación, este tema es escasamente abordado en la literatura.

Para abordar esta problemática se propone el acoplamiento débil en la integración de sensores/actuadores y el concepto de operación dirigida por modelo, así como el modelo iMRP con el cual se propone determinar los requerimientos del modelo y posteriormente crear instancias, así como la aplicación del esquema de referencia ArquiTAM, en el cual se plantea tres niveles de abstracción, Arquitectura de referencia, Modelo de referencia y Aplicación particular, con ello se propone buscar las soluciones que conduzcan a la reusabilidad de código en PLC con un sistema hibrido PC-PLC.

A esta problemática se alcanzó la solución en donde es posible la generación de código para PLC por medio de la instanciación de clases y la implementación del concepto operación dirigido por modelo en donde por medio del modelo generado por iMRP, es posible abstraer las particularidades de él y generar las instanciaciones correspondientes para posteriormente ser cargadas al PLC correspondiente.

Dado que el área de programación de sistemas basados en PLC no ha alcanzado un nivel de madurez considerable para el desarrollo de herramientas, por lo tanto es necesario apoyarse mediante una computadora personal en donde en nivel de madurez de los lenguajes de programación es más alto.

Con forme a lo aplicado en esta problemática se puede concluir que es factible la aplicación de las técnicas en la implementación en sistemas basados en PLC.

AGRADECIMIENTOS

Quiero agradecer primero a Dios por darme la fuerza de continuar cuando he estado a punto de caer, por todas las personas que puso en mi camino y las que quitó de él.

Quiero dedicar este logro a mis padres Ing. Eduardo Hermosillo e Ing. Silvia Rodríguez, por su apoyo incondicional, por su amor, y porque siempre han creído en mí. Porque en los momentos difíciles siempre fueron mi pilar y me impulsaron a ser mejor cada día.

Por cada una de sus enseñanzas soy la persona que soy en la actualidad, muchos de mis logros se los debo a ustedes.

Gracias por guiarme para ser una mejor hija, hermana, esposa, amiga, compañera y futura profesionista, pues con su ejemplo de ética y valores existe en mí un sendero muy marcado el cual siempre seguiré.

Gracias por enseñarme con su ejemplo que el bienestar de una persona, no está por encima del de otra, la fidelidad a la ética es un tatuaje que se debe de llevar siempre y nunca debe ser borrado.

Quiero dedicarlo también a mis hermanas, Patricia y Belem Hermosillo, más que mis hermanas son mis confidentes y mis mejores amigas, siempre han estado allí cuando las necesito y espero siempre estar allí cuando me necesiten.

A mi esposo, que ha estado con migo en esta etapa, en la cual me dio fortaleza para seguir, animo cuando más lejos veía la meta, asi como su paciencia para enfrentar todos esos días difíciles, cuando el desvelo y poca amabilidad estaban presentes, gracias por estar allí cuando lo necesite.

A mi director de tesis, Dr. Jose E. Acosta cano de los ríos, por hacer posible este logro, por su paciencia en los momentos de poco entendimiento y confusión, asi como por su dedicación al

proyecto, por las enseñanzas que recibí a lo largo de estos años, gracias por su apoyo, quiero decirle que es uno de los mejores profesores que he tenido a lo largo de mi formación profesional gracias por toda esa dedicación y carisma la cual hace que las clases sean demasiado cortas y se siga queriendo mas, gracias.

A mis profesores Dr. Pedro Acosta, Dr. Isidro Robledo, M.C. Pedro Márquez, cDr. Rogelio Baray, que a lo largo de esta maestría me forjaron y me dieron sus enseñanzas, sus experiencias, conocimientos, y sembraron en mí la semilla del conocimiento, del saber, por tal motivo mi eterno agradecimiento.

A mis compañeros, que a lo largo de este gran viaje estuvieron conmigo en cada momento, en los cuales algunos fueron buenos y otros no tanto, gracias Gabriel, Andres, Mariana, Humberto, Benjamin, Mariana.

TABLA DE CONTENIDO

LISTA DE FIGURAS

CAPITULO 1

INTRODUCCIÓN

En la actualidad la programación del PLC está enfocada únicamente en el lenguaje de bajo nivel esto conlleva a que la programación sea distinta para cada entorno, así como la realización de cambios y/o anexos sea propiamente determinada por el programador, lo que conlleva a apoyarse mediante una PC la cual tiene una mayor capacidad de procesamiento y en la cual se pueden realizar enfoques diferentes que no se pueden realizar directamente en el PLC.

Uno de los elementos clave en el control de una célula robotizada ha sido el PLC; la reutilización del código de PLC entre diferentes células es una característica deseable que facilitaría el desarrollo y actualización de estaciones de producción automatizadas, sin embargo, la reutilización del código en PLC es un problema escasamente abordado en la literatura.

En el presente proyecto de tesis se propone analizar el problema y realizar un planteamiento que coadyuve al desarrollo de sistemas de control flexibles, basados en PLC. Para tal efecto se tomarán en cuenta varios aspectos tales como el acoplamiento débil en la integración practica de los sensores/actuadores y el concepto de operación dirigida por modelo, buscando con ello soluciones que conduzcan a la reusabilidad de código, de tal manera que de un sistema hibrido PLC-PC, se logre agregar/sustituir/eliminar el conjunto de equipos reutilizando el código del PLC.

Uno de los problemas más recurrentes dentro de la automatización industrial, tiene que ver con el cambio constante de elementos en las industrias, dado que estos se quedan obsoletos por el creciente mercado, esto conlleva a la necesidad de rehacer el código que se encontraba previamente en funcionamiento, asi como cambiar y/o modificar sensores y actuadores dependiendo de las especificaciones de este nuevo elemento. Lo cual conlleva a la necesidad de tener una flexibilidad en este entorno. El sistema hibrido PC-PLC permite dicha flexibilidad al poder generar código ya sea para una adición o reestructuración del sistema actual, también otorga una flexibilidad en la interconexión entre los equipos actuales y los

existentes esto permitirá más vida útil de los equipos y por consecuente un mejor manejo del tiempo y un menor esfuerzo de control.

El presente documento está organizado de la siguiente manera: En el capítulo II se encuentra el marco teórico, en el cual se encuentran los fundamentos de los puntos a tratar posteriormente en el desarrollo de la tesis, en el cual se expone la historia del PLC asi como la historia del intercambio de piezas y el reusó de ellas, también se muestra el desarrollo en el tiempo del concepto flexibilidad y sus definiciones.

En el capítulo III se encuentra el planteamiento del problema, en el cual se manifiesta la problemática a tratar para el desarrollo de la tesis asi como los objetivos generales y específicos y las técnicas para la solución propuesta de ellos.

En el capítulo IV se muestra el desarrollo de una solución propuesta en el cual se explican más a detalle los conceptos a utilizar como acoplamiento débil, operación dirigida por modelo, modelo iMRP, ArquiTAM entre otros, asi como en qué momento estos son aplicados, asi como arquitecturas y modelos específicos para generar una solución a la problemática propuesta.

En el capítulo V se encuentra la validación de los resultados derivados del desarrollo de la solución en donde se muestra un posible ejemplo de utilización y con ello su desarrollo y la ejecución del mismo. Demostrando la flexibilidad del sistema asi como el comportamiento del mismo.

Por ultimo en el capítulo VI se plantean las conclusiones del trabajo de investigación asi como el alcance obtenido y los objetivos que se alcanzaron, también se manifiesta las posibilidades de trabajos futuros y su factibilidad en la implementación en la industria actual.

CAPITULO 2
MARCO TEÓRICO

2.1 ANTECEDENTES

En las décadas de los 70’s y 80’s la flexibilidad se convierte en una característica primordial en los sistemas de manufactura, de donde surge el concepto FMS (sistema flexible de manufactura), haciendo posible responder a la gran variedad de productos, en donde el rediseño o cambios en la distribución de piso no es factible por el alto costo de reconfiguración del sistema FMS, es decir, las maquinas programables permiten modificar las características del proceso sin tener la necesidad de cambios en la estructura, donde la limitación está ligada a la estructura física de la misma. El concepto FMS presenta serias limitaciones en cuanto a flexibilidad pues se requiere modificaciones considerables en hardware para aplicarlo a una familia de productos diferente, por este motivo tiene una baja aceptación en la industria.

El Sistema Reconfigurable de Manufactura (RMS), se presenta como solución para las carencias del FMS pues este nuevo paradigma tiene como meta principal la capacidad de agregar/eliminar equipo al sistema, para obtener la funcionabilidad y capacidad requeridas exactamente cuando estas sean requeridas (Koren, 2006). De esta manera los sistemas deben de ser diseñados con base en módulos de hardware y software los cuales puedan ser integrados de forma rápida cuando sean necesarios. Las características de un sistema reconfigurable son: Modularidad, Capacidad de integración, capacidad de personalización, escalabilidad, convertibilidad y capacidad de diagnóstico (Koren et al., 1999).

El término acuñado como sistema reconfigurable, resulta estar fuertemente asociado a sistema flexible, es decir, el futuro de un sistema reconfigurable es un sistema flexible (Acosta, J., 2016). Existen diferentes definiciones de flexibilidad, sin existir una ampliamente aceptada y difundida, sin embargo para efectos de flexibilidad del sistema de control y su integración con el equipo de producción en (Acosta, J., 2016) se plantean::

> Flexibilidad de producto. Tipo de flexibilidad para el manejo de diferentes productos (mezcla de productos y agregar nuevos productos al sistema), así como variación del volumen de producción (dentro de la capacidad de producción instalada, (ElMaraghy, 2006), sin modificar el conjunto de máquinas de productos disponibles.

Flexibilidad de Maquina. Flexibilidad del sistema respecto a los elementos mecánicos del equipo y sus utillajes. Flexibilidad que puede ser transparente al sistema de control

Flexibilidad en manejo de materiales. Flexibilidad que corresponde a la cantidad de caminos o trayectorias utilizados entre las maquinas en relación con la cantidad de todos los caminos posibles.

Flexibilidad de ruta. Habilidad para procesar un conjunto dado de piezas en maquinal alternativas.

Flexibilidad de operación. Planes de procesamiento diferentes disponibles para fabricación de piezas.

Flexibilidad de proceso. Conjunto de tipos de piezas que pueden ser producidas sin cambios mayores en la preparación de la máquina.
Flexibilidad en el conjunto de máquinas. Habilidad para modificar el conjunto de máquinas:

> Flexibilidad de expansión/reducción. Eliminar o agregar máquinas de tipo ya que existen en el sistema de producción.
>
> Flexibilidad para introducción de máquinas. Agregar máquinas de tipo no existe en el sistema (generalmente nueva tecnología).

Flexibilidad en la lógica funcional. Habilidad para realizar cambios en la lógica de operación del sistema:

> Modificación de lógica funcional. Modificar funciones lógicas del sistema, tal como algoritmos de producción / lanzamiento de tareas, monitoreo de un parámetro de desempeño.

Flexibilidad para introducción de lógica funcional. Capacidad de introducir nuevas funciones.

El desarrollo de sistemas personalizados (especializados para un conjunto previamente definido de equipos de producción) implica un esfuerzo considerable que resulta ser lento, además de resultar en aplicaciones inflexibles (Andersson, 1997; Ferrolho y Crisstomo, 2006). Esto justifica el desarrollo de elementos reusables para las diferentes etapas. La reusabilidad en el área de computación se encuentra en aspectos tales como la reusabilidad de CPU, rutinas para atención a periféricos, funciones genéricas e interfaces de usuario (Van der Aalst, 1999), en la reusabilidad de diseño se encuentra el concepto de patrones y el reusó de modelos para varias plataformas de computación, en la reusabilidad de elementos implica la reusabilidad de elementos abstraídos a nivel conceptual como requerimientos y soluciones propias de domino. Un problema abierto es la reusabilidad del sistema de control a integrar en diferentes conjuntos de equipos de producción.

El trabajo de acoplamiento o integración de sistemas automáticos de producción ha sido abordado desde diferentes perspectivas en la literatura. El acoplamiento es el agrupamiento de dos sistemas, piezas o dispositivos, de tal manera que su operación combinada produce los resultados deseables (RAE, 2014), una gran parte del esfuerzo por satisfacer los requerimientos impuestos por el medio ambiente de operación de objetos distribuidos, surge un problema cuando el sistema requiere manejar más de un tipo de tecnología de software como MSMW y Java, esto provoca un esfuerzo considerable (Pautasso y Wilde, 2009; Su, 2007; Ye y Qin, 2003), las soluciones orientadas a servicios web presentan una mejor opción. Existen iniciativas para la integración especifica de PLC con el sistema distribuido con base en bloques funcionales (Dai y Byatkin, 2012; Vyatkin, 2009), como es la arquitectura del estándar IEC 61499 (IEC, 2005) aplicada en la comunidad de sistemas de producción por lotes que plantea características esenciales en particular a nivel de dispositivo de campo (sensores/actuadores) (Thramboulidis, 2005a)

En el Instituto se ha trabajado en diferentes tesis de maestría y proyectos de investigación sobre el problema de integración flexible entre equipo de producción y sistema informático, obteniendo como resultado un esquema de referencia para la integración con base en el concepto de acoplamiento débil, (Acosta, J. E., 2015; Acosta C. J., Sastrón B. F., 2013; Aguayo, L. B., 2014; Carreón, E. V., 2015; Lara, O. I. 2015; Loya, O. A., 2016; Orona, F. M., 2015).

2.2 REVISIÓN DE LA LITERATURA

Hoy en día hay más de 800,000 unidades de robots por todo el mundo, según datos de IFR (international Federation of Robotics), se pretende que en este año 2017 la producción aumente un 7% de los años anteriores, al día de hoy corea del sur es quien cuenta con más robots instalados por cada 10,000 empleados con 531, Singapur con 398, México ocupa el puesto 30 con 33 robots por cada 10,000 empleados, argentina con 16 y Brasil con 11, como se muestra en la figura 2.1

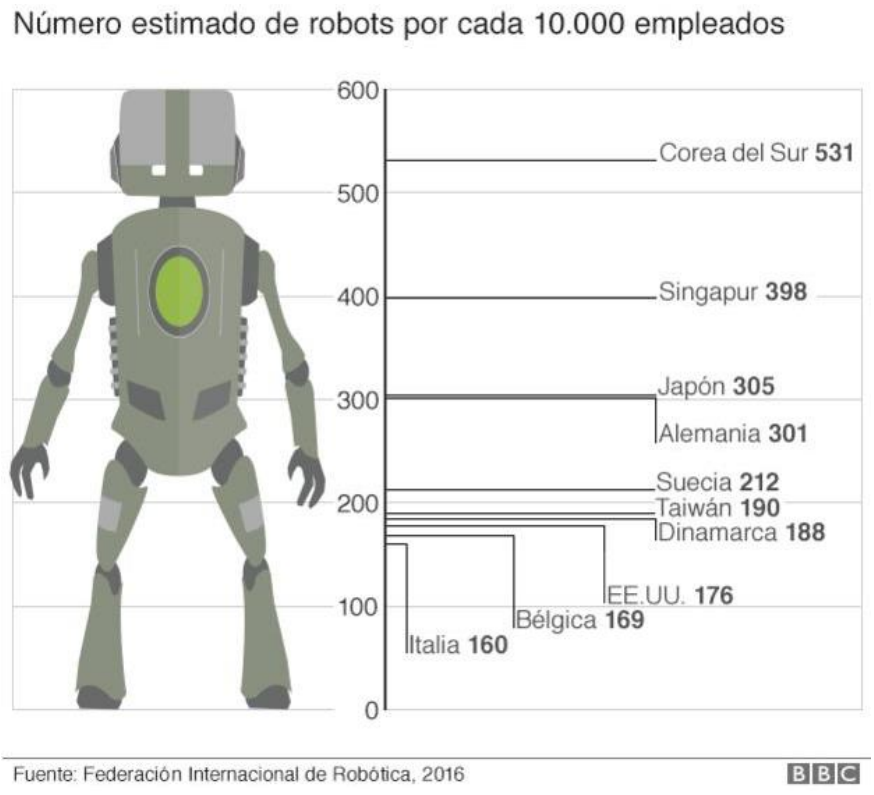


Figura 2.1 Los 10 Países más robotizados

En el año 1979 se introdujo la primera definición de robot industrial aunque no la única las más destacadas son:

"Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover piezas, materias, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas"

Definición de Robot Instituto of América (RIA-1979)

"Un robot industrial es un manipulador de 3 o más ejes, con control automático, reprogramable, multiplicación, móvil o no destinado ser utilizado en aplicaciones de automatización industria. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia)"

Definición de la asociación internacional de estándares (ISO-8373-1998)

2.2.1 Robot Industrial

Un robot industrial es aquel que su objetivo principal es desarrollar alguna función en el ámbito industrial este se conforma por diferentes elementos los cuales son:

Manipulador. Es el elemento robótico o brazo robotizado

Brazo. Es desde la base hasta la muñeca ofrece movilidad al conjunto

Muñeca. Desde el final del brazo hasta el elemento terminal

Elemento terminal. Herramienta con la que se realiza la tarea.

Controlador. Regula cada movimiento del manipulador y procesa información

Dispositivos de entrada y salida de datos. Caja de comandos, teach pendant, teclado, monitor

Dispositivos especiales. Dispositivos que facilitan la tarea del mismo robot manipulador así como adaptaciones de visión artificial

Actuadores. El objetivo es proporcionar movimientos al manipulador atravesó de las articulaciones.

Sensores. Tienen como principal objetivo medir el estado del manipulador o entorno.

Sistema de control. Realiza la supervisión y el control del movimiento del manipulador así como de su entorno.

2.2.2 Grados de libertad.

Se definen como la suma de cada uno de los movimientos independientes de este, así como que puede realizar cada articulación generalmente se refieren a las articulaciones como grados de libertad, como se muestra en la figura 2.2

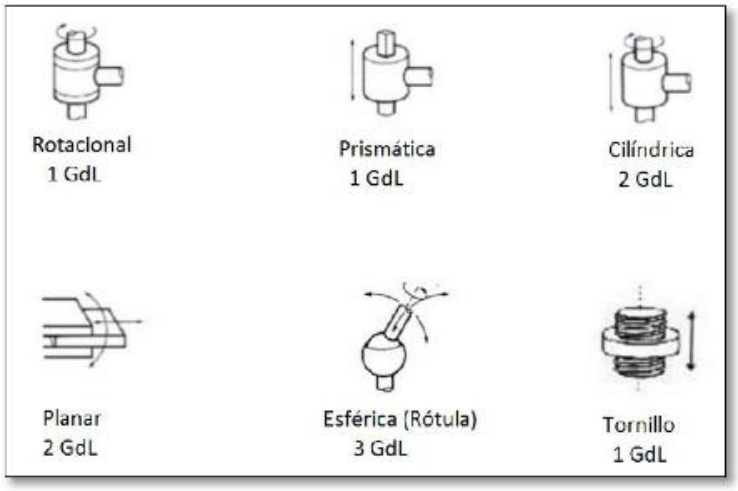


Figura 2.2 Tipos de movimientos y grados de libertad

Los ejes de un robot se nombran mediante letras las cuales están dadas por nomenclatura de abreviaciones en inglés, esto puede variar según el fabricante (figura 2.3).

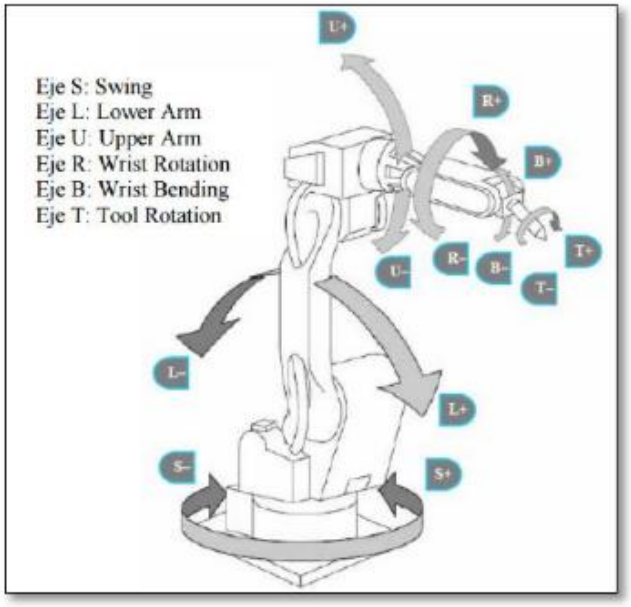


Figura 2.3 Ejes y sus respectivos nombres

2.2.3 Capacidad de carga.

La capacidad de carga de un robot manipulador en su punto P o herramienta depende de la dimensión del robot, así como si es de configuración cartesiana o cilíndrica, también del material de construcción, del tipo de acatadores y de la transición de movimiento un robot de soldadura convencional puede llegar a soportar en su punto P desde 3 Kg hasta 20Kg.

2.2.4 Equipamiento del Robot.

Este equipamiento está formado por distintos elementos que son necesarios para realizar la soldadura con el robot, compuesto por una antorcha de soldadura, un sistema de anticolisión, un paquete energético, una devanadora de hilo, una sirga para guiar el hilo, y un "bobin-holder" para sostener la bobina de hilo o algún otro sistema similar. Además de este equipamiento básico, también se emplean algunos equipos opcionales y de uso común, para realizar la limpieza de la boquilla de la antorcha. Un sistema de "auto-calibración" de la antorcha es también un sistema utilizado. (Coca García J, P. 29) (Figura 2.4).

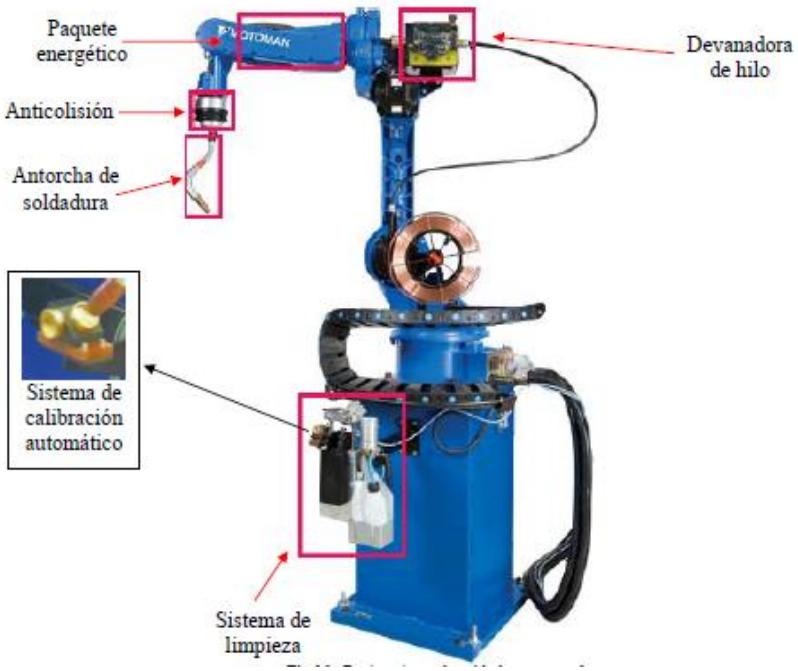


Figura 2.4 Partes de un robot motoman de soldadura (Coca García J, P. 29).

2.2.5 Mantenimiento del Robot.

"El robot requiere únicamente un mantenimiento mínimo durante su funcionamiento. Se ha diseñado para permitir el servicio técnico más sencillo posible:

- Se utilizan motores de CA sin mantenimiento.
- Se usa aceite como lubricante de las cajas reductoras.
- El encaminamiento de los cables se ha optimizado para conseguir la máxima longevidad. Además, en el caso poco probable de una avería, su diseño modular permite sustituirlos fácilmente.

Se requieren las siguientes operaciones de mantenimiento:

- Una vez al año, cambiar el filtro de la refrigeración de la unidad de accionamiento.
- Cada tres años, sustituir las baterías.
- Cambiar el aceite de la muñeca después del primer año y cada cinco años a partir de entonces"

(Álvarez Paramino J, P.116), estas son algunas de las técnicas de mantenimiento propuestas por este autor.

2.2.6 Soldadura

2.2.6.1 Maquina de Soldadura.

Esta es la fuente de potencia, comúnmente conocida como máquina de soldadura, esta se encarga de generar el diferencial de potencial con la corriente necesaria para situarlo en el electrodo y posteriormente a la pieza, así como realizar la acción de soldar, también es la encargada de controlar cada parámetro necesario para realizar una soldadura correcta y eficaz, que puede ser velocidad, tipo de soldadura, tipo de gas, corriente , tensión, altura del arco, velocidad del aportación del hilo o cambio de electrodo, entre otros parámetros.

Este tipo de maquinarias son sencillas de utilizar pero por lo contrario su mecanismo es robusto, al largo de los años este mecanismo ha variado y se ha disminuido de tamaño y peso considerablemente, anteriormente este procedimiento solo es efectuado con corriente alterna, pero el gran avance de la tecnología ha permitido poder realizarlo con corriente continua, lo que ha permitido soldar diferentes tipos de componentes.

Actualmente las fuentes de potencia para soldadura emplean tecnología inverter (figura 2.5) que permite tener una gran eficiencia a bajo consumo, menor tamaño y peso, así como generar un mejor proceso de soldadura, esto gracias a los avances tecnológicos.

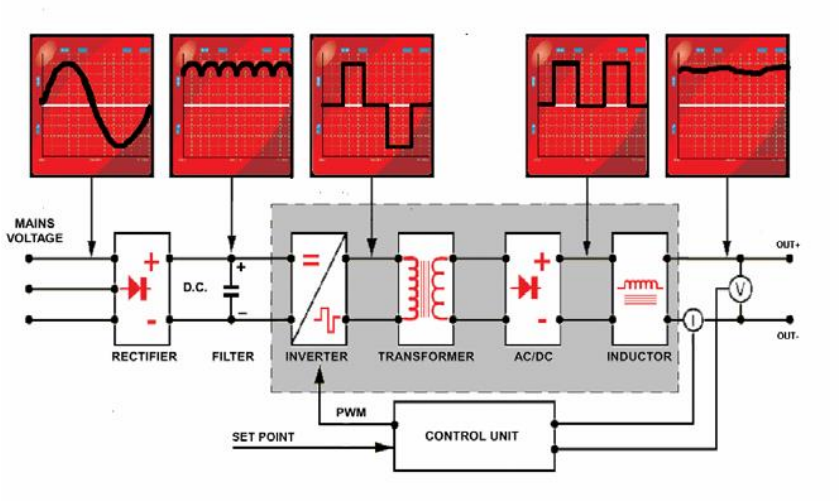


Figura 2.5 Tecnología invertir

2.2.6.2 Antorcha de soldadura

La antorcha de soldadura es el elemento terminal del equipamiento de soldadura para robot. Este depende del tipo de material y del espesor que se desee soldar así como de qué tipo de robot se desea utilizar y de la potencia necesaria para realizarlo, también existen diferentes tipos de antorchas dependiendo de la aplicación

Dependiendo de que materiales y que espesores se desean, también hay diferentes tipos de sistemas de refrigeración ya sea por agua o aire (figura 2.6 y 2.7), a esta parte final de la antorcha se le denomina “cuello de antorcha”, en donde este cuello está formado por distintas piezas, como la boquilla, por donde sale el hilo, difusor de gas para la soldadura, y la tobera de protección. Los cuellos pueden ser de 180º, de 45º y de 22º usualmente.

Las antorchas que son refrigeradas por agua, ésta llega hasta el porta tubo de contacto para refrigerar lo más cerca posible de la punta de contacto y las que son refrigeradas por aire, es el gas de soldadura el que además de proteger refrigera el cuello. El gas de protección fluye alrededor de la punta de contacto con el fin de proteger el baño de fusión, que es el material fundido que al enfriarse crea la soldadura.

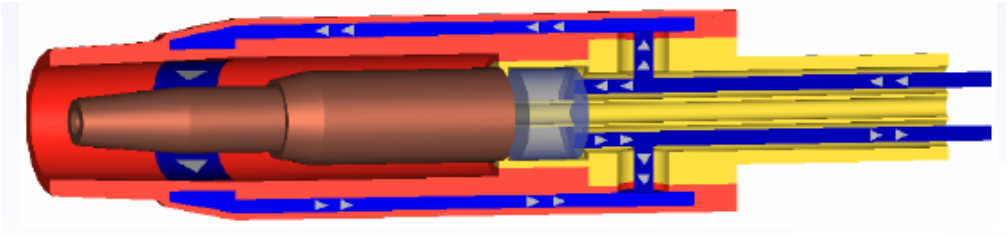


Figura 2.6 Antorcha refrigerada por agua

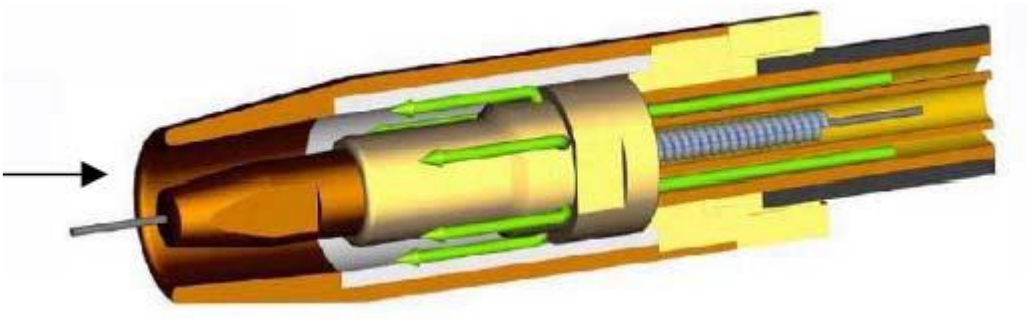


Figura 2.7 Antorcha refrigerada por Aire

Las antorchas por lo general pueden tener un motor pequeño para la alimentación del hilo, a este motor se le denomina “push-pull”, lo que realiza es ayudar al arrastre del hilo (figura 2.8) de soldadura, es muy usado al utilizar hilo de aluminio pues este suelta un pequeño polvo el cual ocasiona problemas de arrastre y con este sistema se evita, también este sistema se sintoniza con los rodillos devanadores de hilo para que estos dos sistemas trabajen en conjunto, este sistema permite trasladar los parámetros de arrastre que se sitúan en la máquina de soldadura a la devanadora al cuello de la antorcha

Figura 2.8 Motor de arrastre

2.2.6.3 Sistema anticolisión

Este es un sistema de seguridad el cual nos permite proteger la antorcha de golpes posibles así como de posibles colisiones como se muestra en la figura 2.9, este sistemas se desplaza en cualquier dirección a causa de una colisión, y se detiene el movimiento del robot, como si se tratase de un pulsador de emergencia, cortando la alimentación al motor y provocando que se detenga totalmente.

Figura 2.9 Sistema anticolisión

2.2.7 Ejes de Base.

Existen en la robótica una extensión del propio robot denominada ejes de base la cual permite que el robot se desplace por una superficie para poder alcanzar todos los puntos a soldar, el cual ha generado un gran avance permitiendo incrementar el rango de trabajo de robot, en innumerables formas, para el caso particular de soldadura, es necesario determinar si las piezas a soldar son de grande tamaño o de tamaño más compacto, pues esto dará respuesta a la pregunta ¿es necesaria una extensión del robot (eje de base) ?. Existen 3 tipos los cuales son el Track neumático Transversal, el track transversal servo-controlado, sistema Gantry.

El track neumático transversal, es un eje de dos o tres posiciones el cual cuenta con un movimiento lineal en donde el robot se encuentra embarcado y con ello el rango del robot aumenta (figura 2.10).

Figura 2.10 Track neumático transversal (Coca García J, P. 24).

El track transversal servo-controlado (figura 2.11) es una evolución del anterior pues este cuenta con una base servo-controlada con "un motor de 1,3 KW de potencia, controlado por un variador de frecuencia para controlar la posición de éste durante toda la carrera del track mediante el encoder absoluto acoplado al motor de translación" (Coca García J, P. 24).

Figura 2.11 Track transversal servo-controlado (Coca García J, P. 24).

Sistema Gantry (figura 2.12), estos sistemas están basados en ser utilizados en una industria mucho más grande como la naval o aeronáutica, su principio fundamental es el de un track transversal servo-controlado, pero a diferencia de este, este hecho para trabajar con sistemas

más grandes pues esta puesto en una poción invertida, para abarcar espacios de trabajo más grandes (Coca García J, P. 26).

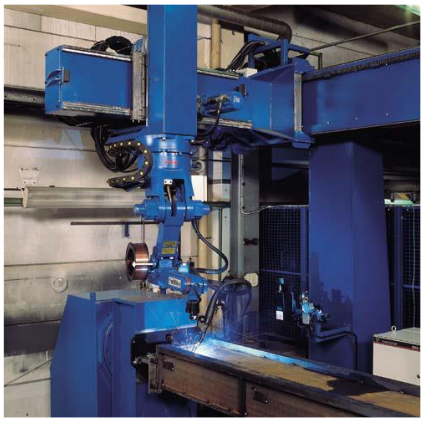


Figura 2.12 Sistema Gantry (Coca García J, P. 26).

2.2.8 Ejes Externos.

"Para posicionar las piezas, a veces es necesario mover, girar o voltear las piezas a soldar para colocarlas en una posición óptima para realizar la soldadura. A estos movimientos y/o giros de las piezas, se les conoce como posicionamiento, existen dos sistemas de control para posicionadores, éstos pueden tener un control neumático, o un servo-control de uno o varios motores en los que se colocan mediante unos platos de sujeción, los utillajes que sujetan la pieza para ser soldada, el posicionador servo-controlado, está formado por uno o varios ejes, según el modelo de posicionador. A estos ejes se les conoce con la denominación de ejes externos, ya que a diferencia de los ejes de base, éstos no se encuentran en contacto con el robot". (Coca García J, P. 27).

Pero también existen condiciones de piezas en las cuales no sea necesaria, rotarla ni reposicionarla para esto existen las mesas neumáticas el cual es un tipo de posicionador (figura 2.13), esta mesa permite para posicionar las piezas siempre en la misma posición, puesto que el control de giro de la mesa es neumático, lo cual no permite movimientos intermedios controlados, (Coca García J, P. 27) esta mesa es aplicable solo para piezas que no superen los 75 kg.

Figura 2.13 Mesa posicionadora (Coca García J, P. 27).

También existen los de ejes externos están destinados a posicionar utillajes para piezas en los que se necesita rotar las piezas o realizar reposiciones en el momento del soldeo (figura 2.14).

Figura 2.14 Eje externo (Coca García J, P. 27).

Los de Eje externo de un motor embargable (figura 2.15) se emplean cuando se pretende tener una mesa giratoria con zona de carga/descarga, donde el operario de la célula prepara la pieza y la retira una vez ésta está soldada, y una zona de trabajo, en la que el robot ejecuta los movimientos y acciones para realizar la soldadura. (Coca García J, P. 29).

Figura 2.15 Eje externo de un motor (Coca García J, P. 29).

Estos son por mencionar algunos, que pueden ser necesarios en la utilización de soldadura robotizada.

2.2.9 Controladores lógicos programables (PLC)

La automatización en su principio se basaba en sistemas eléctricos como son la lógica con relevadores (RLS), esto fue durante los años 60´s al realizar alguna modificación a estos sistemas conlleva muchas horas hombre, por esta razón se realizaron los tableros prefabricados en donde reducían el tiempo de modificaciones y contaban con modelos específicos para realizar un cambio rápido al tener un desperfecto.

General Motors solicito un desarrollo por la gran desventaja que tenían los sistemas con RLS, por lo tanto a finales de los 60´s y principios de los 70´s Bedford Associates creo el primer PLC que llevaba por nombre 084 pues fue el proyecto no. 84 de Bedford Associates, no tenía mucha capacidad pero fue un gran avance contando las limitaciones de los RLS. (Controlador, 2018)

Eventualmente el PLC (figura 2.16) fue cubriendo gran parte de los requerimientos de flexibilidad y adaptación de los sistemas de producción y ofreciendo numerosas ventajas, como la disminución notable de espacio y los diferentes tipos que se encuentran en el mercado, así como la disminución notable de componentes mecánicos y cableados.

Dado a todos los avances que ha tenido el PLC entre ellos el cambio de control de Hardware a Software esto ha permitido que ya no este atado a componentes físicos y los cambios se puedan realizar en software desde un entorno de desarrollo operando en PC, esto simplificando el control y ahorrando tiempo en la realización de ellos.

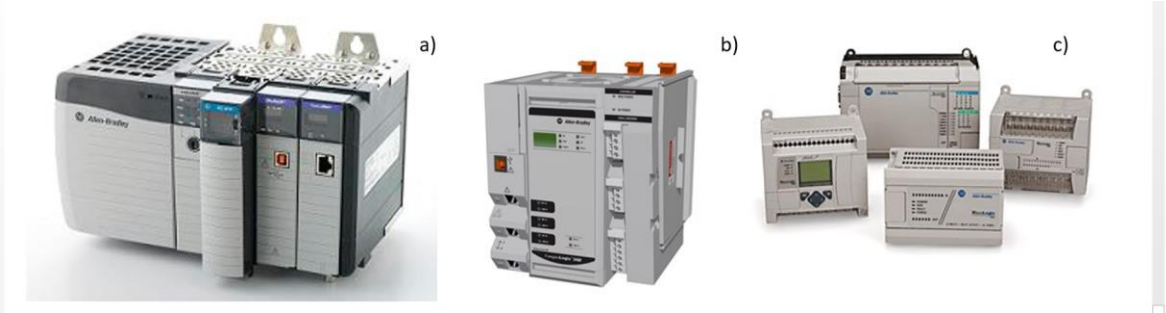


Figura 2.16 Controladores Compact Logix 5480 (a) Logic Controler Systems (b) Micro Logix 1100 Programmable (c)

2.2.10 Paradigma Orientado a Objetos en el PLC

El paradigma orientado a objetos (POO/OOP) y en particular la programación orientada a objeto, abstrae generalidades de las aplicaciones y las encapsula a manera de clases, en donde estas contienen variables, funciones y métodos, la cual es identificada por medio de objetos. Las técnicas de clasificación, herencia, abstracción, cohesión y encapsulamiento es como se define, esta programación alcanzo su difusión en los años 90´s, en la actualidad una gran variedad de lenguajes la utilizan y es soportada por medio de programación orientada a objetos.

Un objeto contiene información la cual permite identificarlo con respecto a otros objetos los cuales pueden pertenecer a la misma clase o a una distinta, esto por medio de los atributos los cuales son distintos en cada clase, para la interacción entre ellos, estos contienen métodos y con ello se efectúa su comunicación.

El lenguaje de programación POO cuenta con una programación estructurada camuflada, en donde los atributos y los métodos están relacionados por la propiedad de conjunto, con esto se determina que una clase requiere métodos para poder tratar los atributos.

A diferencia de la programación tradicional el POO se basa en que primero se definan los objetos y posteriormente se realicen los métodos por si mismos a diferencia de la tradicional que en ella se plantean primero los procedimientos y funciones y posteriormente la estructura en la cual son llamados.

En la ingeniería en sistemas está ampliamente desarrollado lenguajes y/o técnicas en donde se permite el desarrollo se sistemas en base a objetos.

La programación orientada a objetos en el PLC tiene una limitante, la madurez de los lenguajes de programación en ellos, dado que esta no se encuentra lo suficientemente madurada el PLC tiene como apoyo la PC para realizar esta programación en donde es menos complejo su aplicación dado los altos niveles en lenguajes de programación, como es en el caso de la instanciación.

Actualmente se ha trabajado mediante la utilización de las Redes de Petri (RdP) en donde existen muchos trabajos en los cuales se utiliza para modelar y controlar sistemas a eventos discretos dentro de los cuales se puede mencionar (Desrochers, A., 1995) y (Zhou, M.,1996), donde RdP tiene potencial para modelar sistemas, mientras tanto (Silva, M., 1892) tiene un capítulo en donde se ve la implementación al PLC, pero (Zhou, M., 1994 y 1995) da a conocer un método para genera automáticamente el código de programas mediante la traducción de modelos de RdP a contactos para PLC.

Conforme a esto se han realizado más investigaciones como es la de (Zapata, G., Carrasco, E.) en donde ellos elaboran una estructura para ser utilizada en el PLC por medio de la OOP y RdP, así mismo se ha tenido un gran avance al realizar metodologías para la aplicación en el PLC como es en el caso de (Zapata, G., Quintero, L.F) en donde ellos realizaron una metodología para el modelado y generación de código de control mediante redes de Petri para PLC dando como resultado de su investigación la generación de código para el PLC por medio de RdP, según la norma IEC 61133, mediante bloques funcionales.

En los sistemas orientados a objetos se cuenta con diferentes técnicas de modelado para su desarrollo, como son UML (lenguaje modelado unificado), máquinas de estado finito, diagramas de clases, en donde se define como un comportamiento que especifica la secuencia de los estados por la que un objeto pasa atreves de la respuesta a eventos, en las cuales se usan para describir el comportamiento de una clase durante el ciclo de vida.

2.3 Técnica de modelado iMRP

En el año 2005 ISO aprobó el UML (Unified Modeling Lenguage) y en el año 2012 se actualizo la norma a la última versión disponible 2.4.1 la cual no es un lenguaje de programación, es una manera de modelar el comportamiento de un proceso, el cual no especifica la metodología a utilizar, las técnicas que parten de la utilización del UML son orientadas al desarrollo de un sistema específico, con lo cual al modificar algún requerimiento esto conlleva a la modificación del modelo y la recodificación del programa, esta situación se puede identificar con el concepto de implementación dirigida por modelo, por lo tanto estas técnicas con complejas de aplicar en el concepto de operación dirigida por modelo, la técnica de modelado iMRP permite la implementación del concepto de operación dirigida por modelo como técnica básica, con el cual se permite la reusabilidad de código.

El iMRP (integrador modular de recursos de producción) es un método de modelado de la información en el cual es implementado directamente en un sistema informático para coordinar el flujo tanto de piezas, procesamiento y materiales (Acosta J.E, 2016), este modelo permite la abstracción de características particulares y específicas de un sistema, donde la característica principal es que se utilizan únicamente arcos y nodos , estos a su vez están asociados a cuatro acciones principales, Productos, Recurso Tiempo, Operaciones y Equipos, estos son representados por medio de nodos y la interacción de ellos se realiza por medio de arcos, este método permite la abstracción del procedimiento de modo eficaz y sencillo dando características específicas al nodo y al arco (Acosta & Sastron, 2007).

Para entender fácilmente el modelo iMRP (figura 2.17), es necesario tener una vista de los elementos del sistema en donde nos permita identificar fácilmente cada uno de los elementos,

para realizar este grafo hay ciertos lineamientos que hay que tener en cuenta (Acosta J.E 2016):

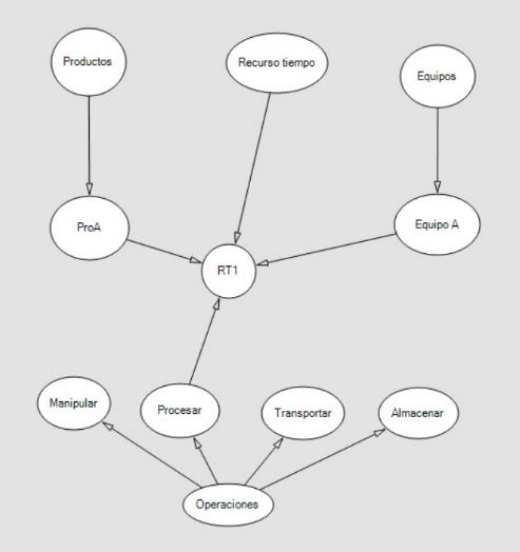


Figura 2.17 Modelo iMRP

Grafo Simple: se cuenta con cuatro tipos de recursos básicos para la fabricación de un producto: Pieza/Producto, Tiempo, Operación y Equipo, el conjunto de un mismo tipo y la asociación entre ellos es representada como grafo simple.

Arco: la interdependencia es representada en el modelo por medio de un arco o arista, esta puede existir en elementos de la misma naturaleza o de diferente naturaleza, las características principales de la asociación son representadas con atributos los cuales son: ID, GrafoID, Cantidad, Var1, Var2, Var3 (Acosta, J.E. Anexo 2016).

2.4 OPC

El OPC por sus siglas en español de OLE for Process Control, es un servidor que se conforma por dos plataformas cliente y servidor, la plataforma de servidor OPC es donde se realiza las interfaces de comunicación utilizando una o más fuentes de datos de sus protocolos nativos típicamente utilizados como son: DCSs, PLCs, Basculas, módulos I/O, controladores, entre

otros, en el módulo de cliente OPC es donde se utiliza típicamente los sistemas SCADAs, HMIs, aplicaciones de cálculo entre otros.

En esta arquitectura el Cliente OPC/Servidor OPC el servidor OPC es el esclavo mientras que el Cliente OPC es el maestro (http://matrikonopc.es/opc-servidor/index.aspx) en donde se cuenta con una dirección bidireccional, lo que proporciona que los clientes puedan leer y escribir en los dispositivos a través del servidor OPC (figura 2.18).

Se cuenta con diferentes tipos de OPC como son los siguientes:

- Servidor OPC DA – Basado en Spezifikationsbasis: OPC Data Access - especialmente diseñado para la transmisión de datos en tiempo real.
- Servidor OPC HDA– Basado en la especificación de Acceso a Datos Historizados que provee al Cliente OPC HDA de datos históricos.
- Servidor OPC A&E Server– Basado en la especificación de Alarmas y Eventos – transfiere Alarmas y Eventos desde el dispositivo hacia el Cliente OPC A&E.
- Servidor OPC UA – Basado en la especificación de Arquitectura Unificada – basado en el set más nuevo y avanzado de la OPC Foundation, permite a los Servidores OPC trabajar con cualquier tipo de datos.

Los servidores de OPC COM/DCOM utilizan una infraestructura de Microsoft Windows la cual es para el intercambio de datos, con ello estos se deben de instalar estrictamente en este sistema operativo, el cual puede soportar comunicación con múltiples clientes OPC simultáneamente.

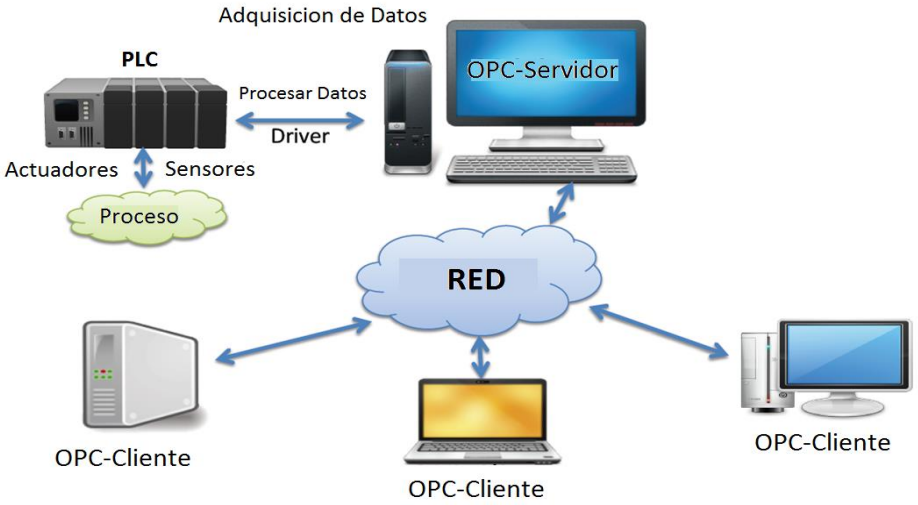


Figura 2.18 Funcionamiento de un OPC

La principal función de un OPC es traducir los datos nativos de la fuente de datos a un formato OPC que sea compatible, esto provoca que la calidad de traducción del protocolo nativo a OPC y de OPC a nativo dependan enteramente de la implementación del desarrollador del servidor OPC.

Los protocolos de comunicación de un OPC varían dependiendo de la utilización, puede ser por medio de conexiones físicas o por su propio protocolo.

CAPITULO 3
PLANTEAMIENTO DEL PROBLEMA

3.1 INTRODUCCION

Uno de los elementos clave en el control de una célula robotizada ha sido el PLC; la reutilización del código de PLC entre diferentes células es una característica deseable que facilitaría el desarrollo y actualización de estaciones de producción automatizadas, sin embargo, la reutilización del código en PLC es un problema escasamente abordado en la literatura. En el presente proyecto de tesis se propone analizar el problema y realizar un planteamiento que coadyuve al desarrollo de sistemas de control flexibles, basados en PLC. Para tal efecto se tomarán en cuenta varios aspectos tales como el acoplamiento débil en la integración practica de los sensores/actuadores y el concepto de operación dirigida por modelo, buscando con ello soluciones que conduzcan a la reusabilidad de código, de tal manera que soporte ingresar un nuevo robot o cambiar uno ya existente utilizando el mismo (núcleo) del programa y facilitara el acople con lo ya existente. Dada las características de las plataformas de desarrollo de sistemas basados en PLC (lenguajes de programación) es importante visualizar la necesidad de modificación de algún parámetro. Desarrollando o aplicando una nueva técnica o tecnología, entonces se habrá justificado objetiva y formalmente el tema de tesis.

En ingeniería de software existen conceptos no utilizados en este ámbito (PLC) como es Acoplamiento débil y Operación dirigida por modelo, en donde este último es ampliamente utilizado en el área de modelado de procesos de negocios, pero no ha sido utilizado en el área de producción así como enfocado a la industria, esto se debe por la falta de técnicas de modelado del mismo dando con ello la falta de abstracciones del modelo y con ello su nula implementación.

En el presente proyecto de tesis se plantea utilizar la técnica de modelado de particularidades denominada iMRP (Acosta & Sastrón, 2007), como base para aplicar el concepto de operación dirigida por modelo.

Con ello se pretende obtener una flexibilidad en el sistema en donde aplicando el concepto de ArquiTAM (Acosta J.E, 2016) en el cual el desarrollo de herramientas planteadas ofrece una flexibilidad tanto en el piso de producción como en el equipo.

3.2 OBJETIVO GENERAL

Desarrollar un sistema flexible de control basado en PLC para una estación robotizada aplicando los conceptos de acoplamiento débil y operación dirigida por modelo.

3.3 OBJETIVO ESPECIFICO

- Aplicar el concepto de acoplamiento débil en la integración PLC-Sensores/Actuadores.
- Desarrollo de un sistema genérico en PLC a operar con base en la técnica de modelado iMRP.
- Aplicar iMRP en la operación de una estación robotizada controlada por PLC.

3.4 TECNICAS A APLICAR EN LA SOLUCIÓN DEL PROBLEMA

Se desarrollara por medio de la estructura ArquiTAM en general, la estructura se utiliza para simplificar la comprensión, diseño y operación de sistemas así como un resumen de la información pertinente en el dominio del problema.

La estructura plantea el desarrollo de un marco de trabajo (framework), el cual es utilizado o particularizado para aplicarse en una situación específica.

La arquitectura de referencia (nivel conceptual) implementa como base para el desarrollo del modelo de referencia o marco de trabajo plantea el uso de técnicas comúnmente empleadas como soporte del problema de integración a nivel componente tecnológico, tales como herramientas de cero configuración, colas de mensaje, servicios web, OPC, permitiendo así ubicar el apoyo de estas técnicas y su relación con los otros componentes de acoplamiento.

La aplicación del marco de referencia en una situación específica (o caso particular) se utiliza el concepto de implementación y operación dirigida por modelo, utilizando la técnica de modelado iMRP (Abstracción de particularidades del piso de producción), como medio de

abstracción de las particularidades del caso específico de aplicación a ser interpretado por el marco de referencia en tiempo de ejecución para la instanciación de la aplicación particular.

CAPITULO 4
DESARROLLO DE LA SOLUCIÓN PROPUESTA

4.1 INTRODUCCION

En el presente capítulo se describe el desarrollo del sistema para obtener la solución del problema planteado en el capítulo anterior, para ello se tomó como base la arquitectura ArquiTAM con los conceptos de Operación dirigida por modelo en el cual se obtiene la reusabilidad del Framework en donde se contempla que el modelo es cambiante pero siempre dentro de una misma gama (o familia) como se muestra en la figura 4.1, lo que permite que se tengan ciertas particularidades específicas para cada caso, por lo tanto es posible crear un modelo genérico en donde se incluya las características comunes a todos los casos (caso general) que son aplicables en todos los tipos de modelos. A partir del modelo genérico se instancia con base en el modelo de particularidades el modelo o caso particular

La instanciación de modelos particulares se lleva a cabo por medio de la utilización del modelo genérico en el cual se permite ingresar un conjunto de variables y métodos específicos los cuales son utilizadas para la creación de objetos, estos se denominan instancias, teniendo en cuenta que el concepto de clase no está fuertemente implementado en la utilización en el PLC esto conlleva a realizarlo desde una computadora personal y con el apoyo de un lenguaje de alto nivel en este caso fue Visual Basis.Net, teniendo esto en cuenta la realización de las instanciaciones generadas por medio del modelo genérico, si se toma como ejemplo la instanciación de modelo genérico tipo “motor” el cual es una clase, al ingresarle diferentes entradas y salidas pueden existir varias copias o instancias de el por ejemplo “motor_1”, “motor_2” y así sucesivamente como se muestra en la figura 4.1

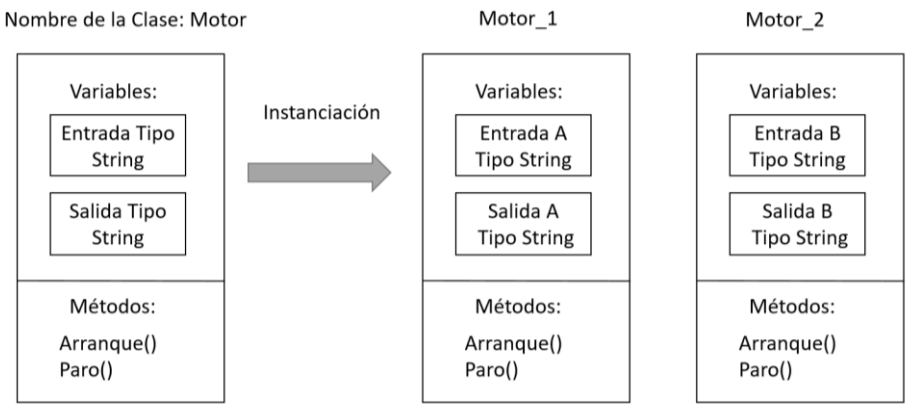


Figura 4.1 Muestra la creación de instanciaciones u objetos por medio de una clase.

Se puede hacer uso de la clase para generar “n” cantidad de objetos, en los cuales se asignan las variables y con ello se crea un nuevo objeto con los valores de variables correspondientes, teniendo así modelos particulares a partir de la instanciación del modelo genérico.

La aplicación de los modelos genéricos en un caso particular está apoyada por el concepto de implementación y operación dirigida por modelo en donde el modelo genérico es adaptado a las necesidades que se tienen en la aplicación particular, es decir, la aplicación (modelo genérico) identifica y genera las instancias requeridas por el caso particular y realiza la operación del sistema.

De esta manera es posible afirmar que el modelo particular es instanciado a partir del modelo genérico mediante la interpretación y operación que se realiza en código en la PC la cual interpreta el modelo de particularidades (iMRP) y opera por medio de la operación dirigida por el modelo de particularidades interpretado como se ilustra en la figura 4.2

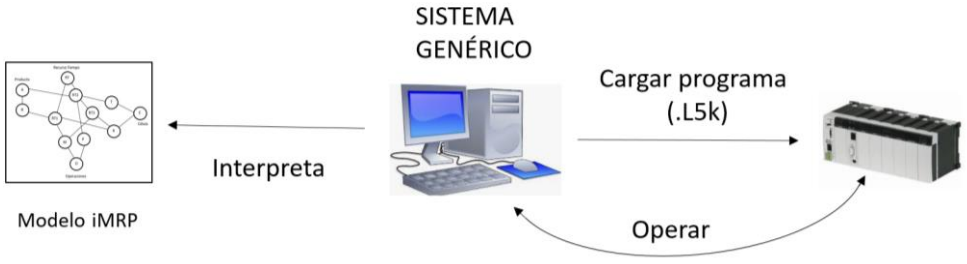


Figura 4.2 Descripción grafica de la operación dirigida por modelo.

Una parte esencial para llevar a cabo la operación del sistema se realiza por medio del acoplamiento débil en donde este es parte de la integración del sistema, en la figura 4.3

Figura 4.3 Sistema Propuesto, esquematización de acoplamiento débil.

Como se muestra en la figura 4.3 el problema del sistema se conforma de dos subsistemas el informático y el físico, a integrar con base en el concepto de acoplamiento débil (Acosta, J. E., 2015) En la implementación del concepto de acoplamiento débil se encuentran cuatro etapas: el sistema de control, protocolo de comunicación, el sistema envoltura y el driver, en donde estas cuatro etapas son propias para la integración del sistema (PC-PLC). En el modelo de particularidades se especifica el elemento del sistema a utilizar para manejar el protocolo de comunicación a utilizar en el caso particular. El sistema de control interpreta el modelo de particularidades (iMRP) para la instanciación del objeto de la clase correspondiente al protocolo de comunicación. Así mediante incorporación dinámica de código (en tiempo de ejecución) es posible manejar en el sistema diferentes tipos de protocolos de comunicación. La siguiente etapa definida en el esquema de acoplamiento débil corresponde al elemento envoltura. Esta etapa responde al manejo en forma remota del PLC. El objeto envoltura crea y se comunica con el objeto driver correspondiente el cual atiende los detalles de comunicación del PLC específico.

El esquema de acoplamiento débil soporta el envío y recepción de ordenes e información hacía y desde el PLC en forma independiente de las particularidades del sistema (tipo de PLC, plataforma de comunicación remota, medios físicos de comunicación, entre otras), aportando así flexibilidad al sistema permitiendo que cada bloque tome sus propias decisiones conforme a lo recibido.

En la figura 4.4 se muestra en forma esquemática el funcionamiento de la solución propuesta, en donde el sistema genérico que se encuentra en una PC que interpreta el modelo iMRP en el cual se abstraen las particularidades del sistema. Entre las particularidades identificadas en el modelo se encuentran las clases a partir de las cuales se generan instancias de código del PLC (en formato ".L5K", el cual es cargado al PLC. Una vez cargados tales objetos en el PLC, el sistema genérico obtiene la lógica de operación del programa en el PLC a partir del modelo iMRP. De esta manera se implementa el concepto de operación dirigida por modelo.

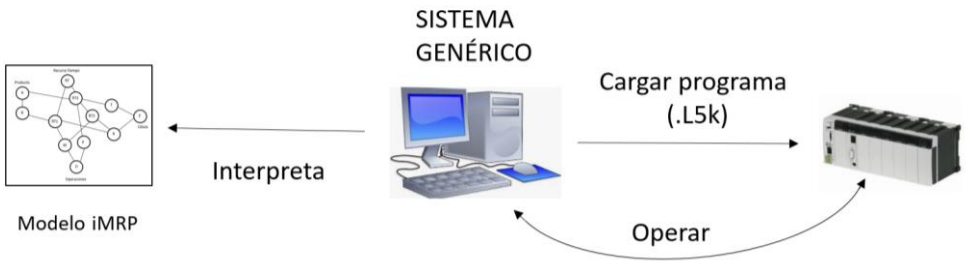


Figura 4.4 Descripción grafica de la solución propuesta utilizando el acoplamiento débil.

4.2 ARQUITECTURA DEL ESQUEMA PROPUESTO

El esquema propuesto para la solución al problema se basa en el sistema genérico o framework en donde se encuentran implementadas las clases comunes a todo sistema esto es, los posibles parámetros o generalidades que pueden ser requeridos en dado momento. Tal sistema genérico es considerado como una meta instanciación de la arquitectura de referencia previamente desarrollada para este tipo de sistemas (Acosta, J. E., 2015).

A partir del modelo iMRP que abstrae las particularidades de un sistema en particular, el modelo genérico implementado en código (framework) instancia los objetos específicos que integran al sistema de control. Una vez implementado el sistema particular (implementación dirigida por modelo) el sistema genérico (convertido en sistema particular) actúa como lanzador de órdenes, dirigido por el modelo iMRP, implementando así el concepto de operación dirigida por modelo.

La lógica a seguir para la generación y operación del programa parte del modelo gráfico de particularidades iMRP, en donde se especifica tanto la estructura del sistema particular como la lógica de operación del mismo, mediante la estructura del producto (nodos), el número de equipos (nodos) y el número de recursos tiempo (nodos). En cada uno de los arcos y nodos se especifica el valor de ciertos parámetros como son la dirección de la envoltura del equipo y la dirección del documento en el cual se encuentran los parámetros para la inserción de código. Así mismo en los atributos de arco se asignan valores de elementos del sistema que permiten la realización de cada una de las operaciones.

En el caso de la implementación del sistema con base en PLC se toman las propiedades específicas de los nodos y arcos del modelo iMRP utilizando las clases especificadas en el modelo e implementadas para el sistema particular, se instancian los objetos que contienen el código correspondiente a cargar en el PLC por la clase respectiva.

En el arco entre el nodo recurso tiempo y nodo equipo (actuador) se define el nombre de la etiqueta (Tag) de la rutina para el manejo del actuador correspondiente (representado por el nodo respectivo). De la misma manera en el arco entre recurso tiempo y nodo operación se define la etiqueta (Tag) en donde se identifica el fin de la operación realizada por la rutina correspondiente al actuador.

El sistema genérico se ejecuta en la PC. La comunicación de la PC con el PLC tanto para la carga de código como para su operación se realiza mediante el esquema de acoplamiento débil en donde el lanzador a través de los objetos envoltura y driver se conectan con el PLC correspondiente, Figura 4.5.

Figura 4.5 Diagrama a bloques representativos de la solución propuesta

4.3 IMPLEMENTACIÓN DIRIGIDA POR MODELO

4.3.1 ArquiTAM

En el esquema de referencia ArquiTAM se plantea una estrategia para desarrollo del sistema estructurada en tres niveles de abstracción: Arquitectura de Referencia, Modelo de Referencia y Aplicación particular. A nivel arquitectura de referencia se desarrolla la parte conceptual del problema como es identificación del problema, requerimientos y conceptos de solución. En el nivel de modelo de referencia se desarrolla el sistema a nivel de aplicación

genérica a manera de marco de trabajo (conocido en el mundo anglosajón como framework). A partir del modelo o sistema de referencia se instancia el modelo o sistema particular tomando como base el modelo de particularidades del caso o sistema particular. De esta manera el sistema o modelo de referencia es de aplicación en diferentes casos particulares bajo los conceptos de implementación (instanciación) y operación dirigida por modelo.

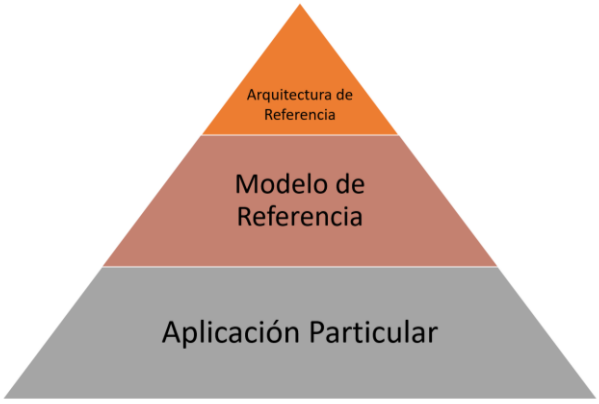


Figura 4.6 Representación del modelo ArquiTAM

El desarrollo del modelo de particularidades de un caso específico de aplicación, implica menor esfuerzo que desarrollar el sistema completo para el caso en cuestión. Tal afirmación es explicada en (Acosta, J. E., 2015) bajo la idea de que el sistema completo de un caso particular está conformado por la implementación de las características genéricas o comunes a cualquier sistema más la implementación de las características particulares del caso en cuestión. Por lo tanto la abstracción o modelado únicamente de las particularidades resulta de menor complejidad que la abstracción del sistema completo, como se indica en la figura 4.7

Figura 4.7 Reducción de la complejidad en el desarrollo del modelo.

El modelo de particularidades, identificado como iMRP (acrónimo de integrador modular de recursos de producción), consiste en una representación gráfica de las particularidades de un sistema (estructura y lógica de operación); es una técnica gráfica de modelado con base en los constructores nodo y arco. Los recursos de fabricación como piezas, equipos, operaciones y tiempo están representados por nodos. Los nodos tienen atributos cuyo significado depende del tipo de recurso representado por el nodo. Por ejemplo: si el nodo representa un equipo en los atributos se representa información tal como su dirección en la red, en el caso del nodo que representa una pieza en la estructura del producto, en los atributos se define la cantidad de tales piezas que se requieren en la estructura del producto terminado. Mediante arcos se establecen dos tipos de relación: secuencia y agregación. La secuencia en la operación está dictada por la secuencia de fabricación del producto, siguiendo la estructura del producto y los recursos tiempo utilizados para la fabricación de la pieza. Cada recurso tiempo está asociado a tres nodos: pieza, operación y equipo, como se muestra en la Fig. 4.8.

En el presente proyecto de tesis la técnica de modelado iMRP aplicada al caso de sistemas de operación automática controlados por controlador lógico programable (PLC). En tal caso los equipos son sustituidos por actuadores, y las propiedades específicas para definir las rutinas correspondientes al manejo del actuador son identificadas en los atributos del nodo correspondiente. El control de cada rutina (inicio y fin de operación) se maneja a través de etiquetas (Tags) definidas en los arcos entre nodos recurso tiempo y nodo actuador (equipo) así como entre nodos recurso tiempo y operación, respectivamente Así el sistema genérico interpretando el modelo de particulares es guiado para la ejecución del sistema implementado en el PLC.

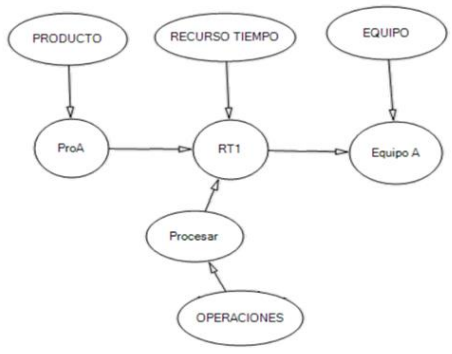


Figura 4.8 Modelo simple de un iMRP

4.3.2 iMRP

La técnica de modelo iMRP (Acosta & Sastrón, 2007) es una abstracción de características especiales del piso de producción, en la cual se utilizan únicamente nodos y arcos los cuales son instanciados a partir de clases, lo cual permite la flexibilidad al momento de crear n número de nodos del mismo tipo, en donde la diferencia entre ellos son los argumentos y el arco de referencia, los cuales tienen los siguientes argumentos (Figura 4.9):

CNodo	**CArco**
ID	ID
GrafoID	GrafoID
Tipo	Cantidad
Nombre	Var1
Var1	Var2
Var2	Var3
Var3	

Figura 4.9 Atributos de los Nodos y Arcos

Para la creación de un modelo iMRP, se utiliza un “Editor de iMRP”, (Gonzales, 2017) el cual fue desarrollado en base a nodos y arcos, en donde se le asignan atributos a cada arco y nodo, teniendo una interfaz dinámica y fácil de utilizar, la aplicación guarda el modelo iMRP creado en un archivo XML, en donde se despliegan todos los atributos anexados a cada arco y nodo, este archivo posteriormente se utiliza en la plataforma Visual Studio para de-serializarlo y utilizar todos los atributos previamente asignados en el editor de iMRP.(Figura 4.10)

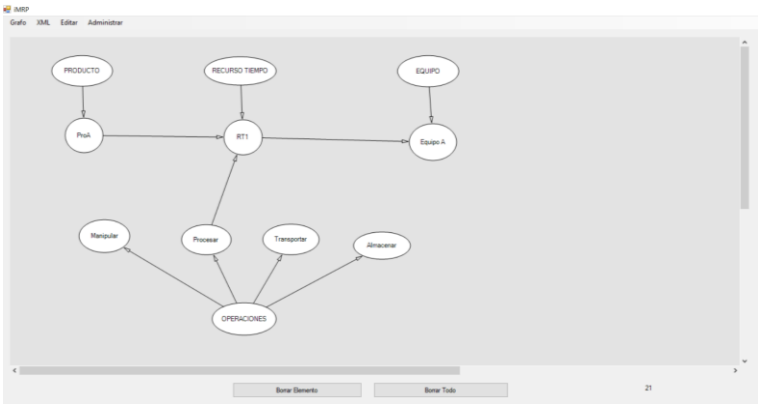


Figura 4.10 Grafo de iMRP

En el Grafo (editor) de iMRP hay ciertos atributos que se generan automáticamente ya sea de tipo Nodo o tipo Arco como se muestra en la figura 4.11

Tipo de Grafo	Producto	Recurso Tiempo	Operación	Equipo
ID	<Automatico>	<Automatico>	<Automatico>	<Automatico>
GrafoID	<Automatico>	<Automatico>	<Automatico>	<Automatico>
Tipo	Clase y Aplicación	Clase y Aplicación	Clase y Aplicación	Clase y Aplicación
Nombre	Opcional	Opcional	Opcional	Opcional
Nivel	<Automatico>	<Automatico>	<Automatico>	<Automatico>
Var1	Var1	Var1	Recepcion de Mensajes	Var1
Var2	Var2	Var2	Recepcion de Mensajes	Var2

Figura 4.11 Atributos del Grafo

Las variables de los diferentes tipos de grafos se utilizan de diferente manera según sea el tipo como se muestra en las figuras 4.12, 4.13 y 4.14

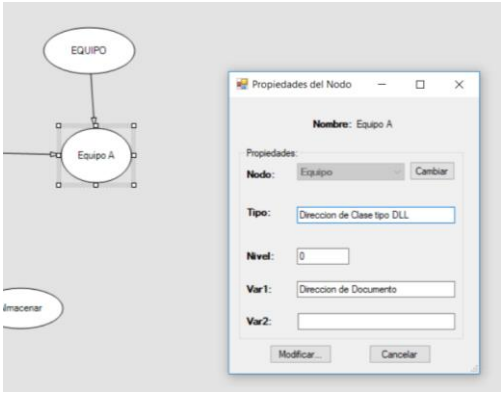


Figura 4.12 Propiedades del Nodo en Grafo iMRP

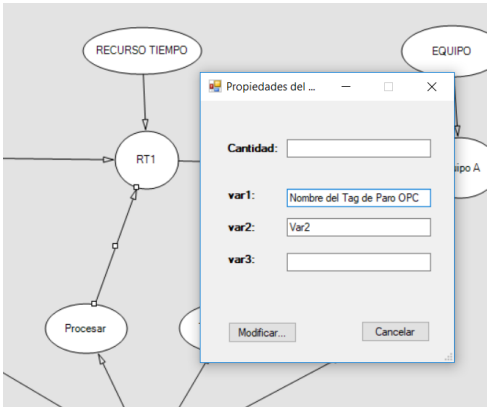


Figura 4.13 Propiedades del Arco procedente de operación a RT en Grafo iMRP

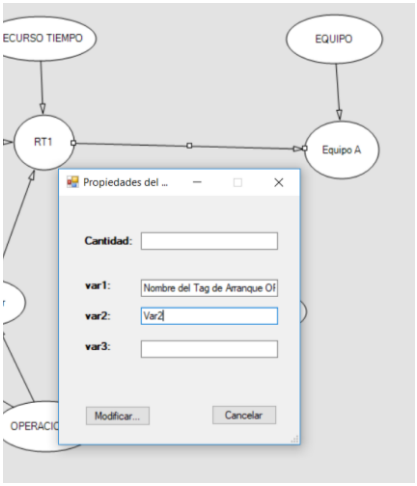


Figura 4.14 Propiedades del Arco procedente de RT a Equipo en Grafo iMRP

4.3.3 Modelado de particularidades.

Como ejemplo de abstracción de particularidades mediante la técnica iMRP, se presenta el modelado de un sistema que consiste en un Silo en donde la función principal es el llenado de una caja la cual se traslada por medio de una banda transportadora y es llenada por medio de una tolva como se muestra en la figura 4.15

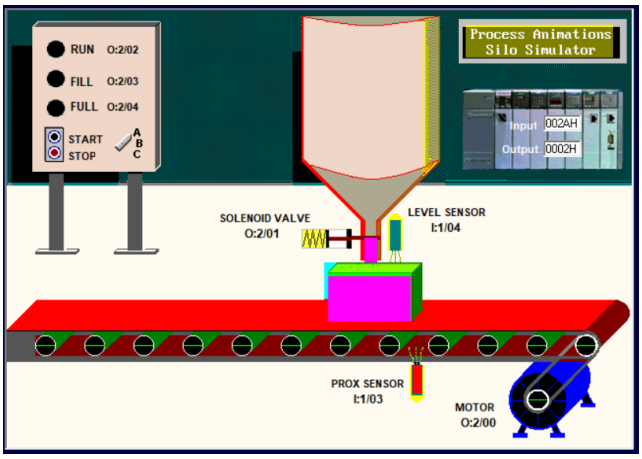


Figura 4.15 Funcionamiento de un Silo

Teniendo en cuenta este funcionamiento se planteó el modelo grafico del iMRP como se muestra en la figura 4.16, en donde se utiliza el grafo producto para abstraer diferentes etapas del producto (caja vacía, caja llena). El nodo P1 representa a la caja vacía. Este nodo asociado al nodo RT1, que representa el recurso tiempo que a su vez utiliza la banda transportadora (nodo banda motor en el grafo equipo), para realizar la operación de transporte (nodo transporte en el grafo operación). De acuerdo a la llógica iMRP este recurso tiempo es utilizado en primer lugar de la secuencia de operación representada en el modelo. De esta manera se acciona la operación de la Banda/Motor activando la etiqueta (Tag) correspondiente que en este caso es el I:1/0 como se muestra en la figura 4.17. El fin de operación del transporte, como todas las operaciones, es indicada mediante la etiqueta (Tag) definida en el atributo Variable1 del arco correspondiente Figura 4.16, a la asociación nodo recurso tiempo y nodo operación, en este caso la etiqueta I:1/3, En la rutina de transporte implementada en el PLC, esta etiqueta es activada mediante el sensor de proximidad que indica que la caja se encuentra en la ubicación requerida (abajo de la tolva para ser llenada). . Con la realización del recurso tiempo RT1, se termina con las operaciones asociadas al nodo P1 (caja vacía). Continuando con la interpretación del modelo iMRP, el siguiente nodo producto a realizar es el nodo P2 (caja llena). El recurso tiempo a realizar es el RT2 en donde se inicia el proceso de la tolva, esto es, el llenado de la caja, el RT2 utiliza el actuador de la tolva (nodo Actuador Tolva) y la operación de procesamiento (nodo procesamiento) para el proceso de llenado de la caja. . El inicio de este proceso se indica a través de la etiqueta I:1/3 (arco entre nodo RT2 y nodo Actuador de Tolva), asociada a la rutina en el PLC del actuador

de la tolva, Figura.4.16. El fin de la operación de llenado es señalada por la etiqueta I:1/4, Figura 4.16, indicada en el atributo Variable1 Del arco entre RT2 y nodo Procesamiento. Esta etiqueta es activada en la rutina del PLC (actuador tolva) al llegar el contenido de la caja al nivel correspondiente. En el modelo iMRP del sistema se indica que el siguiente recurso tiempo a realizar es el RT3. Este recurso utiliza el motor de la banda (nodo motor banda) y la operación de transporte (nodo transporte). Este recurso es empleado para retirar mediante la banda la caja llena. El inicio de la ejecución de este recurso se indica a través de la etiqueta I:1/4, Figura 4.16. El fin de operación es señalado por la rutina del PLC correspondiente al motor banda mediante la etiqueta I:1/2, indicada en el atributo Variable1 del arco entre RT3 y el nodo transporte.

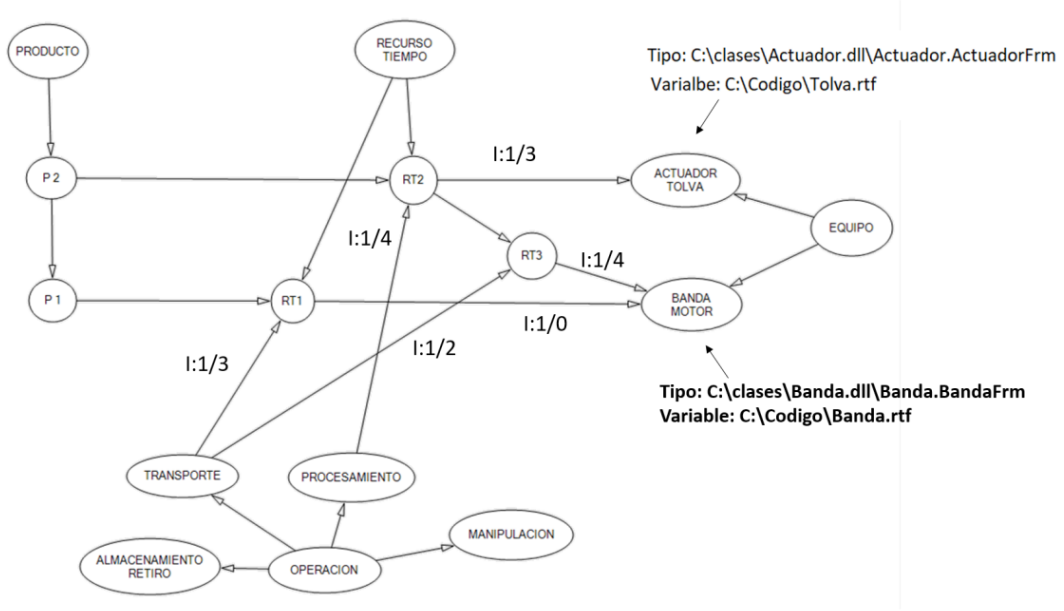


Figura 4.16 Grafo de modelo iMRP del funcionamiento de un Silo

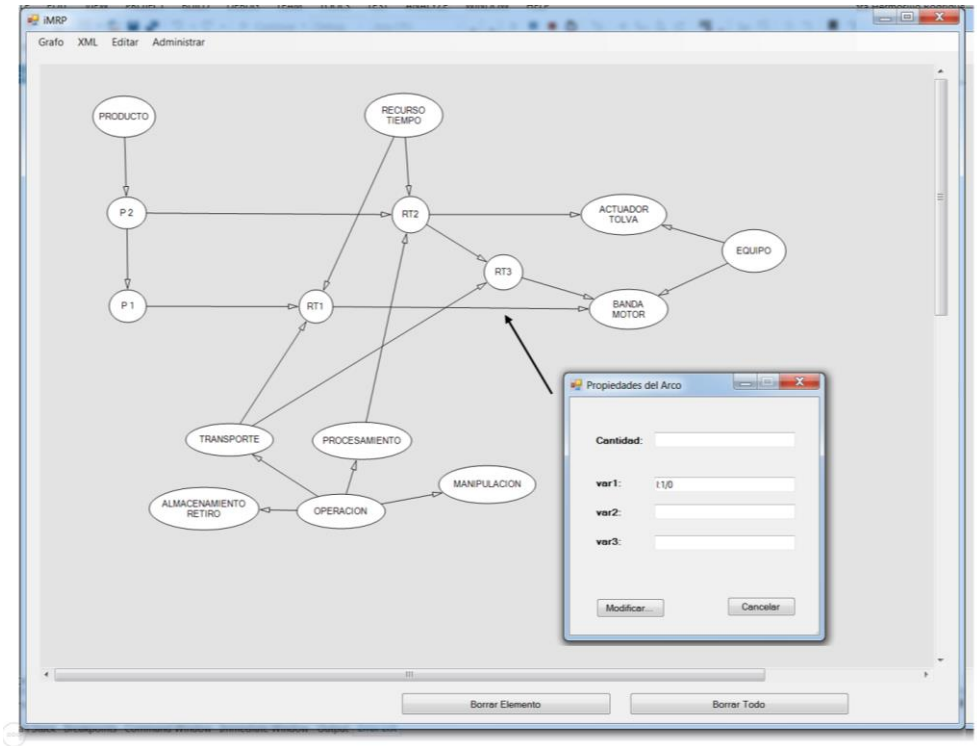


Figura 4.17 Tag de inicio de la Banda/Motor

Teniendo en cuenta el planteamiento del modelo iMRP, este a su vez realiza el proceso de instanciación en los nodos de tipo equipo desde el modelo genérico obteniendo con ello el modelo particular para cada uno de ellos que en este caso son: Banda/Motor, y Actuador/Tolva, cada uno de ellos pertenece a un tipo distinto como se muestra en la figura 4., también cada uno de ellos tienen variables distintas las cuales son propias para la inserción de código, en donde con estas variables se realiza la inserción del código correspondiente y posteriormente este es cargado al PLC.

4.3.4 Implementación y operación dirigida por modelo

Una vez determinada la secuencia de pasos a seguir, es necesario realizar la instanciación de las clases genéricas, para convertirlas en clases particulares. A partir del modelo iMRP, prosiguiendo con el ejemplo anterior la primera instanciación se realiza en el equipo correspondiente a Banda/Motor el cual es de Tipo Banda. En los atributos del nodo se encuentra el tipo de clase en donde se encuentra el código que genera en formato texto el objeto correspondiente a esta banda (archivo .L5K).

Otro objeto a instanciar del sistema particular es el equipo Actuador/Tolva el cual su clase genérica es de Tipo Actuador, en donde se encuentra el código en formato texto el cual será cargado al PLC (Figura 4.18). De esta misma manera se realiza la instanciación de cada uno de los objetos equipos requeridos. Posteriormente se prosigue con la inserción de código en el archivo programa del PLC con el texto correspondiente a los objetos instanciados (código del PLC). Así se genera el archivo con el código del PLC en un documento e (Rodríguez, D, 2017.) que sirve como base para la generación del archivo extensión ".L5K" de código compatible con el PLC para posteriormente cargarlo por medio al PLC compact Logix (RSLogix5000 y RsLinx).

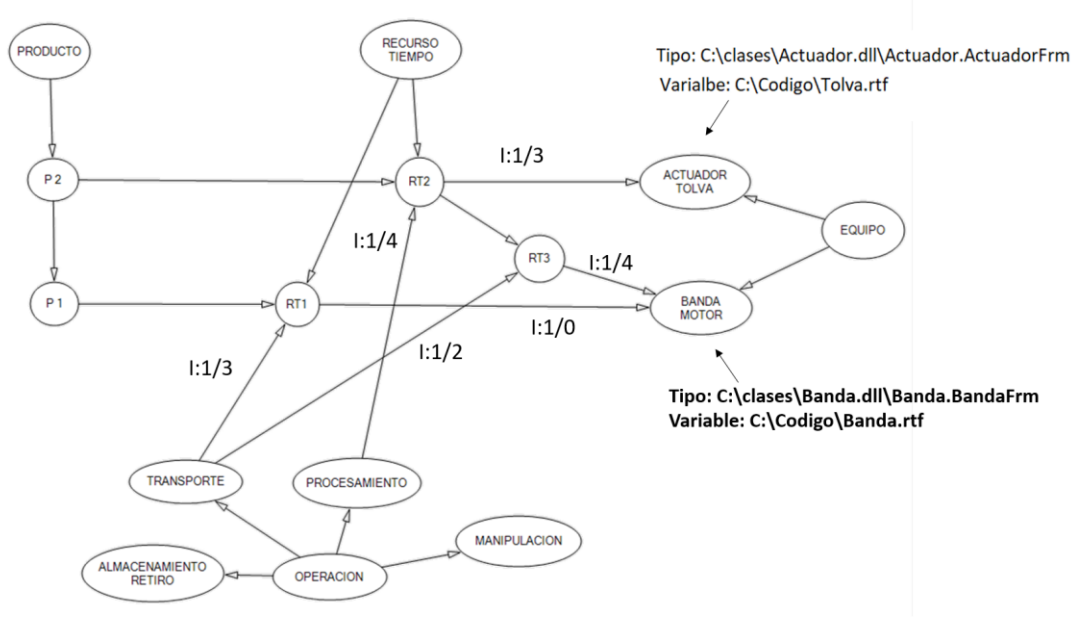


Figura 4.18 Modelo iMRP con propiedades del sistema de control de un Silo

4.3.5 Operación dirigida por modelo.

La operación dirigida por modelo, es aquella en donde el modelo de particularidades es interpretado por el sistema para identificar la estructura y la dinámica del sistema.

En el ejemplo anterior el modelo iMRP es el que determina la secuencia de operación del Silo, como se muestra en la figura 4.18.

El sistema genérico o modelo genérico se ejecuta en la computadora personal y realiza la instanciación del modelo o sistema particular con base en el modelo de particularidades grafo iMRP. En el modelo de particularidades se abstrae la estructura y dinámica del sistema particular, el cual determina la operación del sistema en el PLC coordinado por el sistema en la PC que interpreta el modelo de particularidades, Figura 4.19.

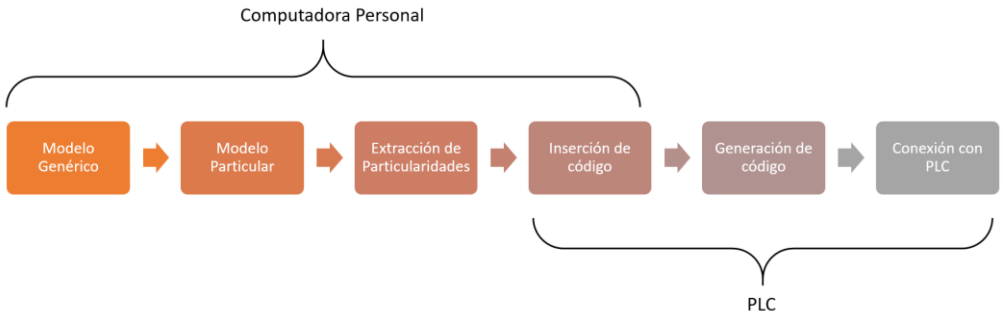


Figura. 4.19 Diagrama a bloques del Modelo Genérico a la Conexión con PLC

Es importante resaltar la comunicación que existe entre el sistema operando en PC y el sistema operando en el PLC. Tal comunicación requiere la implementación de una etapa de acoplamiento entre ambos.

4.4 INTEGRACIÓN PC-PLC

Para la integración de PC-PLC se utilizaron las clases envoltura y Driver, Figura 4.20. En donde la PC da pasó a la envoltura en el cual se determina el tipo de protocolo de comunicación así como también los requerimientos para realizarse, el OPC se aplica en esta sección, en él se determina qué tipo de conexión se realizara y con cual servidor de OPC se enlazara, así como los requerimientos propios de cada servidor, posteriormente cuando el OPC está realizado correctamente este se conecta con el Driver correspondiente al equipo (PLC), en donde por medio del RSLogix 5000 y del RsLinx los cuales son los programas propios de la conexión PC-PLC, se prosigue a realizar la secuencia Driver-PLC

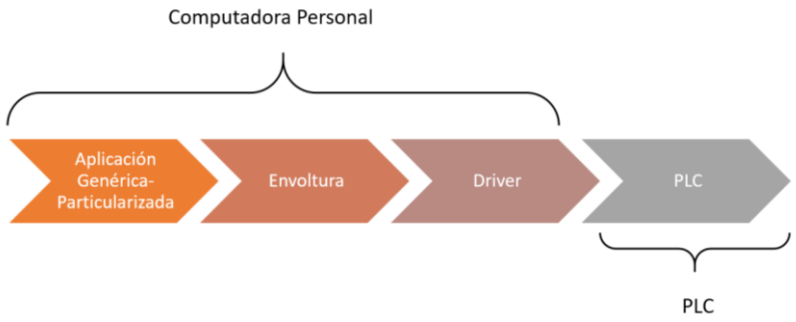


Figura 4.20 Integración PC-PLC

Para dicha integración el OPC permite la conexión con el PLC por medio de la PC, el OPC tiene como función apoyar la comunicación entre la aplicación informática operando en la PC y el programa ejecutándose en el PLC lo que permite que se pueda recibir y/o enviar acciones al PLC, en donde se puede observar el estatus de una entrada o salida específica, así como también modificar el estatus de estas.

4.4.1 OPC

EL OPC (OLE for Process Control), se conforma por dos partes el cliente y el servidor en donde el servidor OPC es el encargado de realizar el enlace correspondiente con el PLC y el cliente desarrolla la interfaz para la comunicación con este como se muestra en la figura 4.21.

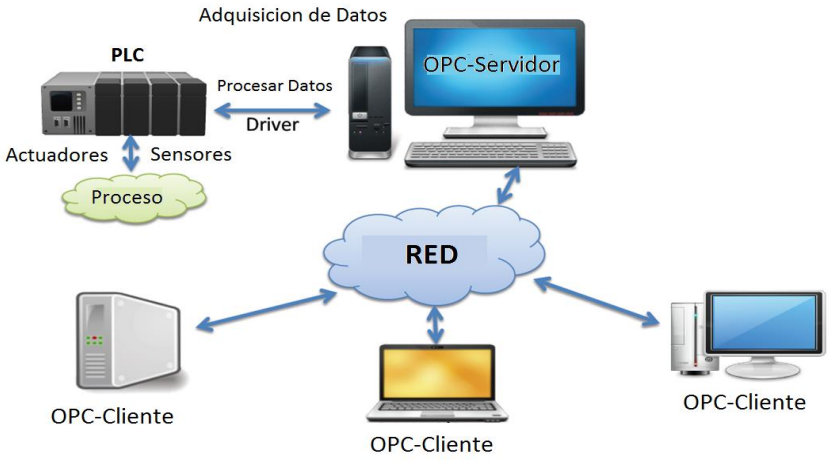


Figura 4.21 Funcionamiento de un OPC

Una comunicación OPC se puede realizar exclusivamente en una PC por lo cual este requiere ciertas aplicaciones para realizarse como son un Servidor y un Cliente, en donde el servidor

es aquel en el cual se encuentran los datos del programa a monitorear y el cliente es en el cual se monitorea y despliegan las variables del programa.

La comunicación entre Servidor OPC – Cliente OPC, son independientes al PLC utilizado, esto da la flexibilidad al sistema, el cual no depende de qué tipo de PLC, ni de qué tipo de comunicación se tiene hacia el PLC, solamente el servidor depende del fabricante del PLC, en cuanto al Cliente no existe variación respecto al fabricante del PLC ya que siempre se comunica a un servidor OPC. Los servidores OPC ofrecen diferentes medios de comunicación los cuales pueden ser por mencionar algunos RS 232, Ethernet, Wifi entre otros, para adaptarse a las necesidades del sistema.

Existen varios SERVIDORES OPC, como son (figura 4.22):

- RsLinx OPC
- Matrikon OPC
- Kepserverex OPC

Figura 4.22 Programas utilizados de OPC libre acceso.

Teniendo en cuenta que cada uno es una configuración distinta, pues no se puede migrar un archivo creado en RsLinx a Matrikon ni a Kepserverex y viceversa.

4.4.2 RsLinx

Para la creación de un Servidor OPC con RsLinx es necesario contar con un PLC físico, pues este no tiene la capacidad de simulación, cuenta con una interfaz muy simple y funcional.

Para la creación de un servidor OPC es necesario tener previamente un programa creado en RSLogix, el cual se encuentre cargado y operando (modo run) en el PLC como se muestra en la Figura 4.23

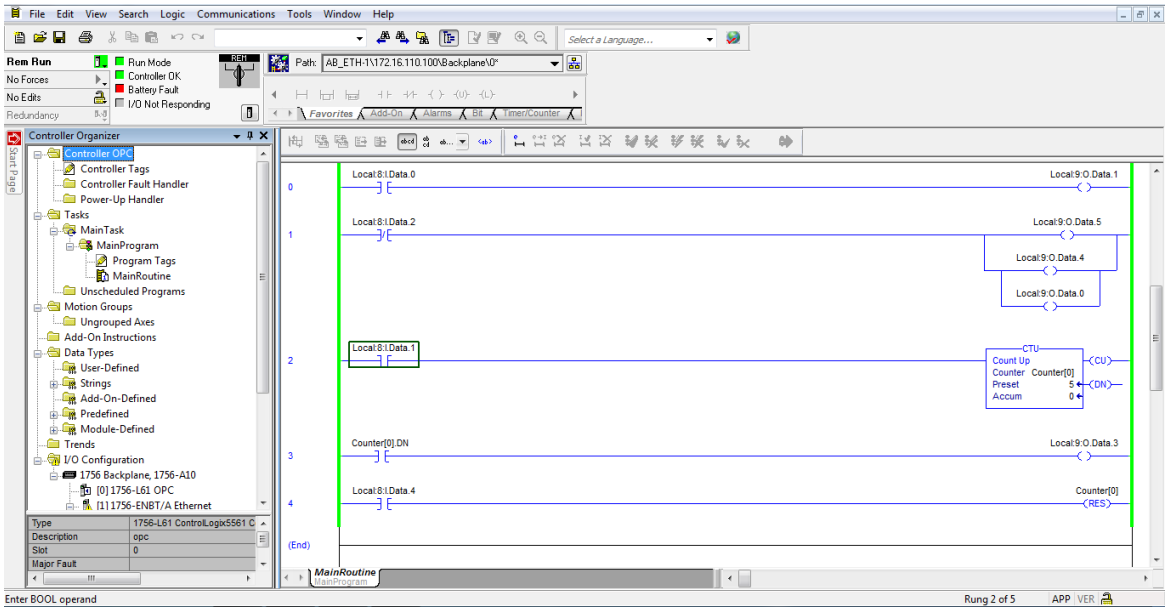


Figura 4.23 Programa en RsLogix5000 cargado al PLC.

Posteriormente en el RsLinx el cual sirve como Servidor en la pestaña de DEE/OPC, seleccionar topic Configuration, Figura 4.24

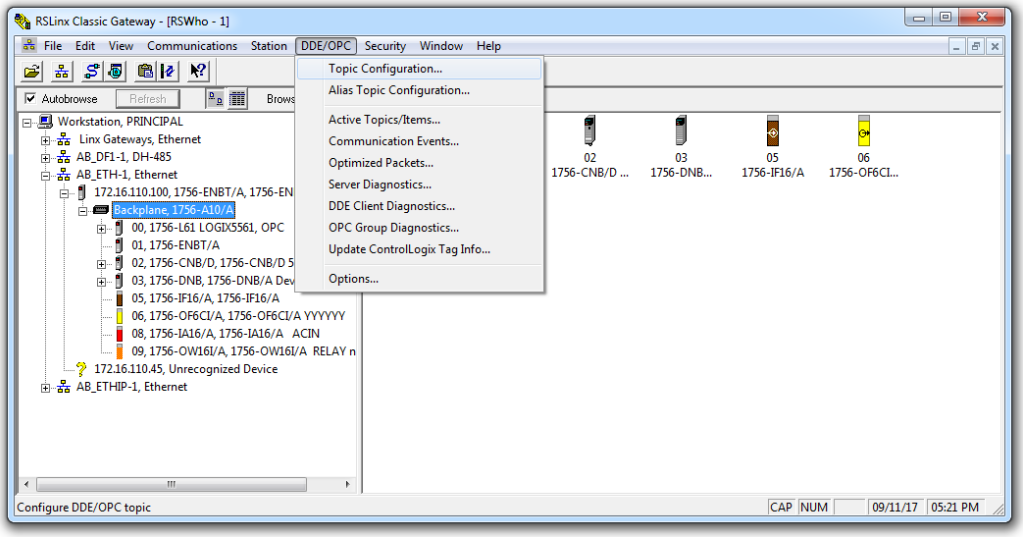


Figura 4.24 RsLinx, Creación del servidor OPC

Esto abrirá una segunda ventana en donde se cargara el programa deseado en este caso el programa que se cargó con anterioridad al PLC y que se encuentra en modo RUN, al crear con éxito el Servidor OPC este mostrara un candado con el cual indica que se creó satisfactoriamente y está en uso, figura 4.25

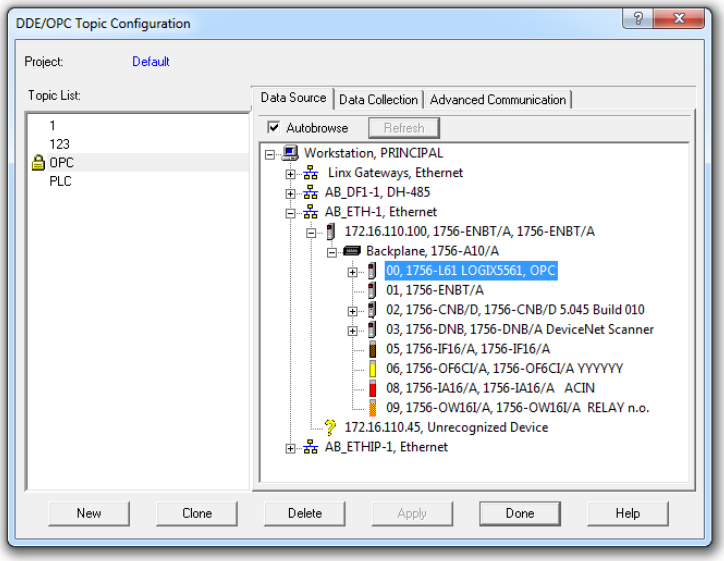


Figura 4.25 Ventana emergente de RsLinx para OPC

Posteriormente al dar click en el botón Done el servidor del OPC está creado y disponible para su comunicación con clientes OPC. RsLinx ofrece la manera de probar el Servidor OPC creado mediante un OPC Test Client, el cual se encuentra en la misma carpeta que contiene el RsLinx, pues es parte del contenido del mismo paquete en la figura 4.26 se muestra el icono correspondiente, posteriormente se abre la pantalla principal como se muestra en la figura 4.27 y se selecciona el tipo de Servidor que se tiene y con ello se enlaza a él, creando el cliente apropiado para ese tipo de servidor.

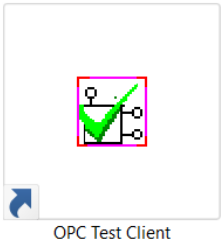


Figura 4.26 Icono del OPC Test Client del RsLinx

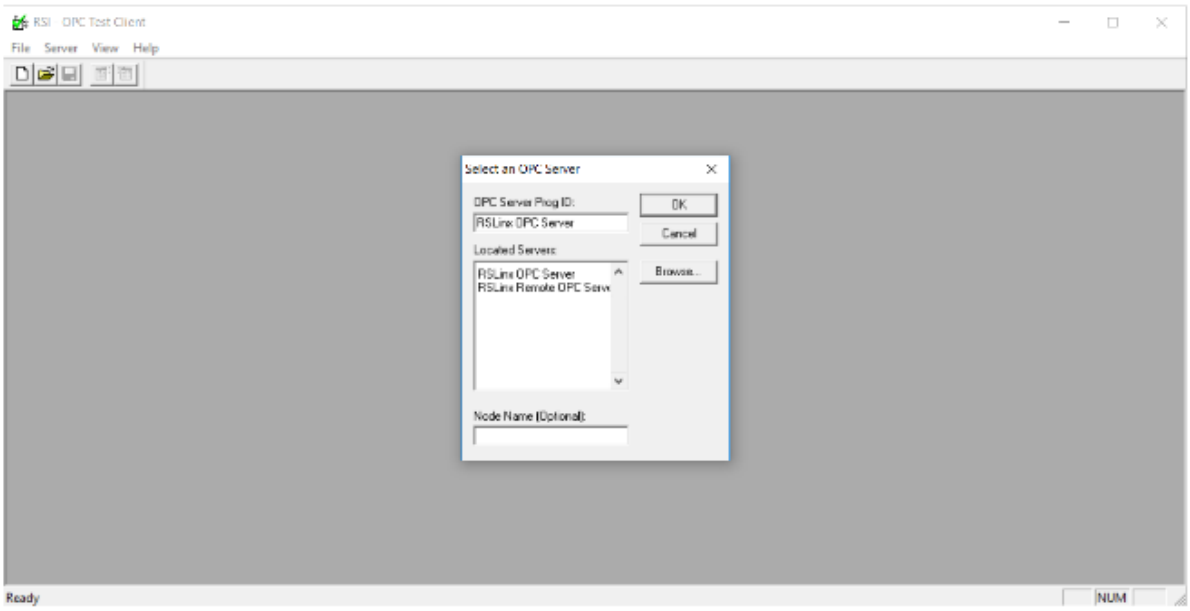


Figura 4.27 Caratula principal del OPC Test Client

Posteriormente se da nombre al grupo de OPC y se da click en OK y con ello el grupo correspondiente esta creado, pueden ser "n" cantidad de grupos en un mismo cliente OPC figura 4.28.

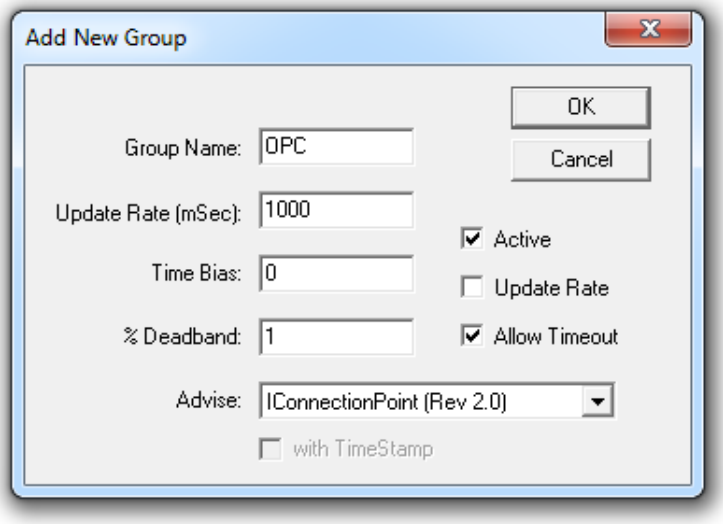


Figura 4.28 Ingreso de nombre del grupo (correspondiente al cliente) en el servidor OPC

La creación de los ítems correspondientes al grupo creado ser realiza en la pestaña de ítems en donde se selecciona Add Ítem, como se muestra en la figura 4.29

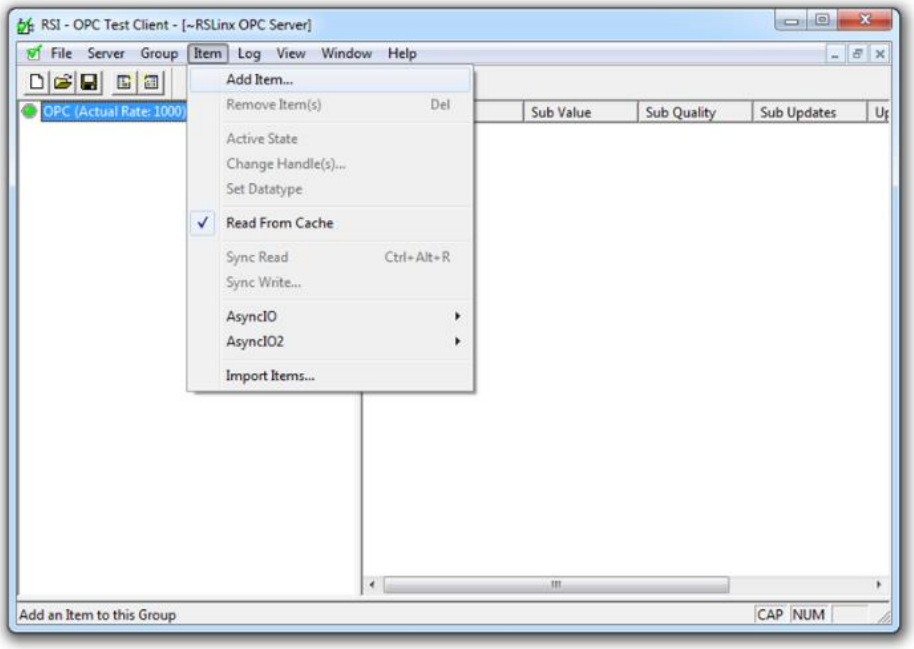


Figura 4.29 Propiedad para añadir Ítems

Esto da la pauta para que abra una ventana en donde se agregan todas las variables a monitorear, en modo Online, se selecciona la variable y se da Add ítem, cuando todas las variables deseadas están agregadas se da valídate y se prosigue a dar click en OK, figura 4.30

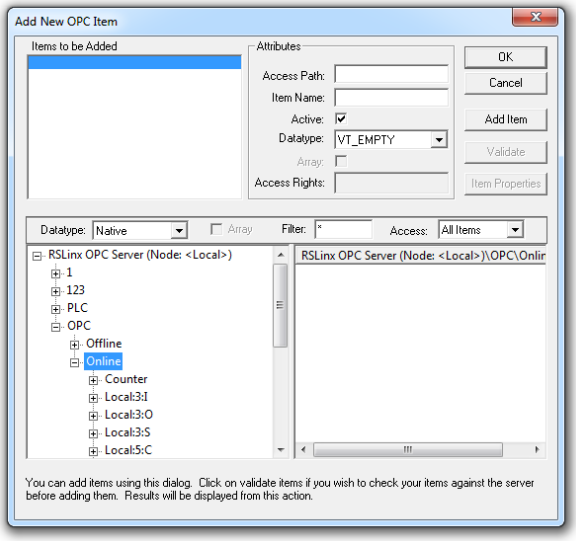


Figura 4.30 Ventana emergente para añadir ítems al servidor OPC en modo online

Este es el último paso de la creación de un SERVIDOR OPC en RsLinx, posteriormente a eso se regresa a la ventana de Cliente en donde se muestra el estatus de cada variable, así

como su nombre, esto se puede guardar para no tener que volver a realizar estos pasos cada vez que se cierre dicha ventana y que se quiera monitorear las mismas variables como se muestra en la figura 4.31.

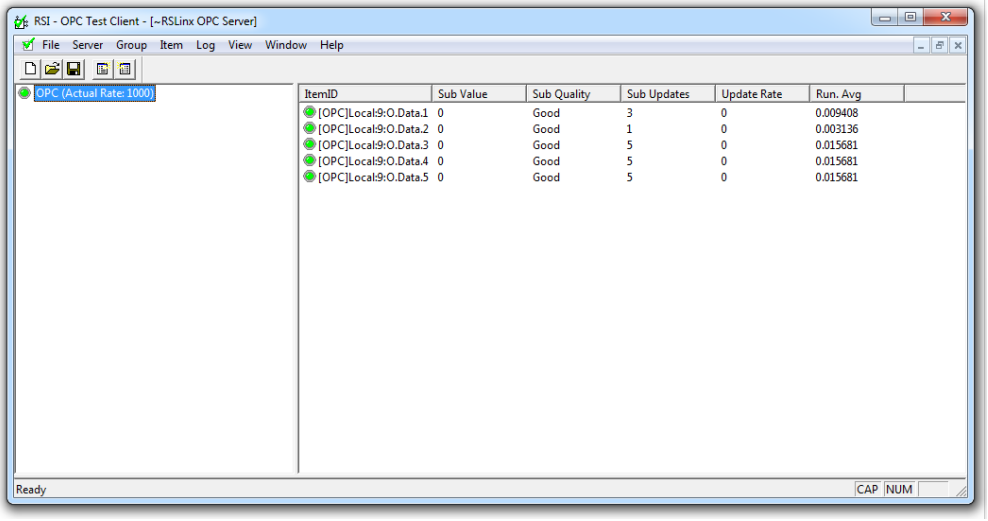


Figura 4.31 Cliente OPC creado con RsLinx en funcionamiento.

Teniendo con ello un Servidor OPC por medio del RsLinx y un Cliente OPC por medio del programa OPC Test Client perteneciente a la mismo Paquete del RsLinx

4.4.3 Matrikon

Para realizar un Servidor OPC con Matrikon no es necesario contar con un PLC físico, pues este paquete tiene la capacidad de simular uno, pero para ello es necesario descargar desde la página de Matrikon (http://www.matrikonopc.com/downloads/index.aspx), el MobileOPC Explorer el cual es el Cliente y el Matrikon OPC Server for AllenBradleyPLCs

Para realizar el Servidor OPC virtual o de simulación es necesario abrir el OPC Explorer, (Figura 4.32) en él se da click en conect el cual hace la conexión con este nodo, pues puede haber muchos nodos en este mismo programa.

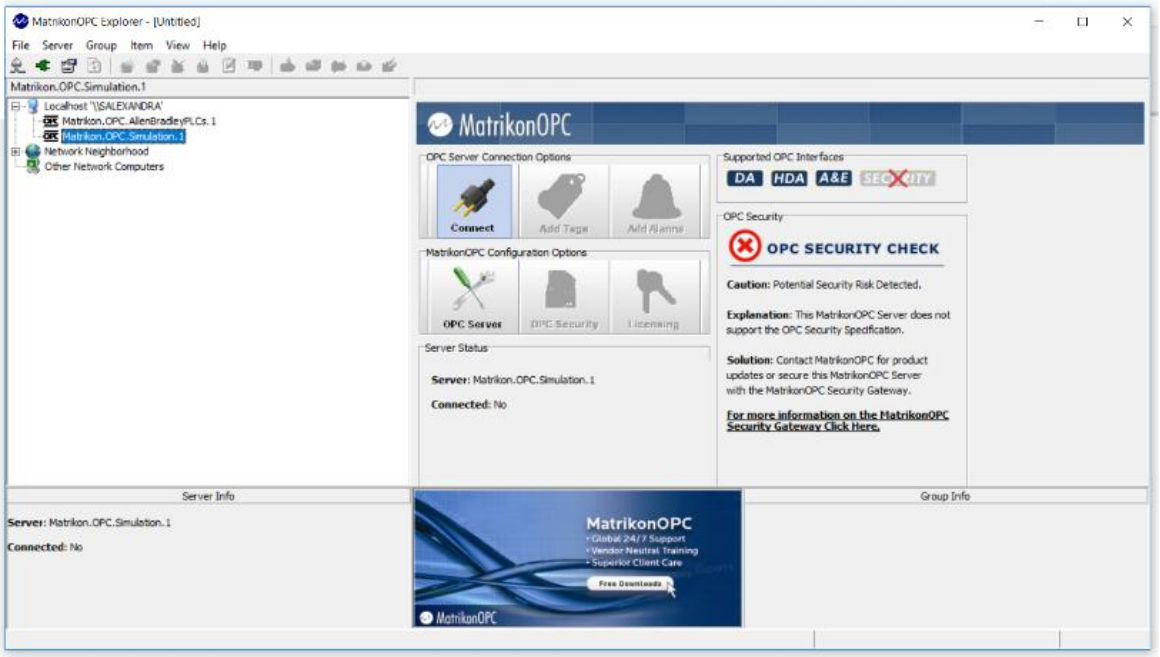


Figura 4.32 MatrikonOPC, caratula principal

Al conectar con el nodo deseado se activan las casillas de add Tags y Add Alarm, OPC Server, en donde en la opción Add Tags es donde se agregaran las variables a monitorear. Figura 4.33

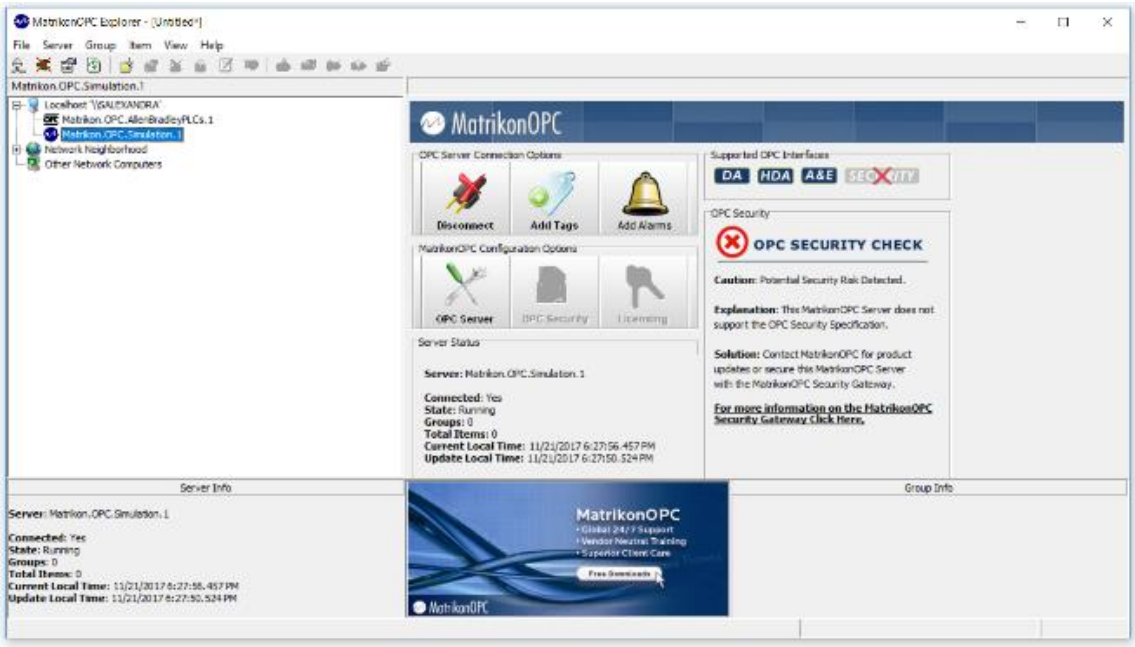


Figura 4.33 Conexión en Matrikon

Posteriormente a esto se abre una ventana en donde se selecciona las variables a agregar, en este ejemplo como son variables de simulación se agregaron variables tipo Random, en donde al darle click izquierdo a Random se muestra la opción de add all ítems como se muestra en la figura 4.34

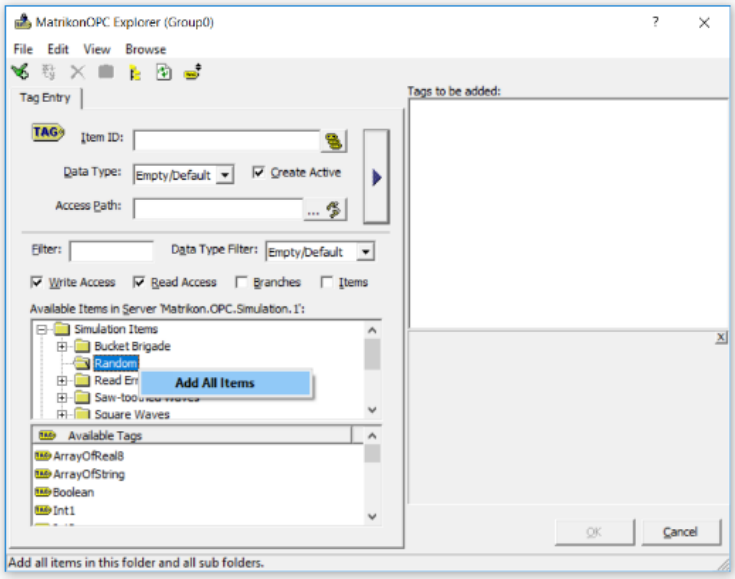


Figura 4.34 Ventana emergente para añadir Ítems al Servidor OPC en Matrikon

Posteriormente a esto se validan los ítems y se da por finalizado el proceso de agregar variables de monitoreo, Figura 4.35

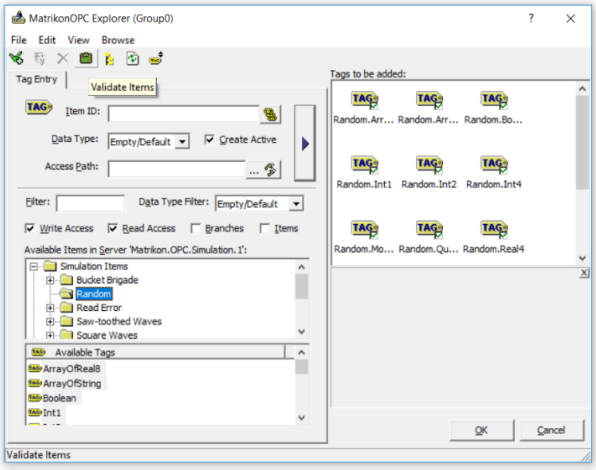


Figura 4.35 Selección de ítems y validación de ellos.

Estas variables se muestran en OPC Explorer el cual es el Cliente OPC y se muestra el valor y el estado en el que se encuentra, así como el nombre de cada una de las variables a monitorear, figura 4.36.

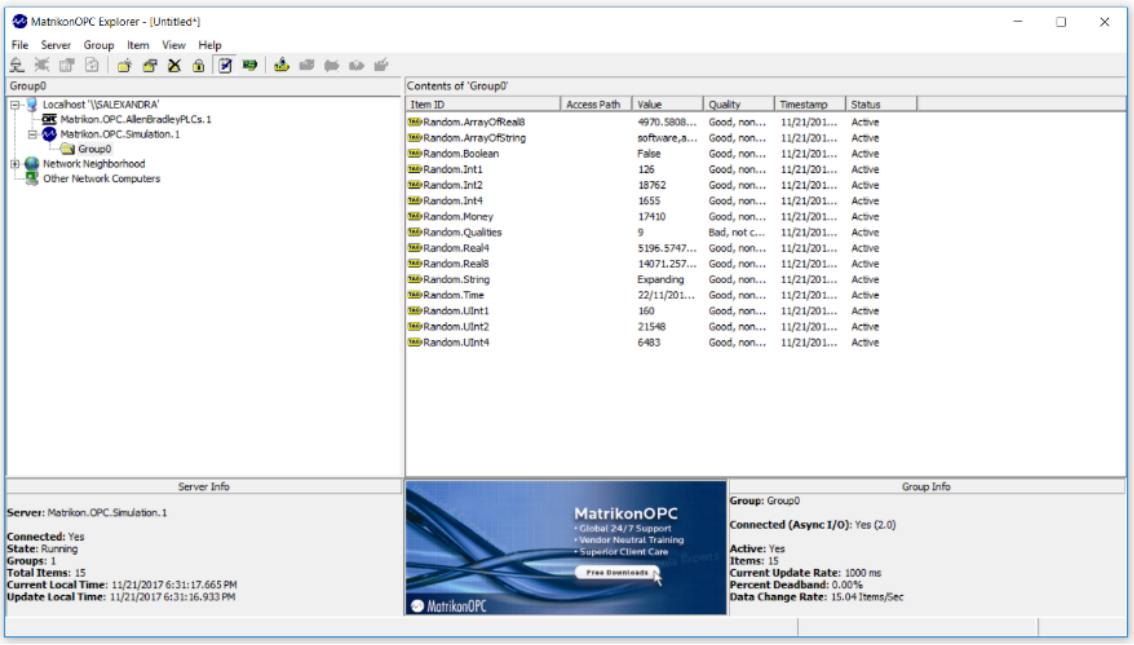


Figura 4.36 OPC en Matrikon Funcionando correctamente.

4.4.4 Kepserverex

Para realizar un Servidor OPC con Kepserverex, al igual que con Matrikon no es necesario contar con un PLC físico, pues este paquete tiene la capacidad de simular uno, pero para ello es necesario descargar desde la página de Kepserverex (https://my.kepware.com/download/demo/ex/).

En KEPServerEX abre una ventana más simple que los anteriores en donde al seleccionar el proyecto el cual es el servidor de OPC, al darle cick en Quick Client, nos traspasa directamente al Cliente OPC, como se muestra en la figura 4.37

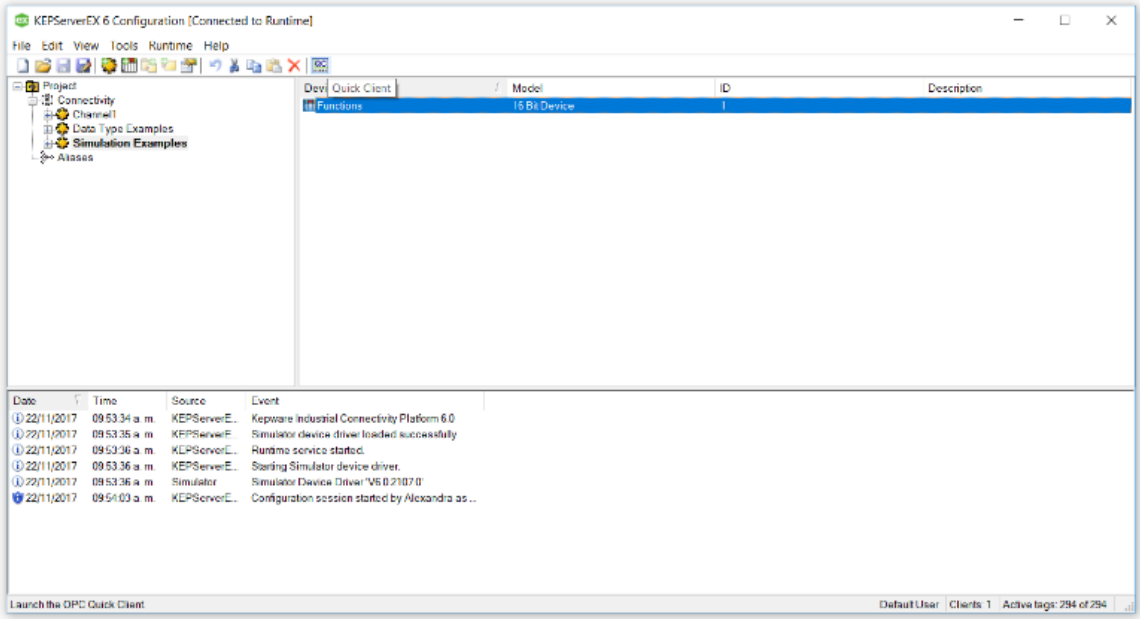


Figura 4.37 Conexión de OPC en Kepserverex

El Kepserverex predeterminadamente contiene cargados y verificados todos los Tags disponibles para la su utilización en manera de simulación, las cuales se encuentran disponibles en modo de carpetas, como se muestra en la figura 4.38 aquí es donde se selecciona el tipo de Tag virtual a utilizar.

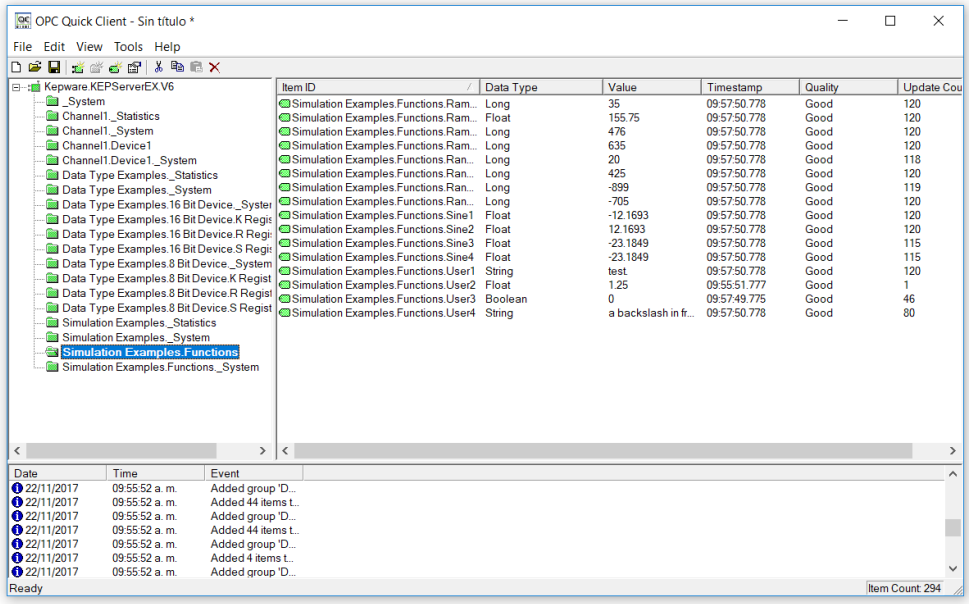


Figura 4.38 Selección de ítems y validación en Kepserverex

Este programa es más simple pero cuenta con limitaciones, una de las cuales y más importantes para el desarrollo de la tesis es que no es compatible con versiones actuales de Visual Studio, el Kepserverex es compatible hasta visual studio 2013, por lo tanto este programa no se utilizó por dicha limitante.

4.4.5 Comunicación Visual Studio – OPC

Para crear una comunicación bidireccional con cualquier servidor de OPC recordando que el Kepserverex solo hasta la versión 2013 de visual studio, dado que las actualizaciones del visual estudio no son compatibles con la versión actual del Kepserverex, esto ha provocado que no se puedan migrar a la versión más reciente y por lo tanto han quedado obsoletas y se ha tenido que cambiar de servidor OPC, por lo tanto es necesario contar con un servidor que cumpla con ciertos lineamientos.

En el programa Visual Studio se realiza la conexión con el servidor OPC al seleccionar cual es el servidor a utilizar en el cual ya se encuentre previamente realizado el OPC y conectado al PLC. Es necesario realizar un Cliente OPC en Visual Stuido como se muestra en la figura 4.39 independientemente de que se haya realizado alguno en otro paquete computacional, con ello nos permite tener acceso a todas las variables ingresadas en el Servidor OPC seleccionado.

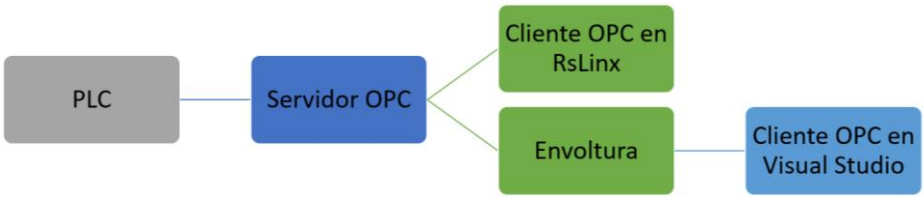


Figura 4.39 Comunicación PLC a OPC en Visual Studio

Para que esta conexión sea exitosa es necesario realizar ciertas especificaciones previamente como son:

4.4.5.1 Verificar si se encuentra conectado previamente.

4.4.5.2 Determinar el Servidor OPC deseado

4.4.5.3 Determinar el grupo al cual conectar

4.4.5.4 Agregar los Ítems

4.4.5.5 Leer Ítems

4.4.5.6 Escribir Ítems

4.4.5.7 Desconectar

4.4.5.1 Verificar si se encuentra conectado previamente.

El cliente OPC que se encuentra en la envoltura, requiere una verificación, en donde se determine si se encuentra en estado conectado, es necesario crear una función la cual verifique este proceso en donde se cuente con un estado conectado, un estado error y un estado conectar, como se muestra a continuación:

```
'Si ya estoy conectado aviso y salgo.
If Conectado Then
   Mensaje = "Error conexión OPC."
   Detalle_Error = "Se ha intentado crear una conexión OPC cuando
                    ya hay una creada."
   Conectar = False
   Exit Function
End If
Try
   Mensaje = "Conectando con el servidor OPC..."
Catch ex As Exception
   Detalle_Error = "Error: " & ex.ToString
   Conectado = False
   Conectar = False
   Exit Function
End Try
Mensaje = "Conexión OPC realizada correctamente."
```

```
    Detalle_Error = ""
    Conectado = True
    Conectar = True
End Function
```

4.4.5.2 Determinar el servidor OPC deseado

Para determinar el servidor de OPC en la envoltura es necesario conocer ciertos parámetros, entre ellos es el tipo y nombre del servidor OPC que se tenga creado y enlazado con el PLC que se va a utilizar, pues se pueden tener “n” numero de servidores y “n” numero de PLC disponibles, con ello la envoltura determina el lineamiento necesario para la conexión entre ellos.

En la envoltura es necesario que contenga el código correspondiente para la selección del Servidor como se muestra a continuación.

```
Mensaje = "Conectando con el servidor OPC..."
ServidorOPC = New OPCServer
ServidorOPC.Connect("Kepware.KEPServerEX.V6") 'para Kepserverex
ServidorOPC.Connect("RSLinx OPC Server") 'para RsLinx
ServidorOPC.Connect("Matrikon.OPC.Simulation.1") 'para Matrikon
```

4.4.5.3 Determinar el grupo al cual conectar

Para crear una comunicación con los grupos de OPC, es necesario crear una instanciación en donde se agrega los grupos del servidor, por lo general todos los grupos se llaman grupo y empiezan desde el número 1, pero esto puede variar según el programa, para añadir los grupos en visual studio se hace de la siguiente manera independientemente de que servidor de OPC se seleccionó pues esta opción ya está previamente cargada en Servidor OPC.

```
Mensaje = "Añadiendo grupo al servidor OPC..."
GruposOPC = ServidorOPC.OPCGroups
GrupoOPC = GruposOPC.Add("Grupo1")
```

```
GrupoOPC.IsActive = True
GrupoOPC.UpdateRate = 1000
GrupoOPC.IsSubscribed = True
```

4.4.5.4 Agregar los ítems

Para añadir los ítems es igual en cada uno solo se tiene que seguir este lineamiento después de add Ítem ("[nombre del programa].nombre de la variable", Numero), en donde el número es el mismo que el ítem agregado, empezando siempre desde Cero.

```
Mensaje = "Añadiendo Ítems al grupo..."
    ReDim ItemOPC(100)  'Dimensionar según las necesidades
    'Introducir un ítem por cada variable del PLC en la que queramos leer o escribir
    'A cada ítem le asignamos un número, que debemos recordar para referirnos a él en el programa
    'Kepserverex
        ItemOPC(0) = GrupoOPC.OPCItems.AddItem("Simulation
                Examples.Functions.Random1", 0)
        ItemOPC(1) = GrupoOPC.OPCItems.AddItem("Simulation
                Examples.Functions.User3", 1)
    ItemOPC(2) = GrupoOPC.OPCItems.AddItem("Simulation
                Examples.Functions.User1", 2)
    'matrikon
        ItemOPC(0) = GrupoOPC.OPCItems.AddItem("Nombre de la variable", 0)
        ItemOPC(1) = GrupoOPC.OPCItems.AddItem("Nombre de la variable ", 1)
    'rs linx
    ItemOPC(0) = GrupoOPC.OPCItems.AddItem("[nombre del programa]nombre
                de la variable", 0)
```

También es posible agregar ítems desde el programa en ejecución esto por medio de una función en la cual se agrega el ítem a la lista de ítems como se muestra a continuación.

```
Public Function añadiritem(ByVal A As String, ByVal Tag As String) As String
    ItemOPC(A) = GrupoOPC.OPCItems.AddItem(Tag, 1 + 1)
      'se agrega un ítem y queda grabado en el número correspondiente
  End Function
```

4.4.5.5 Leer ítems

Para realizar una lectura del servidor de OPC, es necesario crear una función en donde especifiquemos todos los parámetros a cumplir para realizar esta acción en donde en la función LeerItemInt se manda un valor de índice el cual es el ítem solicitado a leer como se muestra a continuación.

```
  'Función para leer un ítem que representa una variable entera
  'Se le pasa el índice del ítem que se va a leer
  'Devuelve el valor de la variable
  Public Function LeerItemInt(ByVal Indice) As Integer
    Dim Valor As Object = Nothing
    Dim Calidad As Object = Nothing
    Dim TimeStamp As Object = Nothing
    If Not Conectado Then
      Mensaje = "Error conexión OPC."
      Detalle_Error = "No hay establecida una conexión OPC."
      LeerItemInt = 0
      Exit Function
    End If
    Try
      ItemOPC(Indice).Read(OPCDataSource.OPCDevice, Valor, Calidad, TimeStamp)
      LeerItemInt = CInt(Valor.ToString)
    Catch ex As Exception
      Detalle_Error = ex.ToString
      Mensaje = "¡Error al leer Item! [Int, Índice " & Indice & "]"
      LeerItemInt = 0
```

```
            Exit Function
        End Try
        Mensaje = ""
        Detalle_Error = ""
    End Function
```

4.4.5.6 Escribir ítems

Para escribir en el servidor de OPC, es necesario crear una función en donde especifiquemos todos los parámetros a cumplir, en este caso solo se pueden escribir valores enteros, se selecciona el ítem a modificar al ingresarle un valor a índice y se modifica por medio de agregarle un valor a entero como se muestra a continuación.

```
    'Función para escribir en un ítem que representa una variable entera
    'Se le pasa el índice del ítem y el valor que se va a escribir
    'Devuelve True si todo es correcto

    Public Function EscribirItemInt(ByVal Indice As Integer, ByVal Entero As Integer) As
Boolean
        Dim Dims() As Integer = New Integer() {1}
        Dim Bounds() As Integer = New Integer() {1}
        Dim Serverhandles As Array = Array.CreateInstance(GetType(Integer), Dims,
Bounds)
        Dim Errores As Array = Array.CreateInstance(GetType(Integer), Dims, Bounds)
        Dim Valores As Array = Array.CreateInstance(GetType(Object), Dims, Bounds)
        If Not Conectado Then
            Mensaje = "Error conexión OPC."
            Detalle_Error = "No hay establecida una conexión OPC."
            EscribirItemInt = False
            Exit Function
        End If
        Try
```

```
            Serverhandles.SetValue(ItemOPC(Indice).ServerHandle, 1)
            Errores.SetValue(0, 1)
            Valores.SetValue(Entero, 1)
            GrupoOPC.SyncWrite(1, Serverhandles, Valores, Errores)
        Catch ex As Exception
            Detalle_Error = ex.ToString
            Mensaje = "¡Error al escribir Item! [Int, Índice " & Indice & "]"
            EscribirItemInt = False
            Exit Function
        End Try
        Mensaje = ""
        Detalle_Error = ""
        EscribirItemInt = True
    End Function
```

4.4.5.7 Desconectar

Para desconectar el servidor de OPC, al igual que se crea una función para conectar se crea una función para desconectar, en donde se remueve todo lo añadido al programa y se desconecta de él como se muestra a continuación.

```
    'Función para deshacer la conexión OPC
    Public Function Desconectar() As Boolean
        Try
            Mensaje = "Desconectando..."
            ItemOPC = Nothing
            If Not IsNothing(ServidorOPC) Then
                ServidorOPC.OPCGroups.RemoveAll()
                ServidorOPC.Disconnect()
                ServidorOPC = Nothing
            End If
            GrupoOPC = Nothing
```

```
            GruposOPC = Nothing
        Catch ex As Exception
            Detalle_Error = "Error: " & ex.ToString
            Desconectar = False
            Exit Function
        End Try
        Mensaje = "Desconexión realizada correctamente."
        Detalle_Error = ""
        Conectado = False
        Desconectar = True
    End Function
```

4.5 ARQUITECTURA DE REFERENCIA

La Arquitectura de Referencia al ser el primer nivel de abstracción en esquema de referencia ArquiTAM es el nivel conceptual donde se identificaron las herramientas a utilizar como el iMRP, el acoplamiento débil, la operación dirigida por modelo, así como un análisis de lo actualmente incorporado a la industria en base a células robotizadas y manejo de PLC. Otro aspecto de la investigación consistió en identificar los elementos de una célula robotizada típica, Figura 4.40

Debido a su complejidad el problema se delimitó al control secuencial del sistema a nivel Sensores/Actuadores, siendo estos una parte esencial en una célula robotizada.

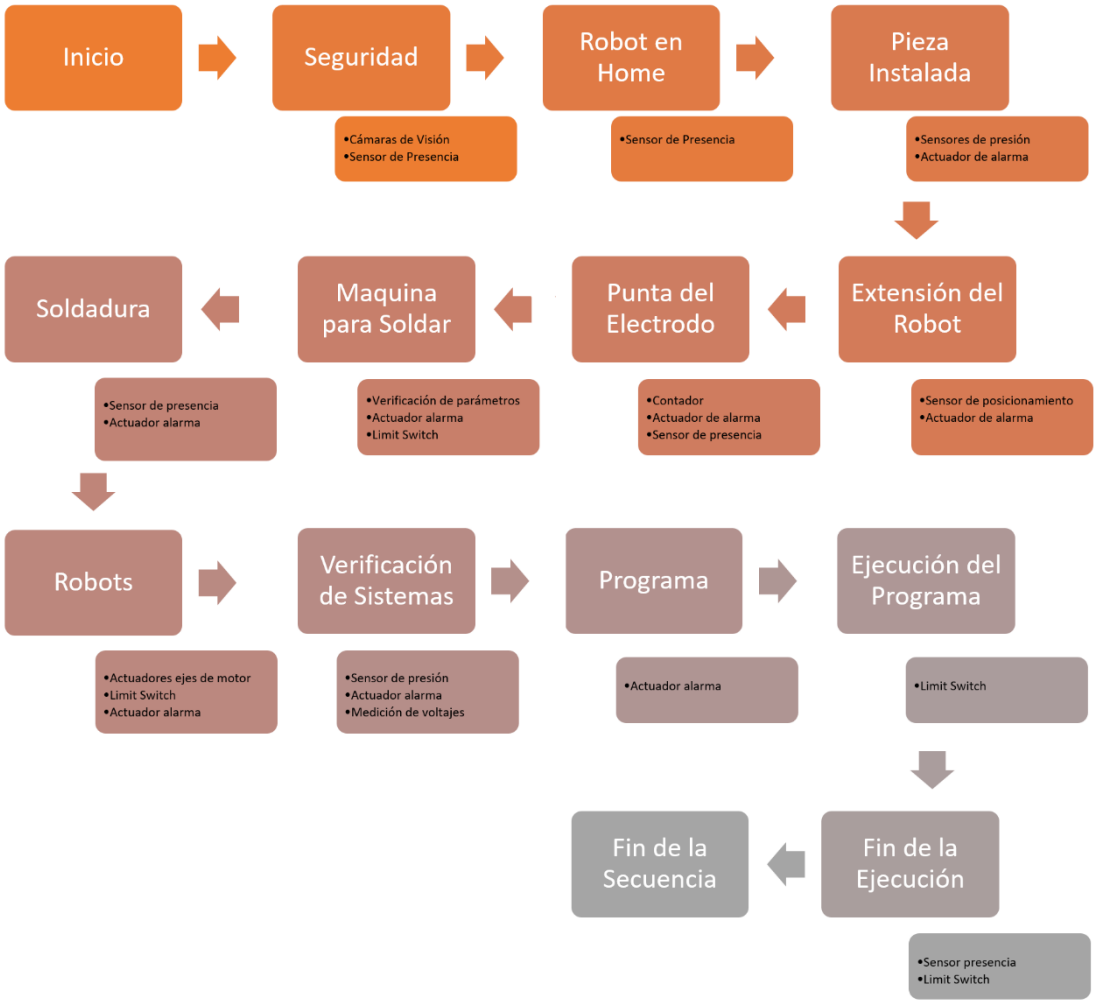


Figura 4.40 Esquema de bloques descriptivos de una célula robotizada.

4.6 MODELO DE REFERENCIA

En el modelo de referencia se plantea realizar una integración de programas realizados en Visual Studio los cuales son el "Editor de iMRP", (Gonzales, 2017), "Aplicación del Paradigma Orientado a Objetos en la Reusabilidad de Código en Controladores Lógicos Programables" (Rodríguez, 2017), "Lanzador de Ordenes iMRP" (2016), con la flexibilidad que cada uno plantea.

4.6.1 Editor iMRP

El editor iMRP es una herramienta la cual permite la creación de grafos iMRP, estos son almacenados en una base de datos de MySQL o bien pueden ser exportados/importados desde un archivo XML (Figura 4.41)

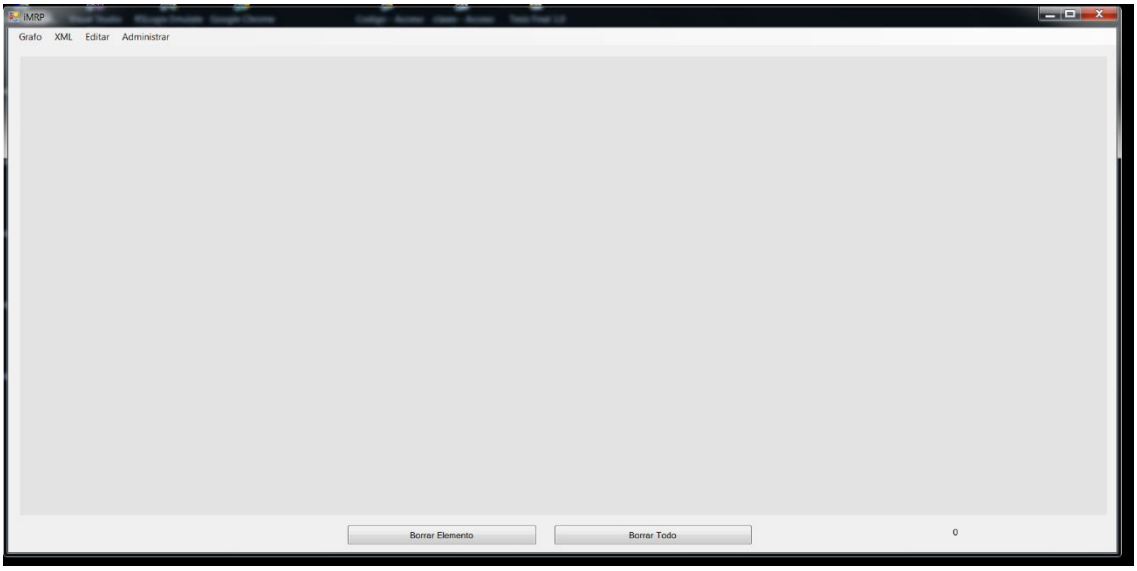


Figura 4.41Ventana emergente del Grafo iMRP.

Esta herramienta permite la creación, modificación y eliminación de grafos iMRP almacenados en la base de datos.

4.6.2 Lanzador de ordenes iMRP

El lanzador de ordenes iMRP es una herramienta la cual se enlaza a la base de datos de MySQL que se utilizó en el editor iMRP, desde él se accede y selecciona el grafo a utilizar, o bien se puede abrir un archivo XML previamente creado en el editor y utilizarse en el lanzador sin la necesidad de la base de datos.

En él se muestra en el apartado de Nodos todos nodos del grafo así como su contenido, en el apartado de Arcos se muestra su contenido así como su ID (Figura 4.42).

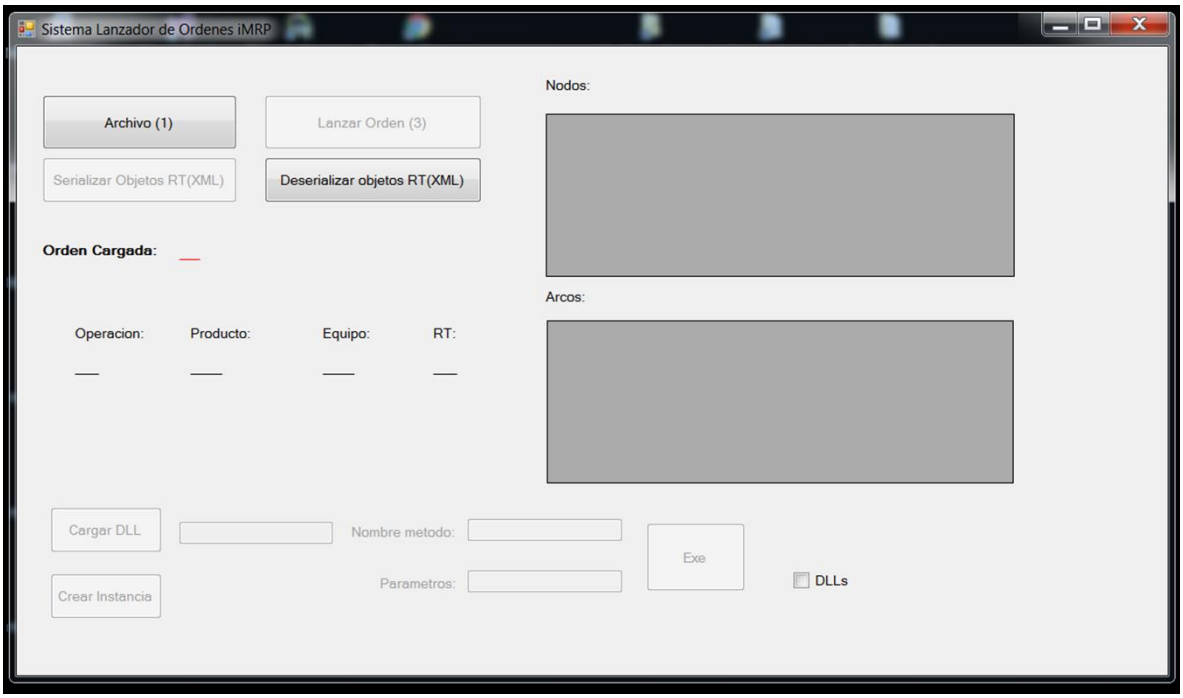


Figura 4.42 Ventana emergente del Sistema lanzador de Ordenes iMRP

4.6.3 Implementación del concepto de acoplamiento débil.

En el acoplamiento débil se encuentran tres objetos, el representante, la envoltura (figura 4.45)y el driver en donde la envoltura permite la interacción entre el representante y el Driver que es propio del equipo, por lo tanto la envoltura es el camino para acceder al equipo, como se muestra en la figura 4.43, en el sistema informático se encuentra incorporado el Servidor OPC, mientras tanto en la envoltura se encuentra incorporado el Cliente OPC, en donde no se tuvo la necesidad de tener un representante como se muestra en la figura 4.44.

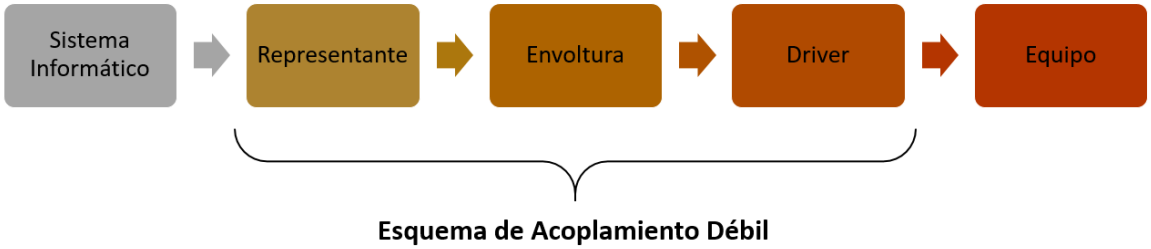


Figura 4.43 Representación inicial del esquema de acoplamiento débil

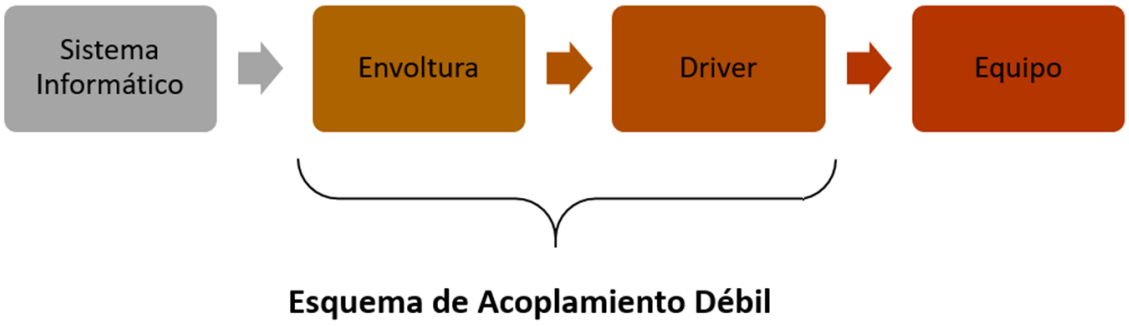


Figura 4.44 Representación final del esquema de acoplamiento débil

Figura 4.45 Envoltura

4.6.4. Instanciación del Modelo Particular

El modelo particular es generado a partir de la instanciación del modelo de particularidades iMRP, en donde a su vez el modelo particular instancia código del PLC, dicho código es instanciado a partir de las clases las cuales son identificadas en el modelo de iMRP, a su vez estas clases son implementadas en el modelo de referencia (o framework), esto conlleva a que el modelo particular es implementado y operado basado en el modelo iMRP.

Dado que la instanciación del código del PLC es proveniente de la abstracción de particularidades del iMRP, posteriormente es necesario realizar una inserción de código, correspondiente a todas las particularidades de código abstraídas.
A continuación se describe la manera de generar el código del PLC correspondiente al sistema particular o específico.

4.6.5 Inserción de Código

El proceso de inserción de código al controlador lógico programable se basa en la Aplicación del Paradigma Orientado a Objetos. En el entorno de desarrollo Studio 5000 para PLC´s Allen Bradley (Rockwell), se cuenta con la característica de escribir el código del programa en formato texto tipo ".L5K".

El programa RsLogix5000 el cual es de la familia Rockwell permite la creación de código tipo escalera de varias maneras una es creando un archivo nuevo, otra es accediendo a un archivo guardado el cual puede ser de formato ".ACD" o ".L5K", esta aplicación utiliza el formato "L5K" para crear código en formato texto que posteriormente se convertirá a formato escalera para ser utilizado en el PLC.

En esta aplicación cuenta con las opciones de cargar un archivo "L5K", salvar los cambios creados en formato "L5K", insertar Tags, insertar código, así como crear líneas, sensores, encoders, piso, detector, actuador, IU y control.(Figura 4.46)

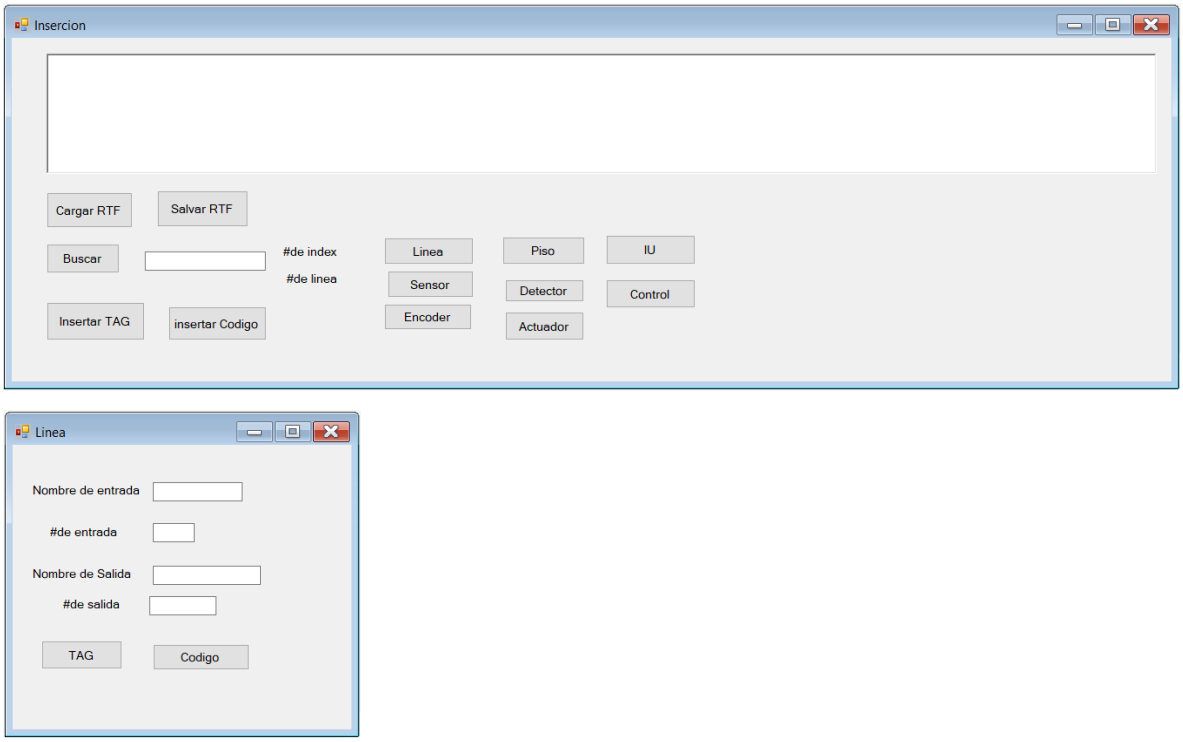


Figura 4.46 Inserción de código, Ventana emergente de inserción de línea

4.6.6 Adaptaciones

Para la integración de las aplicaciones particulares e independientes como son el "Editor de iMRP", (Gonzales, 2017), "Aplicación del Paradigma Orientado a Objetos en la Reusabilidad de Código en Controladores Lógicos Programables" (Rodríguez, 2017), "Lanzador de Ordenes iMRP" (2016), fue necesario realizar una adaptación y modificación al código existente en donde al anexarle ciertas especificaciones y modificar otras con ello se realizó correctamente la integración.

4.6.7 Adaptación del Lanzador de Ordenes iMRP

El lanzador de órdenes de iMRP, como se mencionó anteriormente solo muestra el contenido en los nodos y arcos pero no muestra ni determina a quien pertenece el arco y a donde va, por lo tanto se realizó esta modificación.

- **Mandar variables a la envoltura:**

  Se creó una función en donde se permite almacenar los valores de los arcos, y posteriormente ser mandados a la envoltura.

```
Public Sub manda_lanzador_arcos(ByRef var1, ByRef var2, ByRef var3, ByRef var4,
ByRef var5, ByRef var6, ByRef var7, ByRef var8)
    'variables de los arcos añadidos
    var1 = SVar1
    var2 = SVar2
    var3 = SVar3
    var4 = SVar4
    var5 = SVar5
    var6 = SVar6
    var7 = SVar7
    var8 = SVar8

     End Sub
```

- **Identificación de variables:**

  Se anexo el código correspondiente para la identificación de las variables en los arcos y nodos, asi como determinar de dónde proceden, a donde van y su contenido

```
Dim LisNodAr As New Arco()
        LisNodAr.var1 = Calendarizador.BuscarArcoOperacion(rt, listOp, listaArcos,
listEq)  'me agega a var1 el numero del arco para buscarlo despues (operaciones)
        LisNodAr.var2 = Calendarizador.BuscarArcoEquipos(rt, listOp, listaArcos, listEq)
'me agega var2 el numero del arco para buscarlo despues (equipos)
        Dim ArcOp1 As Arco
        Dim ArcOp2 As Arco
        Dim AñadirVariable As Arco
        ArcOp1 = listaArcos.Find(Function(b) LisNodAr.var1 = b.zOrder) 'busca el arco de
operaciones
        ArcOp2 = listaArcos.Find(Function(b) LisNodAr.var2 = b.zOrder) 'busca el arco de
equipos
```

```
listArc_Nod_Var.Add(ArcOp1) 'agrega el arco de operaciones
listArc_Nod_Var.Add(ArcOp2) 'agrega el arco de equipos
'si son mas de 4 RTS añadir mas if aqui.
'aquí añade la variable v1 y v2 a su rt correspondiente o mas bien a su
        nodo correspondiente.
If listArc_Nod_Var.Count = 2 Then
   AñadirVariable = listArc_Nod_Var.Item(contador)
   SVar1 = AñadirVariable.var1
   TextBox6.Text = SVar1
   contador = contador + 1
   AñadirVariable = listArc_Nod_Var.Item(contador)
   SVar2 = AñadirVariable.var1
   TextBox7.Text = SVar2
   contador = contador + 1
Else
   If listArc_Nod_Var.Count = 4 Then
      AñadirVariable = listArc_Nod_Var.Item(contador)
      SVar3 = AñadirVariable.var1
      TextBox11.Text = SVar3
      contador = contador + 1
      AñadirVariable = listArc_Nod_Var.Item(contador)
      SVar4 = AñadirVariable.var1
      TextBox12.Text = SVar4
      contador = contador + 1
   End If
   If listArc_Nod_Var.Count = 6 Then
      AñadirVariable = listArc_Nod_Var.Item(contador)
      SVar5 = AñadirVariable.var1
      TextBox14.Text = SVar5
      contador = contador + 1
      AñadirVariable = listArc_Nod_Var.Item(contador)
```

```
            SVar6 = AñadirVariable.var1
            TextBox15.Text = SVar6
            contador = contador + 1
        End If
        If listArc_Nod_Var.Count = 8 Then
            AñadirVariable = listArc_Nod_Var.Item(contador)
            SVar7 = AñadirVariable.var1
            TextBox16.Text = SVar7
            contador = contador + 1
            AñadirVariable = listArc_Nod_Var.Item(contador)
            SVar8 = AñadirVariable.var1
            TextBox17.Text = SVar8
            contador = contador + 1
        End If
         End If
```

- **Abrir generador de código:**

  En el lanzador de órdenes de iMRP se añadió la opción de abrir el generador de código o inserción de código en donde se es posible realizar el código en formato texto para su integración al PLC.

```
    Dim sNOmbre As String
    Dim sNNombre As String = "Generador de Codigo"
    OpenFileDialog1.ShowDialog() 'abre dialogo
    sNOmbre = OpenFileDialog1.FileName ' guarda direccion
    objE = System.Reflection.Assembly.LoadFrom(sNOmbre) 'carga dirección a objetoE
    Dim arg As Object() = New Object() {sNNombre} 'aqui se agrega el nombre que va a agarrar el nuevo sub new (_nombre)
    objM = objE.CreateInstance(TextBox18.Text)
    objM.Nombre = "Generador de Codigo"     'el nombre puede ser cualquiera
```

```
objM.Propietario = Me
codigoenviar()
objM.enviarcodigo(eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9)
```

- **Mandar a operar la envoltura y los valores correspondientes:**

En esta sección se manda a opera la envoltura correspondiente, asi como se envían los valores correspondientes del arco al que pertenece el RT actual.

```
        Dim arg3 As Object() = {ArcOFF, ArcON}
        Equipo.[GetType]().GetMethod("Operar").Invoke(Equipo, arg)  'aqui se va a la
rutina operar de la tolva/banda/etc
Public Sub variables_acrco_actual(arc_act_var1 As String, arc_act_var2 As String)
    ArcOFF = arc_act_var1
    ArcON = arc_act_var2
     End Sub
```

- **Enviar dirección de documentos:**

En esta sección se describe la manera de implementar el envio de la ruta de los documentos que contienen el código a anexar en la aplicación de inserción, esta dirección es proporcionada por las envolturas y posteriormente son enviadas a la aplicación de inserción.

```
Public Sub codigoenviar()
    Dim i As Integer = 0
    Do While ListBox1.Items.Count > i
      eq1 = CType(ListBox1.Items(i), String)
      i = i + 1
      If ListBox1.Items.Count = i Then
      Else
        eq2 = CType(ListBox1.Items(i), String)
        i = i + 1
```

```
End If
If ListBox1.Items.Count = i Then
Else
    eq3 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
If ListBox1.Items.Count = i Then
Else
    eq4 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
If ListBox1.Items.Count = i Then
Else
    eq5 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
If ListBox1.Items.Count = i Then
Else
    eq6 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
If ListBox1.Items.Count = i Then
Else
    eq7 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
If ListBox1.Items.Count = i Then
Else
    eq8 = CType(ListBox1.Items(i), String)
    i = i + 1
End If
```

```
        If ListBox1.Items.Count = i Then
        Else
            eq9 = CType(ListBox1.Items(i), String)
            i = i + 1
        End If
    Loop
End Sub
```

Dado los cambios anteriores al Lanzador de ordines iMRP, el resultado final de ellas se muestra en la figura 4.47.

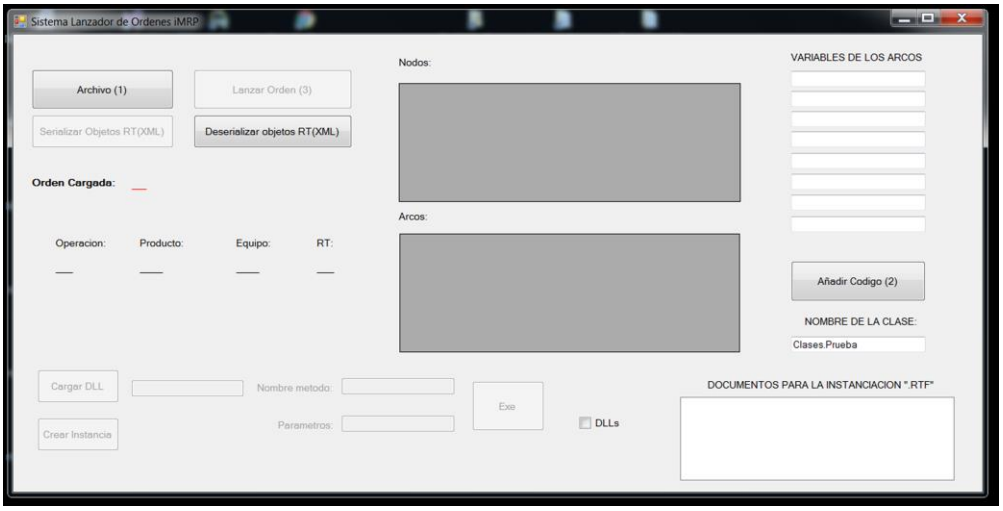


Figura 4.47 Sistema lanzador de ordenes iMRP con modificaciones

4.6.8 Adaptación de Envolturas

- **Estado actual de la envoltura:**

  En esta sección lo que se realizo es mostrar en la envoltura que acción se está realizando ya sea que este en proceso o allá terminado el proceso, también proporciona con cual producto y cual Tag están en utilización.

```
Public Sub Fin_Operacion(TexTag As String, nTag As String)
```

```
Label3.Text = TexTag + nTag
Label6.Text = TexTag + Producto.nombre
Dim s As String = TexTag
Dim sub1 As String = "Fin"
Dim b As Boolean = s.Contains(sub1)
If b Then
   objM.conectando()
   Terminando_con_OPC()
End If
 End Sub
```

- **Creación del Cliente OPC:**

  En esta sección lo que se anexo al código original fue la creación del cliente OPC en donde con ello se enlaza, al servidor de OPC seleccionado previamente en el programa como se mencionó con anterioridad en el la sección 4.3.3.4.2 del presente documento

```
Dim sNOmbre As String
Dim sNNombre As String = nombre
OpenFileDialog1.ShowDialog() 'abre dialogo
sNOmbre = OpenFileDialog1.FileName ' guarda direccion
objE = System.Reflection.Assembly.LoadFrom(sNOmbre) 'carga direccion a objetoE
Dim arg As Object() = New Object() {sNNombre} 'aqui le agregas el nombre que va a agarrar el nuevo sub new (_nombre)
    objM = objE.CreateInstance(txt_clase.Text, False, BindingFlags.[Default] Or BindingFlags.InvokeMethod, Nothing, arg, CultureInfo.CurrentCulture, Nothing)
objM.Nombre = nombre          'el nombre puede ser cualquiera
    objM.Propietario = Me
```

- **Tag de Arcos:**

  Dado que se anexaron rutinas en el lanzador de ordenes una de ellas fue enviar los arcos a la envoltura, esta envoltura lo que realiza es obtener los datos de los arcos y proceder a conectarse con el OPC, posterior mente procede a escribir el valor de “1” en el Tag de ON y con ello iniciar el proceso.

```
Public Sub Operar3(ByRef varoff As String, ByRef varon As String) 'recibe las variables de los arcos del rt actual
    txtOFF.Text = varoff
    txtON.Text = varon
    objM.enviar_Tags(txtOFF.Text, txtON.Text)
    objM.conectando()
    objM.escribir("1")
     End Sub
```

- **Botones Anexos:**

  Los botones anexos son para poder corroborar que el programa esté funcionando correctamente, en dado caso que se cuente con alguna señal de anomalía se puede corroborar por medio de estos botones anexos, en donde uno procede a conectar directamente el OPC y el otro procede a enviar al OPC los Tags para su conexión. (Figura 4.48)

```
    objM.enviar_Tags(txtOFF.Text, txtON.Text)
    objM.conectando()
```

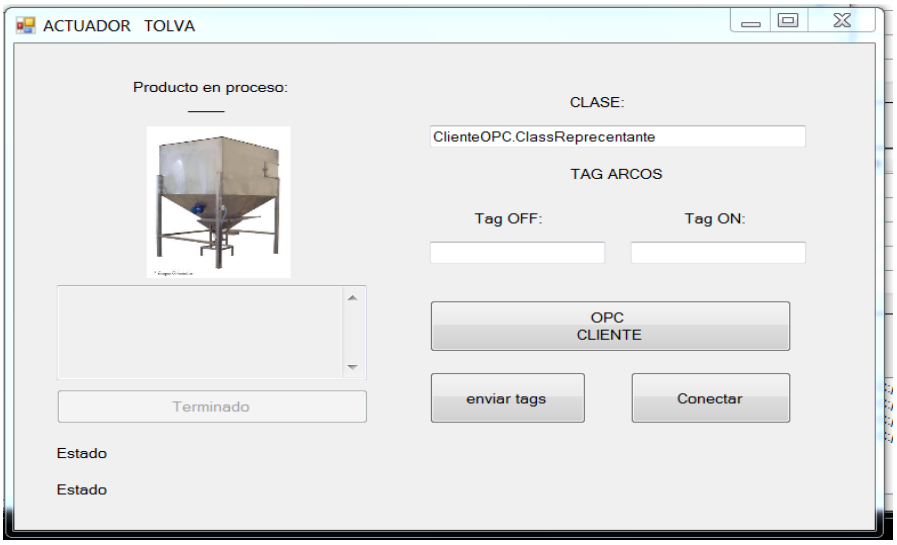


Figura 4.48 Envoltura con modificaciones

4.7 OPC.

Para la interconexión lanzador de ordenes iMRP - Servidor OPC - Cliente OPC - PLC, es necesario tener en cuenta la estructura del sistema, donde es a través del lanzador de órdenes y por medio de la envoltura en donde se realiza la conexión con el servidor OPC y posteriormente es creado un Cliente OPC con el cual se realiza la lectura y escritura de las variables, las cuales se encuentran en el PLC en funcionamiento.

Para esto el lanzador de órdenes iMRP tiene la posibilidad de realizar cambios en el PLC por medio de la envoltura el cual es el camino para su conexión, el lanzador de ordenes iMRP asigna los valores a determinada etiqueta por medio de las funciones de lectura y escritura de ítems en donde estas permiten realizar cambios a determinadas etiquetas o ítems que se dieron de alta en el cliente OPC de la envoltura.

A su vez el PLC puede transmitir valores hasta el lanzador de ordenes iMRP, pupes entre ellos existe una comunicación bidireccional por medio de la envoltura que a su vez contiene la conexión con el servidor OPC y el cliente OPC, el PLC como se encuentra en funcionamiento y conectado a un Servidor OPC y este a su vez a dos clientes OPC (uno en

un paquete computacional independiente y otro en la envoltura en visual studio), todo cambio realizado en el PLC se verá reflejado en el servidor y a su vez en los clientes.

Para realizar la lectura por medio del cliente opc este es llamado por medio de clases, en donde se usan las propiedades de propietario, aquí es donde se muestra si está conectado correctamente al servidor de OPC y cuáles son las etiquetas o Tags con los cuales está funcionando, también muestra el valor que contiene esa etiqueta.

Esto se realizó por medio de rutinas, las cuales son llamadas desde la envoltura.

- Rutina Conectando:

```
Public Sub conectando()
    If OPCRockwell Is Nothing Then
        OPCRockwell = objpropietario
    End If
    If OPCRockwell.Conectado Then
        If OPCRockwell.Desconectar() Then
            Timer1.Enabled = False
        Else
            ListBoxMensajes.Items.Add(OPCRockwell.Detalle_Error)
        End If
        ListBoxMensajes.Items.Add(OPCRockwell.Mensaje)
    Else
        If OPCRockwell.Conectar() Then
        Else
            ListBoxMensajes.Items.Add(OPCRockwell.Detalle_Error)
        End If
        ListBoxMensajes.Items.Add(OPCRockwell.Mensaje)
    End If
    If OPCRockwell.Conectado Then
        leeritems()
```

```
End If
 End Sub
```

- Rutina Leer Items

```
Public Sub leeritems()
    OPCRockwell.añadiritem(i, txtOFF.Text)
    txt2Off.Text = OPCRockwell.LeerItemInt(i)
    i = i + 1
    OPCRockwell.añadiritem(i, txtON.Text)
    txt2on.Text = OPCRockwell.LeerItemInt(i)
    i = 0
    Timer1.Enabled = True
      End Sub
```

Estas rutinas son pertenecientes a una clase, en donde en ella se encuentran las funciones principales por ejemplo, escribir ítems, conectar, desconectar, leer String, entre otros.

- Leer Ítems Enteros

```
'Función para leer un ítem que representa una variable entera
  'Se le pasa el índice del ítem que vamos a leer
  'Si todo va bien devuelve el valor de la variable
  Public Function LeerItemInt(ByVal Indice) As Integer
    Dim Valor As Object = Nothing
    Dim Calidad As Object = Nothing
    Dim TimeStamp As Object = Nothing
    If Not Conectado Then
      Mensaje = "Error conexión OPC."
      Detalle_Error = "No hay establecida una conexión OPC."
      LeerItemInt = 0
      Exit Function
```

```
        End If
        Try
            ItemOPC(Indice).Read(OPCDataSource.OPCDevice, Valor, Calidad,
TimeStamp)
            LeerItemInt = CInt(Valor.ToString)
        Catch ex As Exception
            Detalle_Error = ex.ToString
            Mensaje = "¡Error al leer Item! [Int, Índice " & Indice & "]"
            LeerItemInt = 0
            Exit Function
        End Try
        Mensaje = ""
        Detalle_Error = ""
         End Function
```

- Desconectar:

```
Public Function Desconectar() As Boolean
        Try
            Mensaje = "Desconectando..."
            ItemOPC = Nothing
            If Not IsNothing(ServidorOPC) Then
                ServidorOPC.OPCGroups.RemoveAll()
                ServidorOPC.Disconnect()
                ServidorOPC = Nothing
            End If
            GrupoOPC = Nothing
            GruposOPC = Nothing
        Catch ex As Exception
            Detalle_Error = "Error: " & ex.ToString
            Desconectar = False
```

```
            Exit Function
        End Try
        Mensaje = "Desconexión realizada correctamente."
        Detalle_Error = ""
        Conectado = False
        Desconectar = True
    End Function
```

Dando como resultado la ventana que se muestra en la figura 4.49.

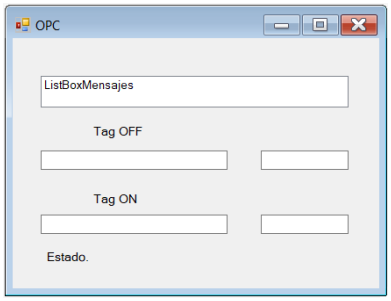


Figura 4.49 OPC para envoltura

4.7.1 Adaptación de Inserción de Código

Para la inserción de código desde un archivo y no propiamente crearlo en la aplicación fue necesario anexarle al código original un espacio donde se mostrara lo que contiene el archivo, también fue necesario utilizar la propiedad de propietario para la interacción con el lanzador y poder recibir y enviar información entre ellas.

- **Lista de Documentos añadidos:**

En esta sección se reciben desde el lanzador de órdenes las direcciones de los documentos que contienen el código en formato texto, estos son mostrados en la venta para una mejor visualización de ellos.

```
Public Sub enviarcodigo(ByRef eq1 As String, ByRef eq2 As String, ByRef eq3 As String,
ByRef eq4 As String, ByRef eq5 As String, ByRef eq6 As String, ByRef eq7 As String,
ByRef eq8 As String, ByRef eq9 As String)
```

```
    If eq1 IsNot Nothing Then
        ListBox1.Items.Add(eq1)
    End If
    If eq2 IsNot Nothing Then
        ListBox1.Items.Add(eq2)
    End If
    If eq3 IsNot Nothing Then
        ListBox1.Items.Add(eq3)
    End If
    If eq4 IsNot Nothing Then
        ListBox1.Items.Add(eq4)
    End If
    If eq5 IsNot Nothing Then
        ListBox1.Items.Add(eq5)
    End If
    If eq6 IsNot Nothing Then
        ListBox1.Items.Add(eq6)
    End If
    If eq7 IsNot Nothing Then
        ListBox1.Items.Add(eq7)
    End If
    If eq8 IsNot Nothing Then
        ListBox1.Items.Add(eq8)
    End If
    If eq9 IsNot Nothing Then
        ListBox1.Items.Add(eq9)
    End If
    cargar_origen_final()
End Sub
```

- **Cargar Documento:**

Para anexar al documento original el cual ya contiene todas las propiedades del PLC, es necesario ingresar el número correspondiente a él empezando por el cero “0”

```
    Dim i As String = TextBox10.Text
cargar_documento_añadido(ListBox1.Items(i))
```

- **Cargar Código:**

Para cargar cada documento correctamente es necesario seguir una serie de pasos, este botón inicializa las variables correspondientes cada vez que se carga un documento.

```
    Dim aRutina As String
'paso 3 inicializacion de variables
aRutina = "PROGRAM MainProgram"
TextBox4.Text = aRutina
txtBuscar.Text = aRutina
'paso 4
buscar2()
'paso 5
agregar_Tags()
'paso 6
aRutina = "ROUTINE MainRoutine"
TextBox4.Text = aRutina
txtBuscar.Text = aRutina
'paso 7
buscar2()
'paso 8
agregar_codigo()
```

Dado los cambios anteriores, el resultado final de la aplicación de inserción de código es lo mostrado en la figura 4.50.

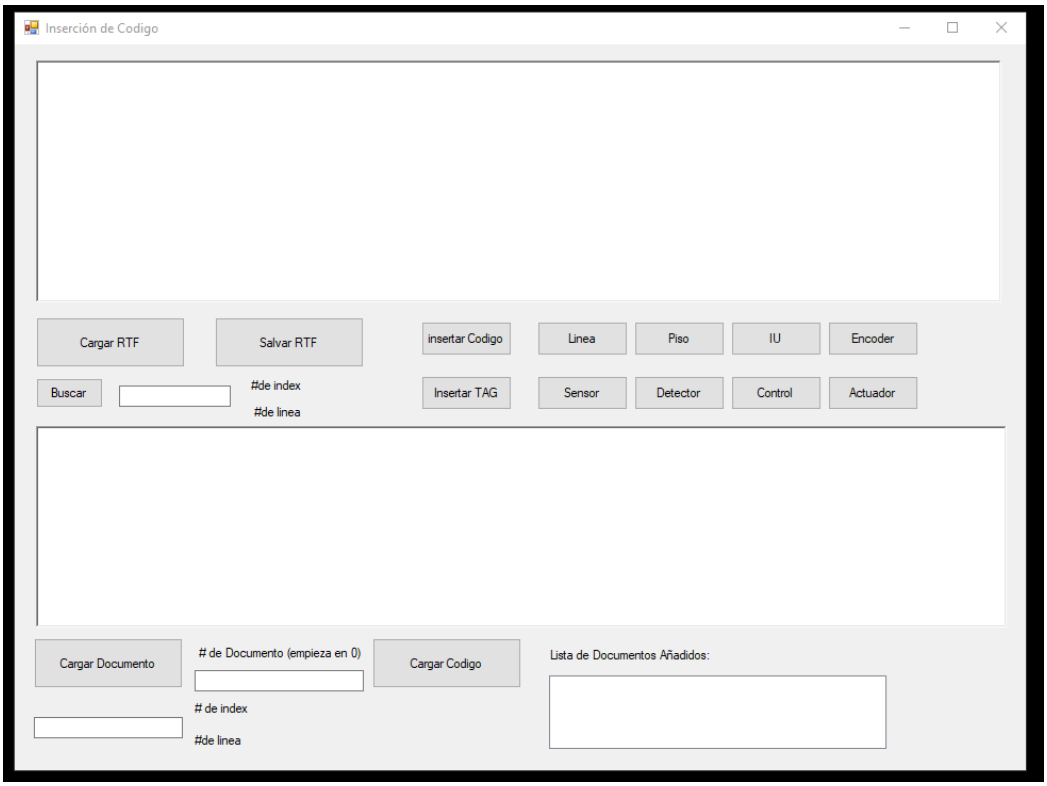


Figura 4.50 Inserción de código con modificaciones

En la ventana de Línea también se realizaron modificaciones, en donde ahora es posible anexar el nombre de los Tags así como la dirección completa del Tag, sin ninguna restricción de tipo ni de abreviaturas, quedando la ventana como se muestra en la figura 4.51.

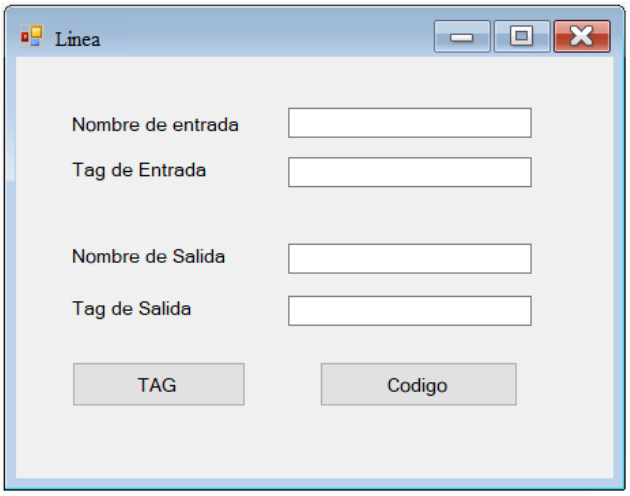


Figura 4.51 Inserción de línea con modificaciones en condigo

4.8 OPERACIÓN DEL SISTEMA PARTICULAR BASADO EN MODELO.

En el lanzador de ordenes iMRP se interpreta el modelo iMRP, el cual contiene todas las propiedades especificas del modelo como son los Tags en los arcos, las clases de los nodos, las direcciones del documento que contiene el código, al interpretarlo este realiza la instanciación de todas las clases contenidas en el con los parámetros correspondientes a cada clase particular, posteriormente se obtiene cada envoltura correspondiente a cada equipo del modelo iMRP, con ello se realiza la inserción de código y posteriormente la conexión con el PLC, dado esto se realiza la creación del servidor OPC y del cliente OPC, posteriormente la envoltura se encarga de realizar la conexión con el servidor y la creación del cliente OPC.

Al tener todos los parámetros realizados el lanzador de ordenes iMRP determina el orden en el cual será llevado a cabo la operación del sistema por medio de la interpretación del iMRP que se realizó previamente al modelo iMRP y con ello se operan los sensores y actuadores en el orden correspondiente.

CAPITULO 5
VALIDACIÓN Y RESULTADOS DE LA SOLUCIÓN PROPUESTA

5.1 EJERCICIOS DE VALIDACIÓN

La simulación consistirá en implementar un sistema de control basado en PLC a partir de un sistema genérico y el modelo de particularidades. A partir del modelo de particularidades del sistema particular abstraído en un Grafo iMRP, el sistema genérico identificado como lanzador de órdenes instancia el código a cargar en el PLC. Una vez cargado en el PLC el programa así generado, es ejecutado y comunicado con el lanzador de órdenes (en ejecución en la PC). Un servidor OPC permite la interacción entre el Lanzador de órdenes y el PLC, esto en la plataforma de RSLOGIX 5000, RsLinx y Visual Studio.

Como ejercicio de validación se plantea un sistema a controlar por PLC, el sistema consiste en una banda transportadora, que conduce envases a ser llenados debajo de una tolva (Figura 5.1). Como actuadores se encuentran el motor de la banda y el cilindro neumático para abrir y cerrar la compuerta de vaciado de la tolva; así mismo se encuentran los sensores de caja llena y presencia de caja.

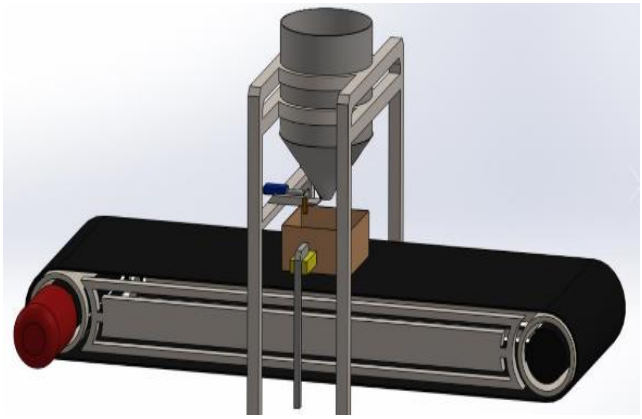


Figura 5.1 Ilustración del sistema de validación

El grafo de iMRP figura 5.2 cuenta con dos nodos de grafo producto el P1 representa la caja está vacía y el P2 cuando esta se encuentra llena, así como con tres tipos de recursos tiempo a ejecutar el RT1 el cual está ligado a una operación de transporte para la banda, posteriormente pasa al producto P2 en donde se realiza el RT2 el cual depende de una operación de procesamiento en donde se indica que la caja ha llegado al lugar adecuado y el

actuador está en funcionamiento –llenando-, posteriormente el RT3 procede con la operación de transporte de la caja que se encuentra llena.

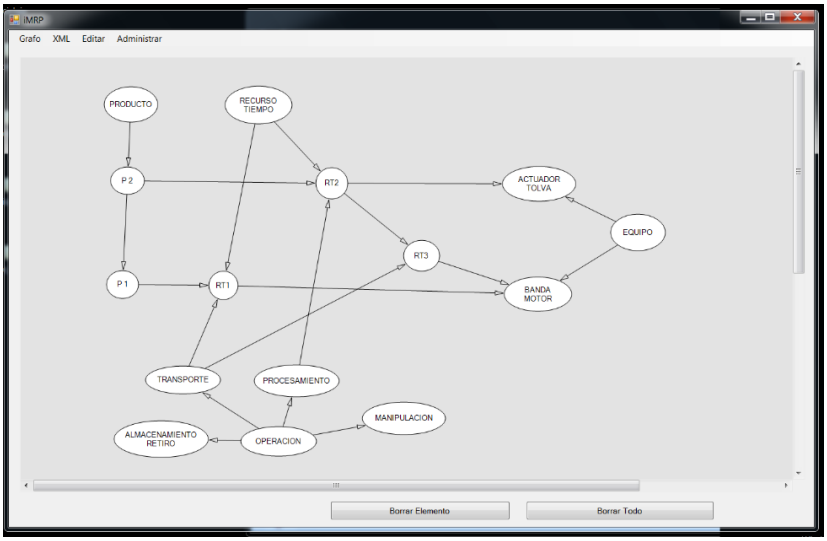


Figura 5.2 Grafo del Modelo iMRP del ejercicio de validación.

Cada nodo de Recurso tiempo está ligado a un nodo actuador y a una operación en donde en el arco RT a Actuador se proporciona el Tag para activar la rutina del actuador. En el arco RT a Operación se proporciona el Tag en donde se indica el fin de la operación, Figura 5.3

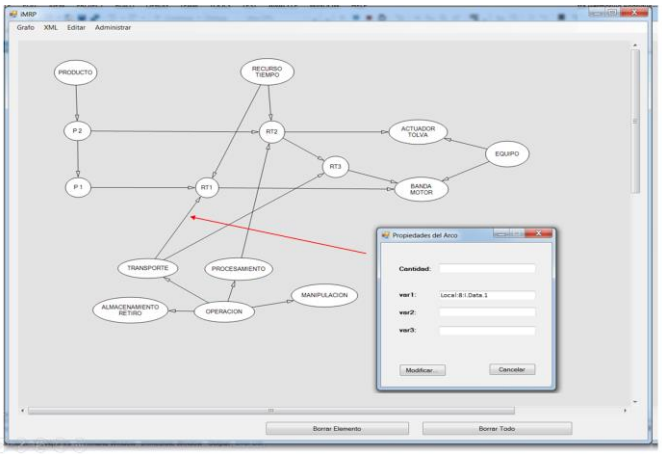


Figura 5.3 Propiedades del arco indicacion de fin de operación

Los nodos cuentan con propiedades figura 5.4 en la cual en el nodo tipo equipo, en la propiedad tipo se introduce la dirección de la envoltura correspondiente a ese equipo, y en la propiedad Var1, se introduce la dirección del archivo correspondiente a ese equipo**.**

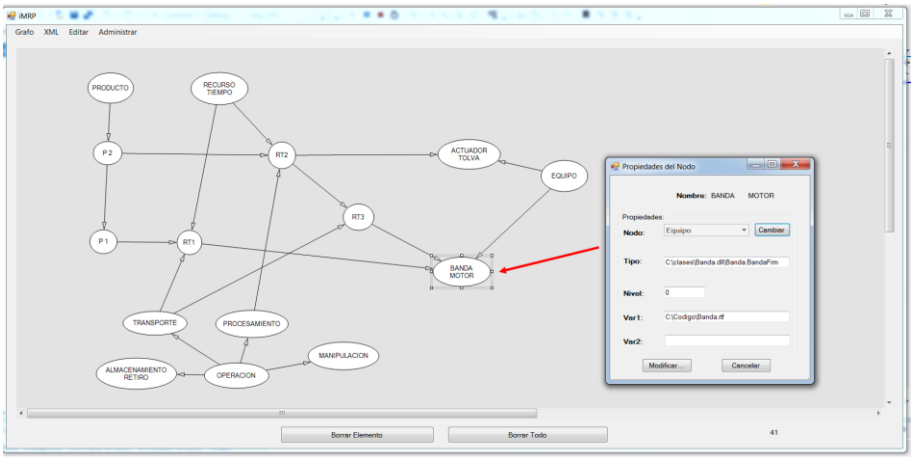


Figura 5.4 Propiedades de los nodos

En el modelo iMRP que se muestra en la figura 5.5 se ilustra el contenido de cada atributo de arco correspondiente a cada RT para el ejercicio de validación

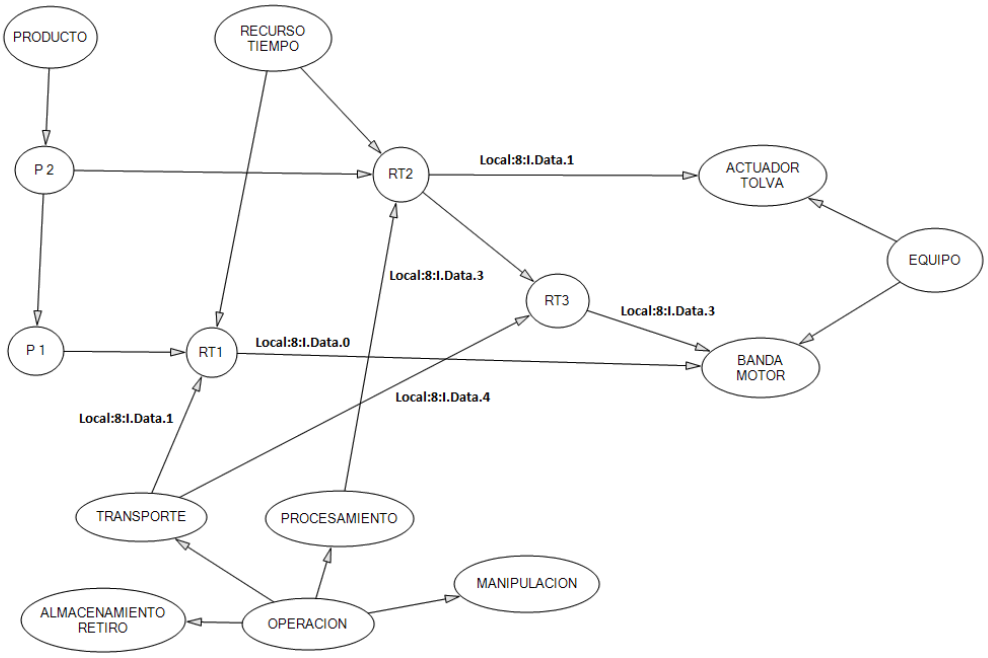


Figura 5.5 Modelo iMRP con propiedades del ejerció de validación.

Una vez realizado lo anterior se prosigue a la utilización del lanzador de ordenes iMRP en donde por medio de la base de datos MySQL se accede al grafo iMRP creado en el Editor como se muestra en la figura 5.6

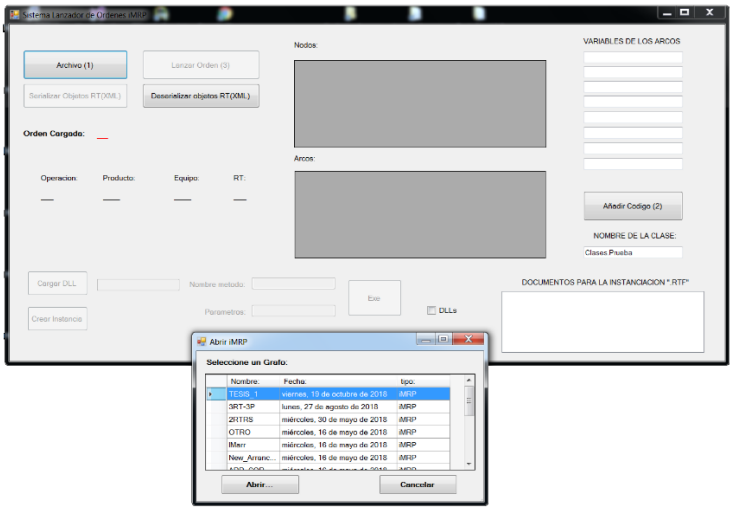


Figura 5.6 Acceder al Grafo del Modelo iMRP en el lanzador de ordenes iMRP.

Así es como se muestra el lanzador de ordenes iMRP una vez que el modelo iMRP esta seleccionado y cargado en el (figura 5.7).

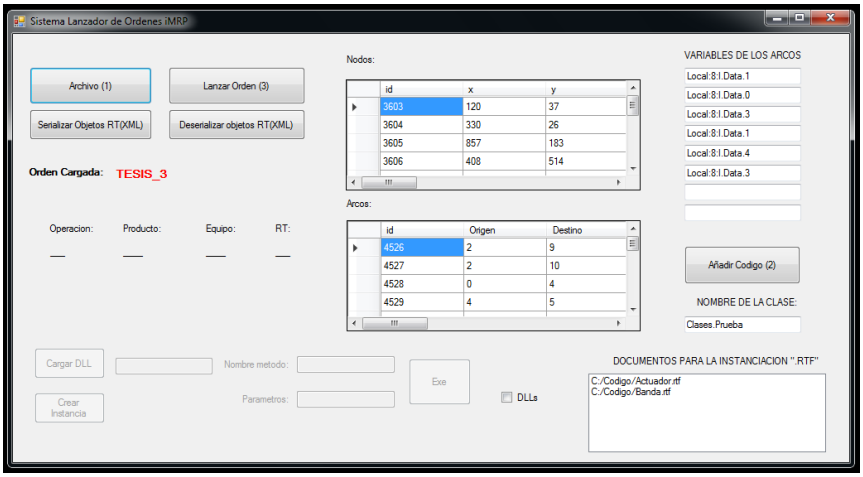


Figura 5.7 Lanzador de ordenes iMRP con el modelo correspondiente.

El lanzador instancia cada una de las envolturas así como también muestra las direcciones de los archivos de estos y muestra los valores en los arcos pertenecientes a todos los recursos tiempo.

A su vez el lanzador despliega las ventanas de los objetos envolturas correspondientes para el modelo iMRP seleccionado (Figura 5.8)

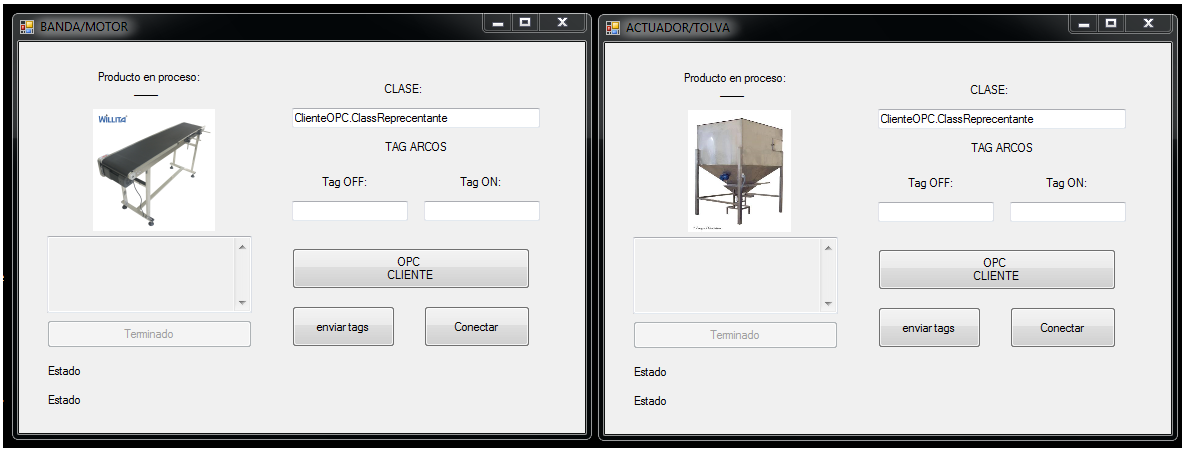


Figura 5.8 Envolturas correspondientes para el ejercicio de validación.

Cada envoltura abre su propio cliente OPC correspondiente, por medio de un archivo tipo ".DLL" (Figura 5.9).

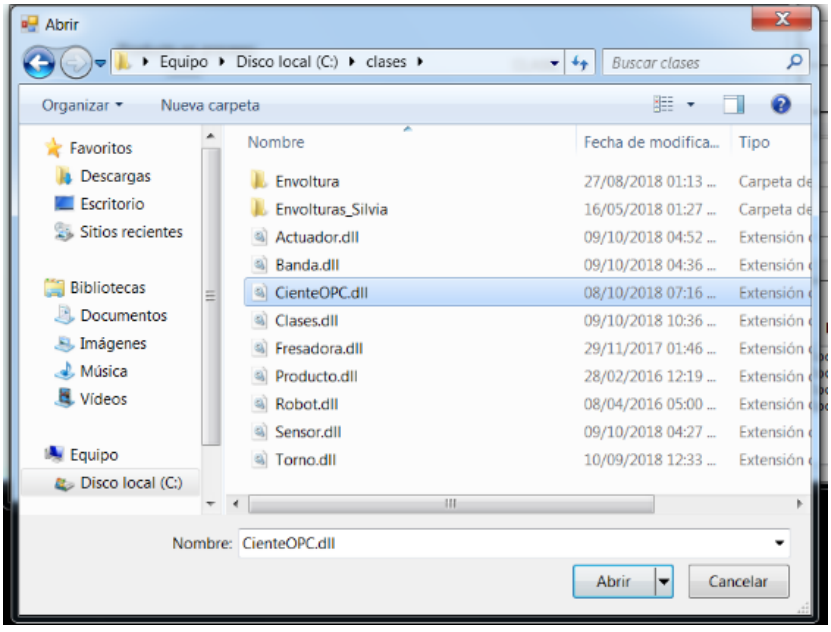


Figura 5.9 Selección del archivo correspondiente de tipo ".dll"

A su vez se realiza la inserción de código por medio de lanzador de ordenes iMRP en donde al ingresar a la sección de Añadir código como se muestra en la figura 5.7 es necesario seleccionar el archivo “Clases.dll” el cual es correspondiente para realizar la inserción de código como se muestra en la figura 5.10

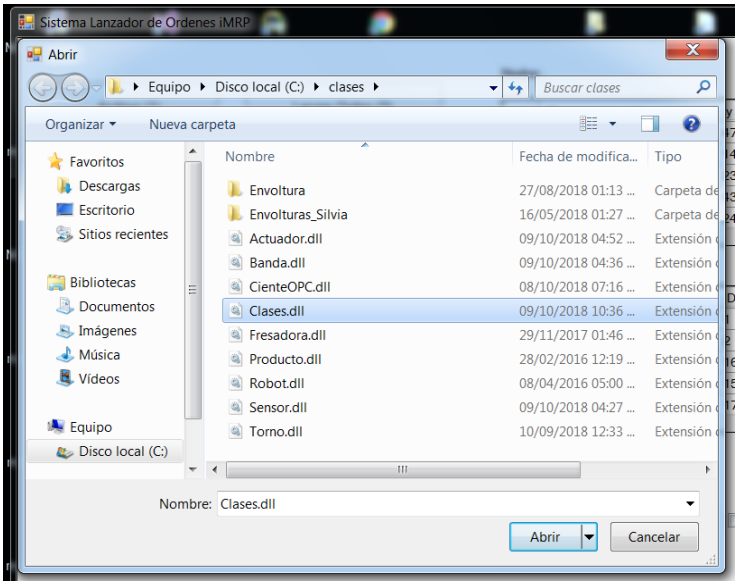


Figura 5.10 Selección del archivo con extensión “.dll” para la inserción de código

La aplicación de inserción de código pide al usuario ingresar manualmente el archivo de formato “.RTF” en el cual se l se escribirá el programa del PLC (figura 5.11) –previamente preparado, (Rodríguez, D., 2017)-

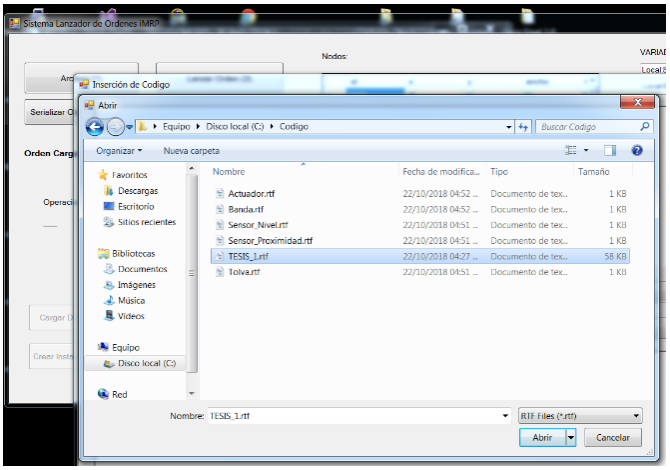


Figura 5.11 Selección del archivo con requerimientos del PLC

Una vez cargado el archivo, se despliega la aplicación para la inserción de código en donde ya se encuentra ingresado el archivo que contiene los requerimientos del PLC y las direcciones de los archivos que contienen el código para la inserción como se muestra en la figura 5.12.

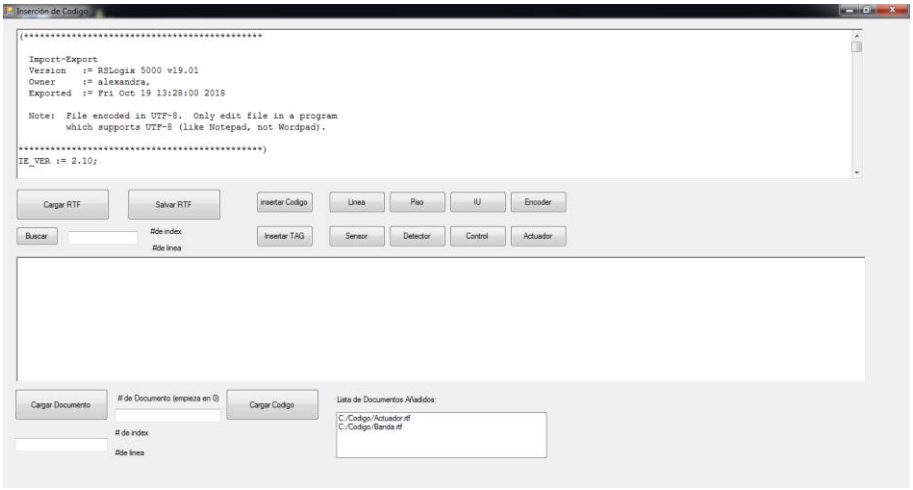


Figura 5.12 Caratula principal de la aplicación de inserción de código

Se prosigue a cargar en el PLC manualmente cada uno de los documentos proporcionados por el lanzador, los cuales pertenecen a cada envoltura, esto se realiza al ingresar el número del documento a cargar empezando por el cero (“0”) como se muestra en la figura 5.13.

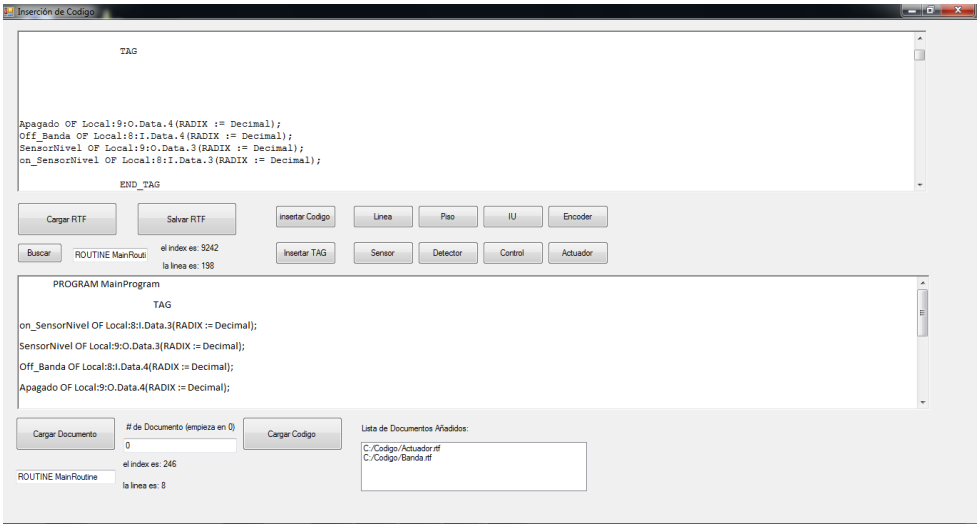


Figura 5.13 Realización de inserción de código.

Posteriormente se guarda el archivo en formato “.L5K” como se muestra en la figura 5.14.

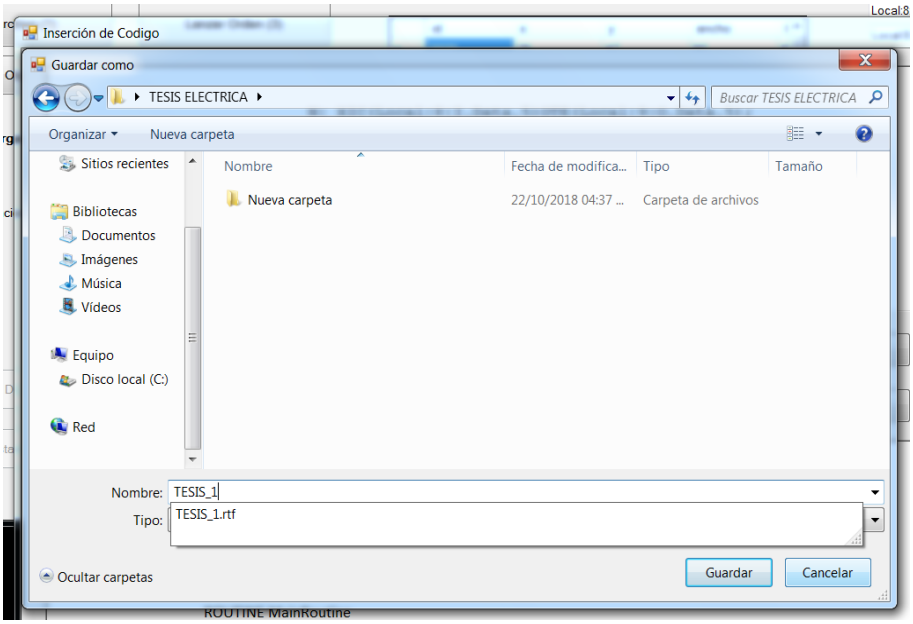


Figura 5.14 Guardar archivo con las inserciones de código.

Al ingresar al programa RsLogix5000 y seleccionar abrir un archivo existente, permite abrir archivos con formato “.ACD”, “.L5K”, “.L5X” y “XML”, por lo tanto se seleccionó el archivo creado en la aplicación de inserción de código (figura 5.15).

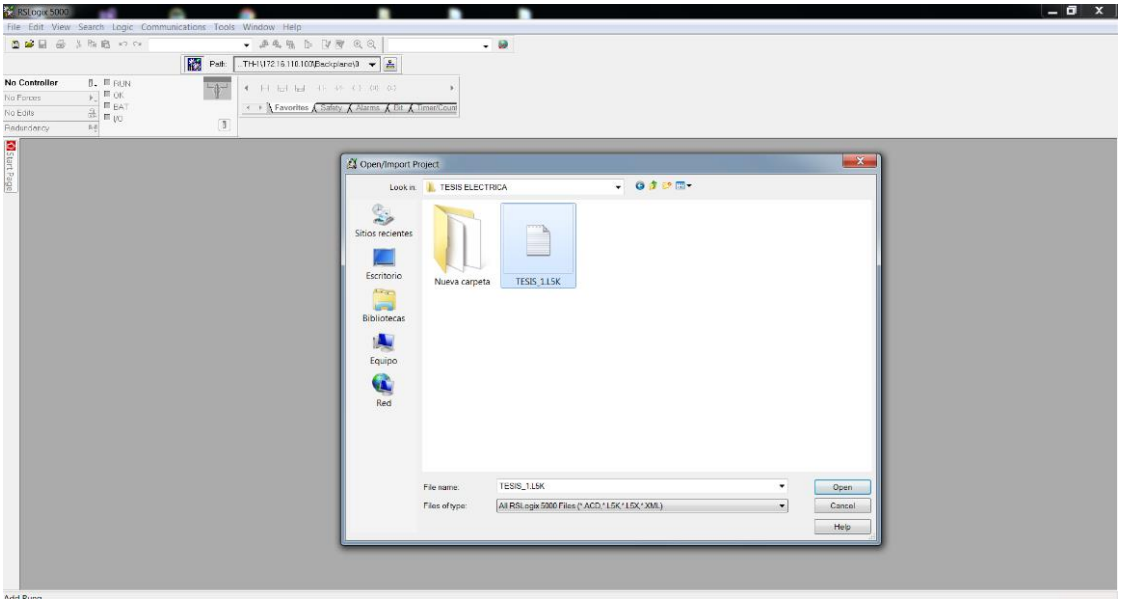


Figura 5.15 Selección del archivo con extensión “.L5K”

Se muestra lo creado por la inserción de código en código tipo escalera (figura 5.16).

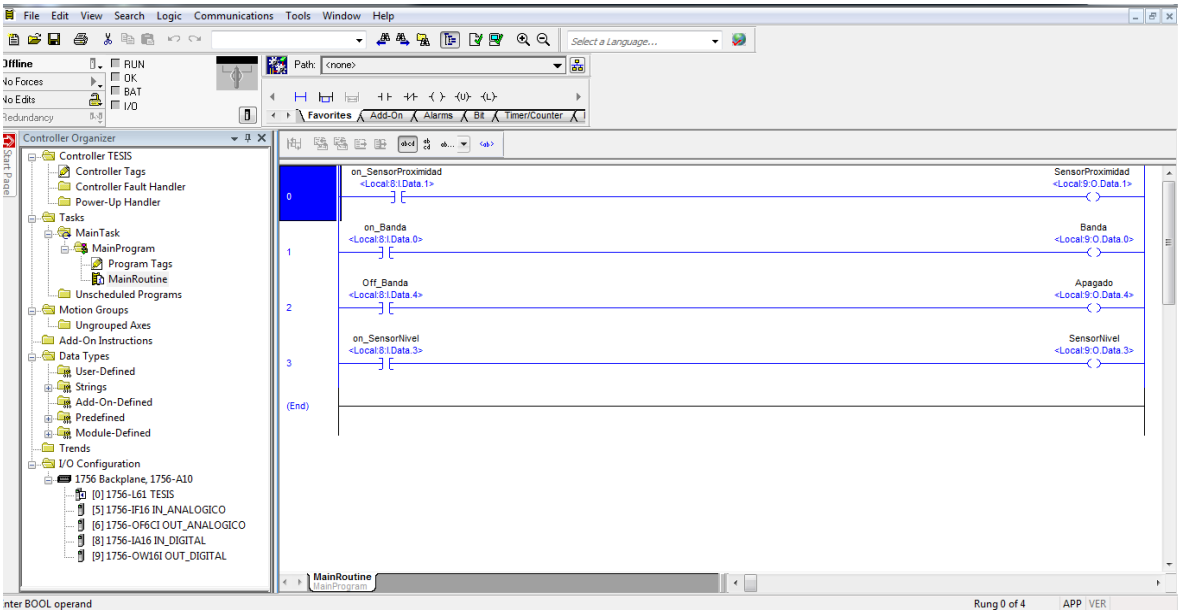


Figura 5.16 Código tipo escalera.

Se prosiguió con la realización de un servidor OPC, por medio del RsLinx, para esto el programa debe de estar previamente cargado y en modo run en el PLC deseado como se muestra en la figura 5.17.

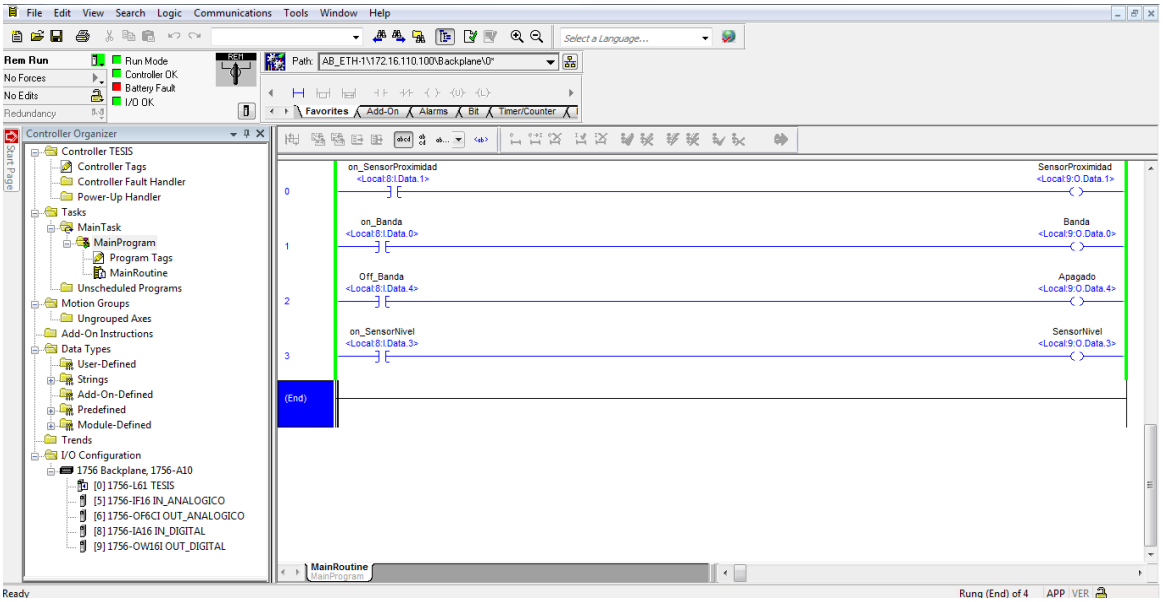


Figura 5.17 Código tipo escalera cargado y en modo run.

Posteriormente se realiza la creación del servidor OPC por medio del RsLinx como se ilustra en las siguientes figuras 5.18, 5.19, 5.20 en donde se selecciona el PLC en el cual se realizara el OPC y se da aplicar y done, al realzar esto se crea el servidor OPC.

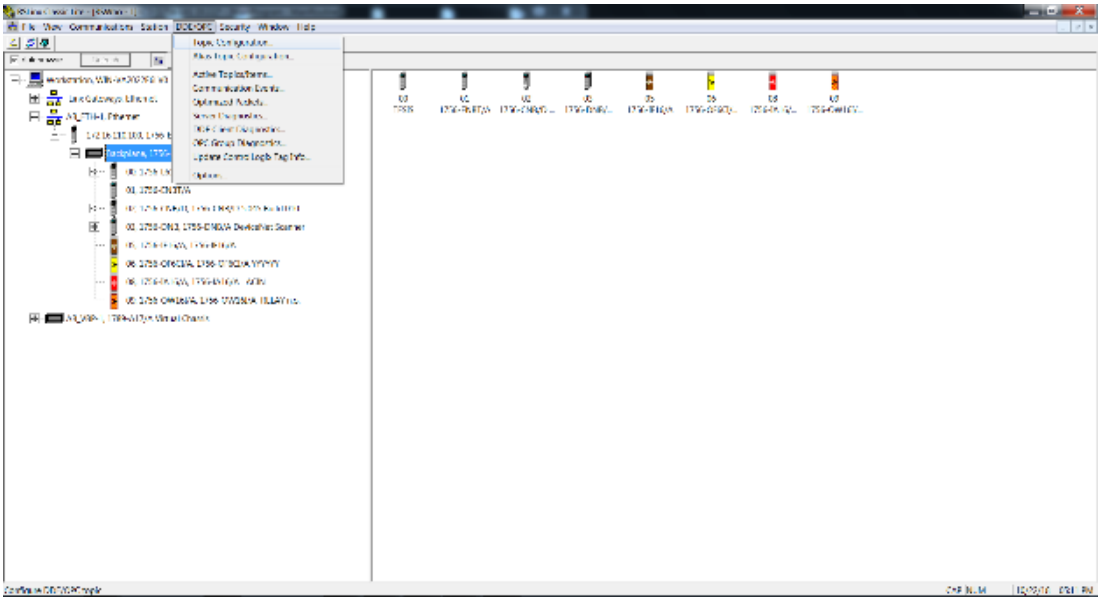


Figura 5.18 Primer paso para crear un OPC en RsLinx

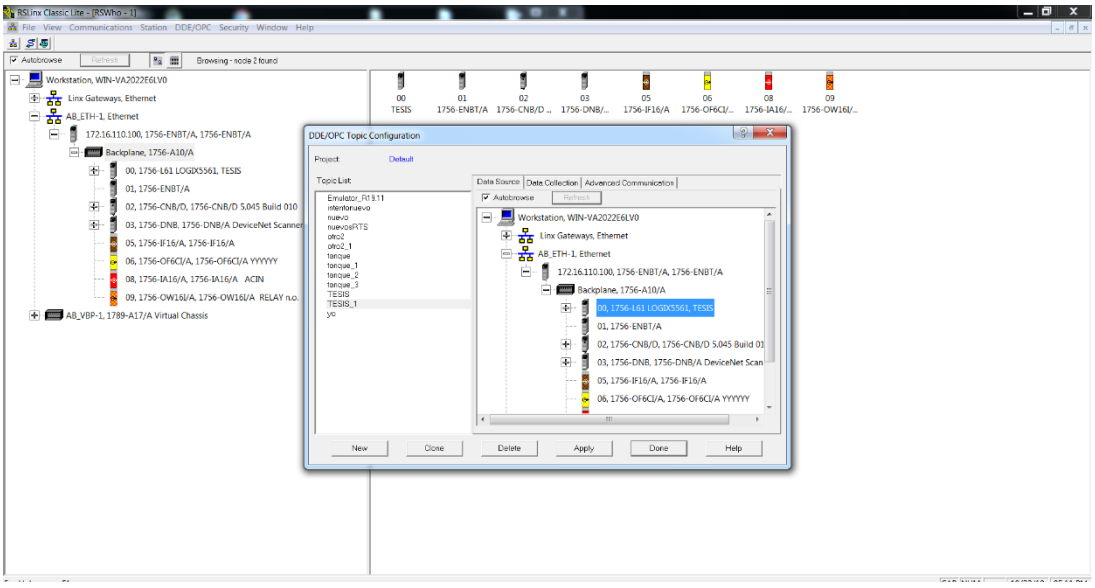


Figura 5.19 Selección del PLC

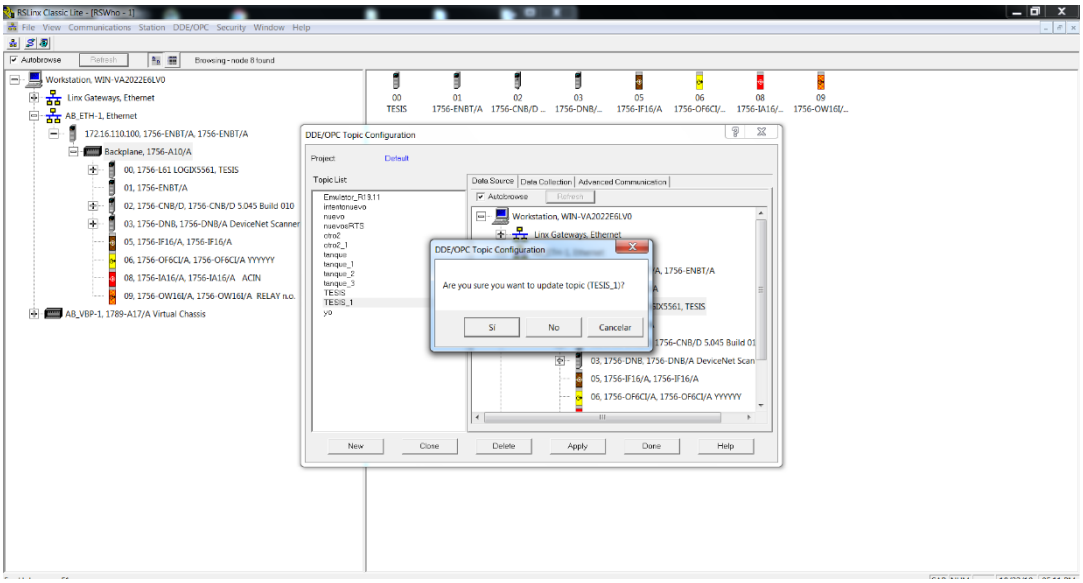


Figura 5.20 Aplicación del OPC en el PLC seleccionado

Posteriormente se abre el paquete computacional OPC Test Client el cual es perteneciente al RsLinx (figura 5.21), se selecciona el servidor deseado en este caso es el RsLinx OPC Server y con él se realiza el Cliente para la visualización de las variables

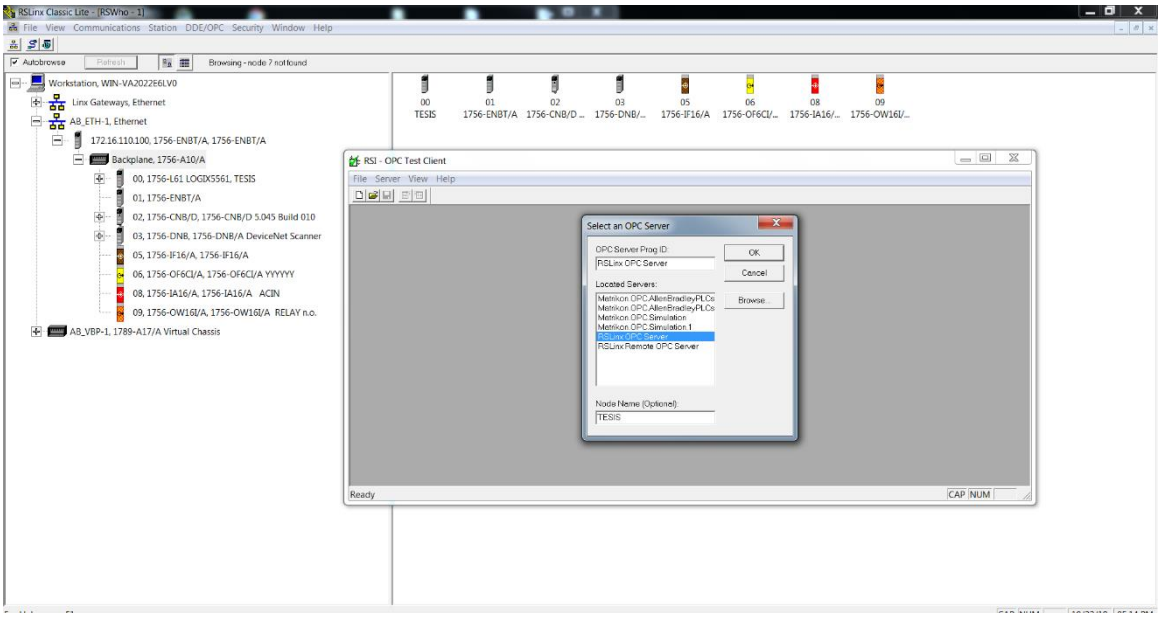


Figura 5.21 Selección del servidor OPC.

Se prosigue a realizar el grupo en el cual contendrá las variables o ítems y posteriormente se añaden cada uno de estos ítems, se validan y posteriormente de despliegan en el OPC Test Client como se muestra en la figura 5.22

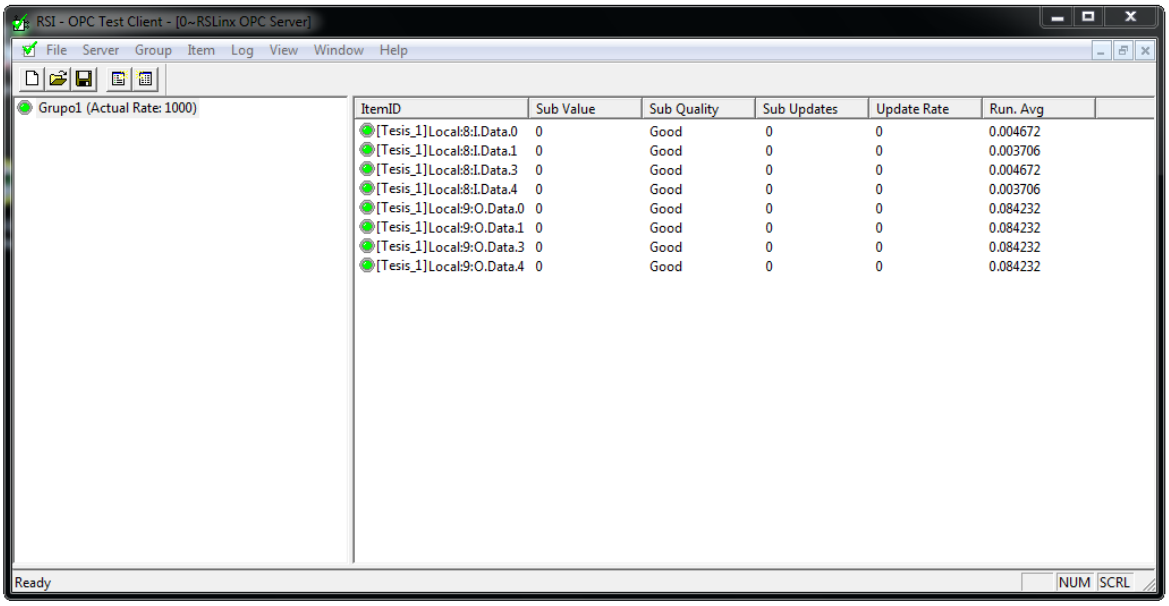


Figura 5.22 monitoreo de variables del OPC en Cliente OPC

Después de la realización del OPC se prosigue a lanzar la orden en el Sistema lanzador de ordenes iMRP, en el cual el lanzador de ordenes iMRP identifica cual es el primer producto a realizar el cual es P1 el cual pertenece al Recurso Tiempo RT1 por medio de una operación de Transporte realiza la operación en la banda/Motor, estos tienen como Tag de inicio Local:I:8.Data.0 y Tag de fin de operación Local:I:8.Data.1 como se muestra en la figura 5.23, al identificarlos prosigue la envoltura correspondiente a realizar la conexión con el servidor OPC y a crear el cliente OPC con los Tags identificados para esa acción, como se muestra en la figura 5.24 una vez realizada la conexión el lanzador de ordenes iMRP ejecuta la rutina correspondiente para manda a operar el Tag de inicio en el diagrama escalera en el PLC como se muestra en la figura 5.25.

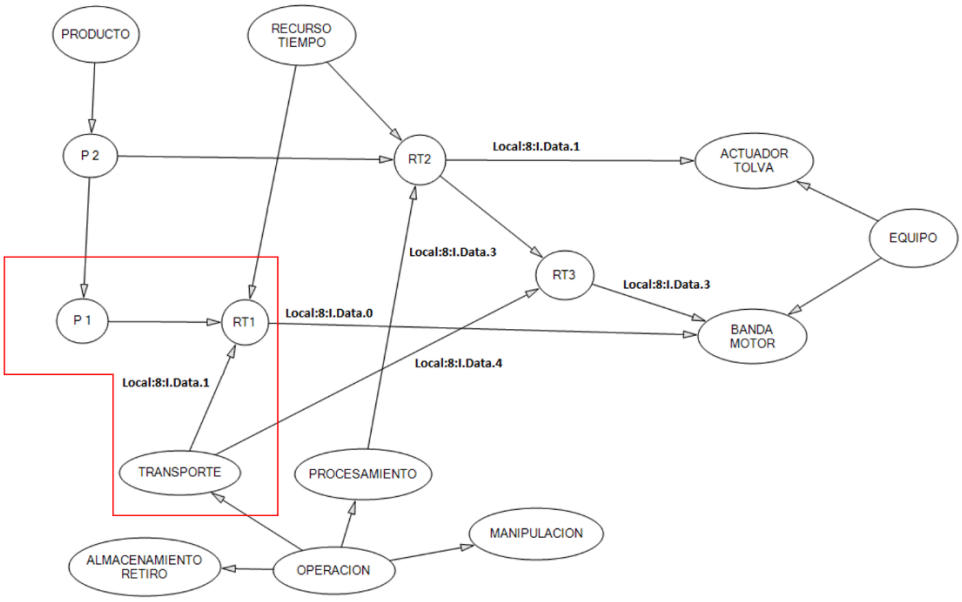


Figura 5.23 Identificación de la primera operación

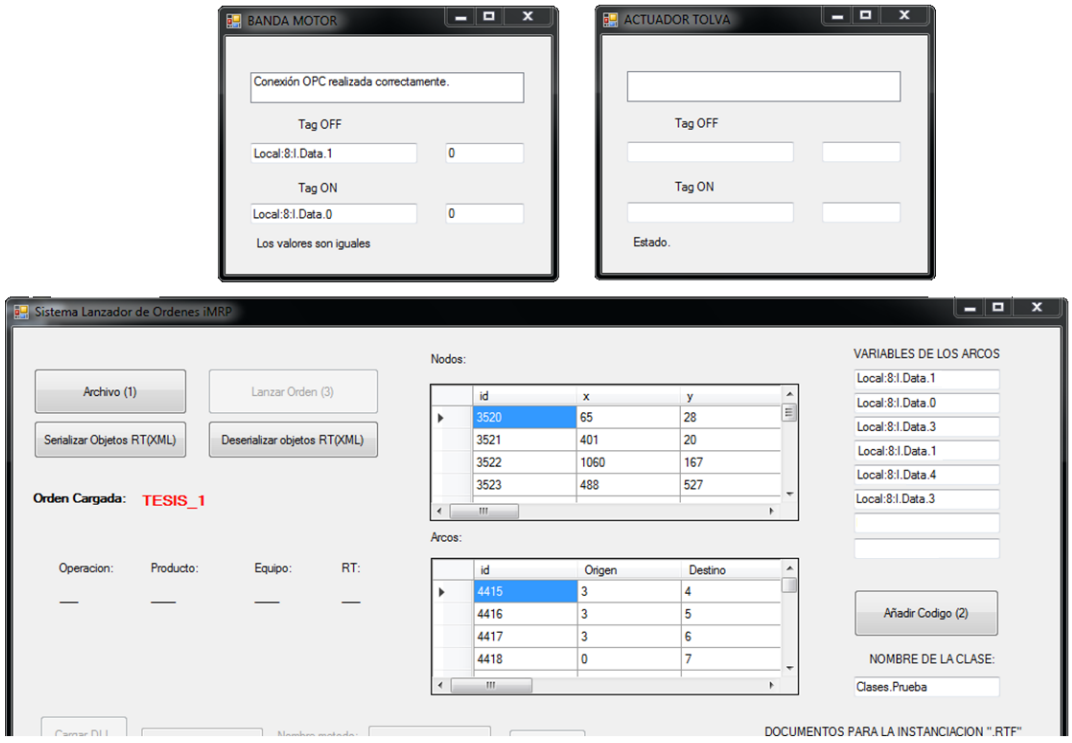


Figura 5.24 Conexión con el servidor OPC por parte de la envoltura.

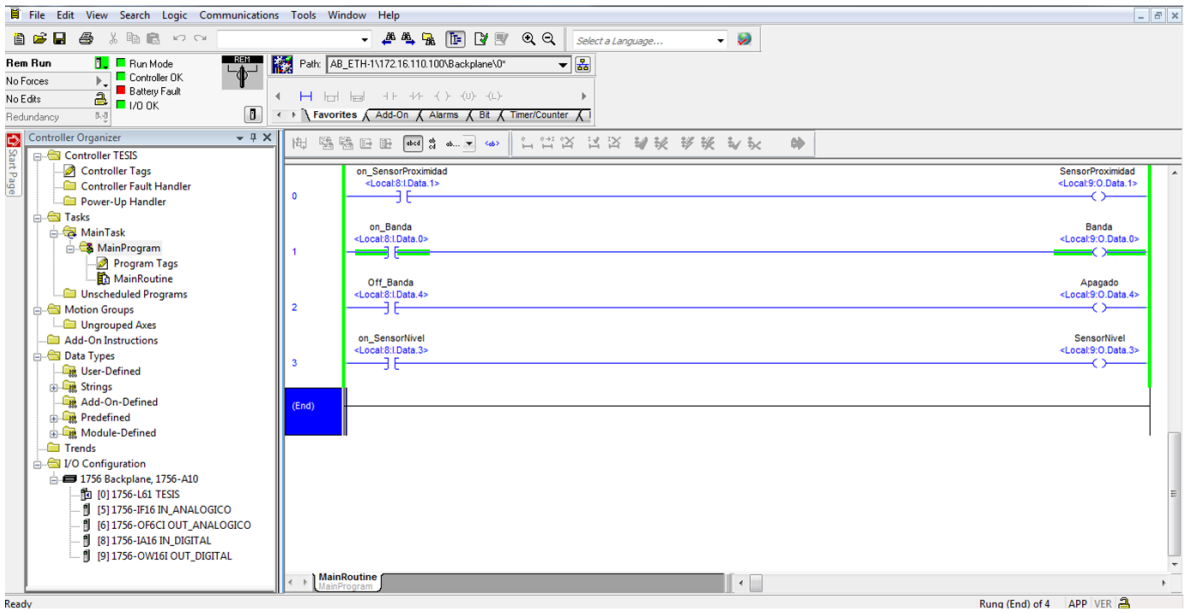


Figura 5.25 Diagrama escalera en PLC en funcionamiento.

Una vez que el Tag de fin de operación este accionado en el PLC como se muestra en la figura 5.26. En la envoltura se muestra el estatus actual de esta así como en qué tipo de operación esta y si esta ya fue concluida, como se muestra en la figura 5.27

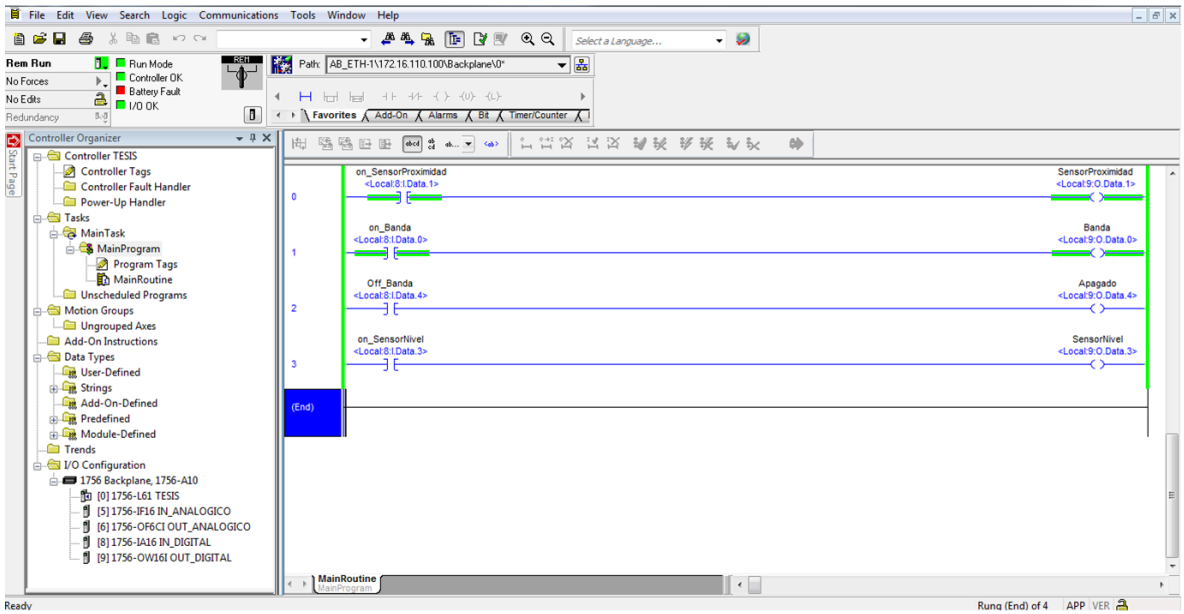


Figura 5.26 Accionamiento de Tags en código tipo escalera.

Figura 5.27 Estatus de la envoltura.

Posteriormente la envoltura realiza una desconexión del servidor OPC y el lanzador de ordenes iMRP prodigue con el nodo hijo del RT actual en caso que este cuente con alguno si no es el caso prosigue con el siguiente producto, como lo es en este caso prosigue con el producto P2 el cual pertenece al RT2 y realiza una operación de procesamiento en el actuador/tolva, como se muestra en la figura 5.28

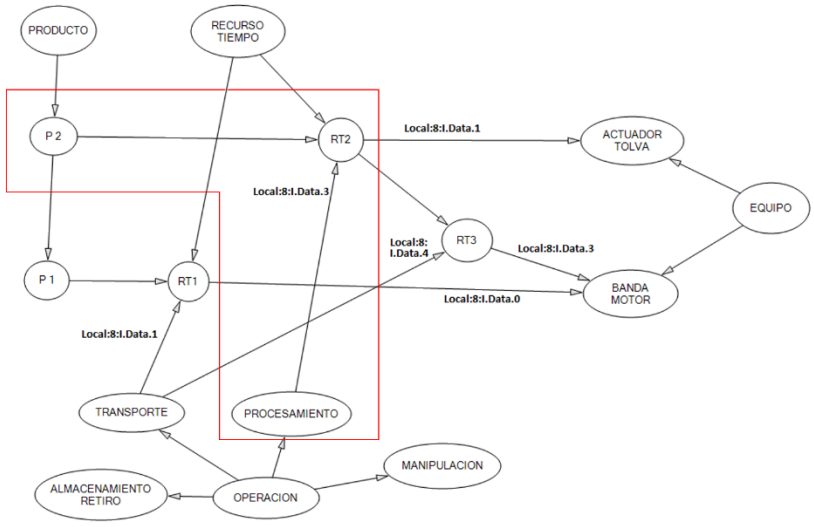


Figura 5.28 Realización del siguiente RT por parte del lanzador de ordenes iMRP

Esto se realiza consecutivamente para cada RT que se encuentre en el Modelo iMRP.

En la envoltura se realiza una conexión al servidor OPC y un cliente OPC cada vez que se realiza una operación en la siguiente secuencia de figuras 5.29 se puede observar como es la

realización de estas conexiones al ir avanzando en la operación por parte del lanzador de ordenes iMRP

Figura 5.29 secuencia de la interconexión servidor OPC – cliente OPC

Una vez realizada toda la operación del modelo iMRP por parte del lanzador de ordenes iMRP, las envolturas muestran que el proceso se encuentra concluido y se muestra un

mensaje donde indica que se concluyó satisfactoriamente la operación del modelo iMRP como se ilustra en la figura 5.30.

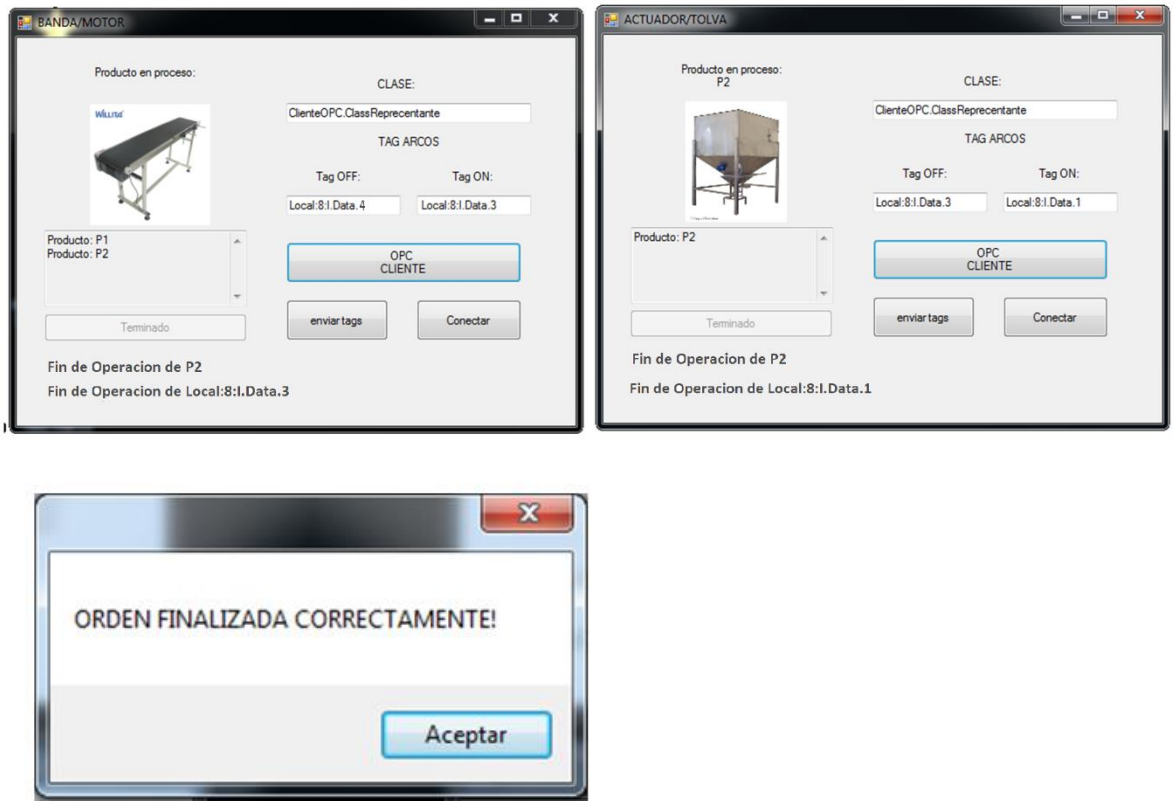


Figura 5.30 Estatus final de las envolturas

A la par del proceso en el lanzador de ordenes iMRP, el OPC está realizando sus cambios y monitoreo como lo muestran la figura 5.31, esto es por medio de la envoltura en donde esta manda a realizar el cambio en el OPC y con ello en el PLC se produce el cambio, dando así como resultado la interacción entre el lanzador de órdenes, la envoltura y el PLC.

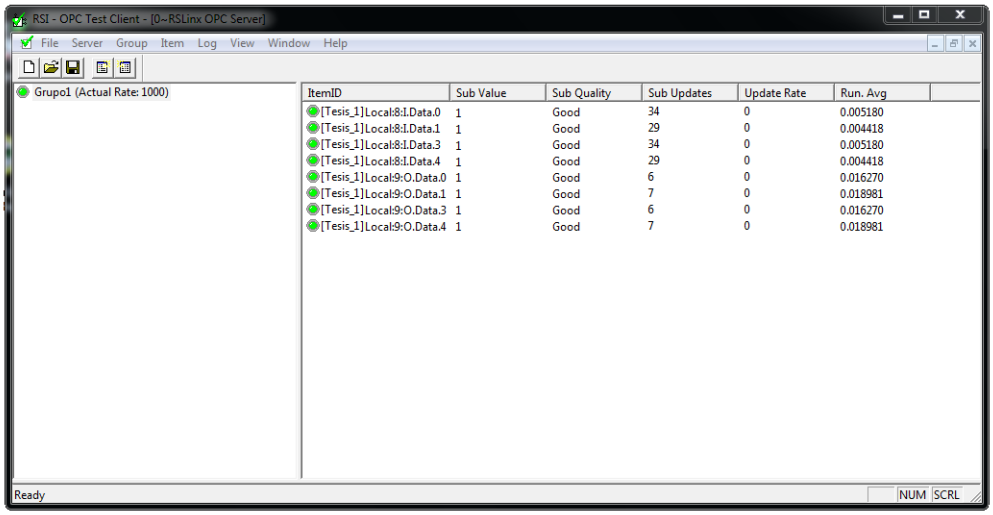


Figura 5.31 Cambios en el cliente OPC por medio del lanzador de ordenes iMRP.

5.2 ANÁLISIS DE RESULTADOS

Los resultado obtenidos son satisfactorios pues se realizó correctamente la interacción entra cada aplicación así como se obtuvo el funcionamiento esperado del sistema controlado por el PLC, implementado y guiado por modelo.

La interacción del Grafo iMRP con el MySQL, se realizó correctamente obteniendo con ello una base de datos sólida y confiable para posteriormente utilizarla en el Sistema Lanzador de Ordenes iMRP, en donde la interacción con el OPC por medio de la envoltura se realizó correctamente al seleccionar como servidor de OPC el RsLinx Server y posteriormente el RsLinx Cliente, en donde la creación de ítems y grupos fue relativamente fácil al tener la conexión ya establecida con el RSLogix5000, esto fue realizado por medio del Lanzador de Ordenes el cual satisfactoriamente abstrajo los documentos necesarios con las rutinas requeridas y posteriormente se cargaron exitosamente al PLC por medio del archivo con extensión ".L5K", al tener el PLC en modo operación, la realización de lanzar la orden y su secuencia que fue determinada por el Grafo iMRP, fue correctamente interpretada y con ello se realizó la Operación dirigida por modelo de una manera fácil y correcta.

CAPITULO 6
CONCLUSIONES

6.1 CONCLUSIONES

Utilizar el esquema de referencia ArquiTAM, permitió enfocarse en cada uno de los puntos independientemente del resto de los puntos en el desarrollo, teniendo un mayor aprovechamiento, al contar con una arquitectura de referencia sólida, esto permitió ver claramente los puntos necesarios a utilizar en el modelo de referencia y descartar los que no son propiamente necesarios y/o indispensables; con ello se realizó un modelo de referencia sólido y completo para posteriormente ser aplicado en la aplicación particular de manera satisfactoria.

Al utilizar el iMRP como técnica de modelado en lugar de alguna otra técnica como pudiera ser máquina de estados finitos, se realiza de una manera más practica el cambio en el modelado, además de representar en una sola grafica las particularidades de estructura y lógica de operación del sistema; si por alguna razón se quiere agregar o quitar algún elemento, con esta técnica es mucho más sencillo pues en la máquina de estados finitos cambia substancialmente el modelo y en particular su implementación en código; en iMRP puede simplemente llegar a cambiar una rama de él y no todo el modelo. El modelo del sistema mediante la técnica de modelado iMRP es relativamente simple de aplicar y de interpretar por computadora, en este tipo de modelo se representa tanto la estructura del sistema como la secuencia de operación del mismo. Cuenta con constructores nodo y arco, con significado dependiente del tipo de grafo los cuales son, Productos, Recurso Tiempo, Equipos y Operaciones que a su vez las Operaciones son clasificadas en cuatro tipos Transportar, Procesar, Manipular, Almacenar/Retirar. Mediante estos constructores originalmente dirigidos al modelado a nivel máquina, es posible modelar el sistema a nivel sensores y actuadores, elementos básicos en la implementación de un sistema controlado por PLC.

La aplicación utilizada para almacenar el modelo iMRP en la base de datos (MySQL) resulta sólida y confiable, y de fácil acceso para su utilización por el Sistema Lanzador de Ordenes , es necesario tener previamente instalado el paquete Visual Studio en donde este y todos las aplicaciones se desenvuelven así como también tener instalado todos los paquetes correspondientes de Rockwell, en donde la versión del RsLinx es muy importante esta no

debe ser una versión Lite, pues no cuenta con la opción de crear OPC correctamente, también es necesario considerar que dependiendo de la versión que se tenga instalada de Rockwell es necesario para la compilación del Visual Studio para evitar problemas de compatibilidad, dado todas estas observaciones, se puede concluir que se cumplió satisfactoriamente el objetivo general de esta tesis así como los objetivos específicos, en donde se aplicó el concepto de acoplamiento débil en la integración de PLC-Sensores/Actuadores, así como en la interconexión de PC-PLC y se desarrolló el sistema genérico para PLC el cual opera con base en la técnica de modelado de iMRP, con ello se obtiene una flexibilidad bastante considerable en la realización e integración del PC-PLC y una reducción de tiempo hombre en su implementación, además se logró la operación en una estación que contiene un PLC.

Del desarrollo del presente proyecto de tesis se puede concluir que es factible la aplicación de la técnica de implementación y operación dirigida por modelo en sistemas basados en PLC, sin embargo, el nivel de madurez de los lenguajes de programación para PLC, en específico la capacidad de manejo de programación orientada a objeto, hace necesario el apoyo de un sistema operando en PC para la implementación del sistema genérico que interprete el modelo de particularidades.

REFERENCIAS
BIBLIOGRAFÍA

Acosta, J. E., *Esquema de Referencia para acoplamiento débil entre Sistema informático y equipo de producción*, Director: Francisco Baguena, Tesis Doctoral Universidad Politécnica de Madrid, 2015.

Acosta C. J., Sastrón B. F., (2013*). Loose Coupling Based Reference Scheme for Shop Floor-Control-System /Production-Equipment Integration*. Journal of Applied Technology and Research (JCR), Vol. 11, N. 3. **ISSN:** 1665-6423.

Aguayo L. B., Desarrollo de un Sistema de Cero Configuración (uPNP) para Control Informático de Equipo en el Piso de Producción, Tesis de Maestría, Instituto Tecnológico de Chihuahua, Enero 2014

Alder, K., *Engineering the Revolution: Arms and Enlightment in France*, 1763-1815. (2nd Ed.) Chicago: University of Chicago Press 2010. ISBN: 978-0-226-01265-0

Andersson, N. Design principles for open and reusable shop-flor control software, *Computers in Industry*, 1997. 33 (2-3). 285-293. DOI: 10.1016/S0166-3615(97)00034-1

Carreón, E. V., *Plataforma Plug and Play para Células de Manufactura Basada en el Estándar ISO 9506.* Tesis Maestría. Facultad de Ingeniería Universidad Autónoma de Chihuahua. Junio 2015.

Controlador Lógico Programable, Wikipedia.org, 2018, https://es.wikipedia.org/wiki/Controlador_l%C3%B3gico_programable.

Dai, W., Vyatkin, V. Redesing Distributed PLC Control Systems Using IEC 61499 Functions Blocks. *Automation Science and engineering* IEEE Transactions on. 2012, 9(2), 390-401

Desrochers A. *Applicattion of Petri Nets in manufacturing systems* IEEE Press 1995.

ElMaraghy, H. A. Flexible and reconfigurable manufacturing systems paradigms, *International Journal of Flexible Manufacturing Systems* 2006. 17(4), 261-276. DOI: 10.1007/s10696-006-90028-7

Ferrolho, A., Crisóstomo, M. Intelligent Control and Integration Software for Flexible Manufacturing Cells, *Industrial Informatics* IEE Transactions on. 2007 3(1), 3-11.

Ford H. *My life and work*, Garden city, N.Y.: Doubleday, page & co. 1992 (1922 1ra Ed.) [eBook-Project Gutenberg, 2005]. WEB :< http://public-library.uk/pdfs/6/860.pdf>

Ford, H. *Mass Production*. (Enciclopedia Británica, Ed.) (13th Ed).1926

Fujimoto, T. The Evolutions of Production Systems: Exploring the Sources of Toyota's Competitiveness. *Annals of Business Administrative Science*. 2012. 11, 25-44

IEC. International Standard IEC61499, *Function Blocks*, Part 1 – Part 4. International Electo-technical Commission (IEC). 2005

Kolluru R., Smith S., Meredith, P., Loganantharaj, R., Chambers, T., Seetharaman, G., D'Souza, T. A. Frameworki for the Development of Agile Manufacturing Enterprises. In *Proceedings of the 2000 IEEE International conference on Robotics & Automation*. 2000. P. p. 1132-1137

Koren Y General RMS Characteristics, Comparison with Dedicate and Flexible Systems. *In Reconfigurable Manufacturing Systems and Transformable Factories,* Springer Berlin Heidelberg. 2006, p.p. 27-45

Koren Y., Heisel, U., Jovene, F., Moriwaki, T. Pritschow, G., Ulsoy, G., Van Brussel, H. Reconfigurable Manufacturing Systems. *Annals of CIRP*, 1999. 42(2), 527-540

Lara, O. I., Desarrollo de sistema de pruebas funcionales para motores de combustión interna, Tesis Maestría. Instituto Tecnológico de Chihuahua Enero 2015.

Loya, O. A., Sistema de integración para equipos de manufactura con aplicación móvil. Tesis de Maestría. Instituto Tecnológico de Chihuahua. Junio 2016.

Meharabi, M. G., Ulsoy, A. G., Koren, Y., Heytler, P. Trends and perspectives in Flexible and reconfigurable manufacturing systems, Journal of Intelligent Manufacturing 2002. 135-46

Mehrabi, M. G. Ulsoy, U. G., Y., Reconfigurable manufacturing systems: Key to manufacturing Journal of Intelligent Manufacturing. 2000. 11,403-419

Orona, F., M., Aplicación del Estándar ISA S88 y Bloques Funcionales en el Diseño Reusable del Sistema Informático de Control de Producción en un Ambiente de Producción por Lotes. Tesis Maestría. Facultad de Ingeniería. Universidad Autónoma de Chihuahua. Junio 2015.

Pautasso, C., Wilde, E. Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design. *Proceedings of the 18th international conference on world wide web*.2009, 911-920.

Pokhem, C. Patriotiska Testamente. [En Línea.] Stockholm 1761, Web < https://books.google.com.mx/books>

RAE. Diccionario de la Lengua Española (22nd ed.). 2014. Madrid; Espasa Calpe.

Rodriguez, D., Acosta, J. E. “Aplicación del Paradigma Orientado a Objetos en la Reusabilidad de Código en Controladores Lógicos Programables”, Instituto Tecnológico de Chihuahua, Octubre 2017

Silva, M. *Las Redes de Petri en la automática y la informática*. Madrid, 1982.

Smith, J. S., Brett, A. P. *Simulations as a decision-making tool for real –time control of flexible manufacturing systems, Robotics and Automation*, 1998, Proceedings, 1998 IEEE International Conference on 1998. 586-590.

Su, J. component-based Intelligent Control *Architecture for Reconfigurable Manufacturing Systems*. Director: F. Franc Chen. Tesis de Doctorado. Virginia Polytechnic Institute and State University. Blacksburing, Virginia, USA. 2007

Thramboulidis, K. IEC 61499 in Factory Automation, Int Conf. on Industrial Electronics, Technology & Automation (CISSE-IETA 05). 2005a 1-9.

Van der Aalst, W. M. On the automatic generation of workflow processes based on product structures.*Computers in Industry*. 1999. 39(2). 97-11 DOI: 1.1016/S0166-3615(99)00007-X.

Vyatkin, V. Bridging the Gap between PLC Programming Languages and Distributed Systems. *IEEE Industrial Electronics Magazine* (December). 2009, 40-48.

Womack. J. P., Jones, D. T., Roos, D. *The Machine That Changed The World* (1st Ed.). New York: HarperPerennial. 1991. ISBN: 0-06-097417-6.

Woodbury, R. S. *The Legend of Eli Whitney and Interchangeable Parts*. Technology and Culture. 1960, 1(3), 235-253.

Ye, S. X., Qui, R. G. An Architecture of configurable Equipment connectivity in a Future Manufacturing Information System. In *Computational Intelligence in Robotics and Automations. Proceedings 2003 IEEE Internationals Symposium on*. 2003, p.p. 1144-114.

Zapata, G., Carrasco, E. *Estructuras generalizadas para controladores lógicos modeladas mediante redes de Petri*, 2002

Zapata, G., Quintero, L.F. *Metodología para la generación de código normalizado a partir de modelos en redes de Petri jerárquicas*, Universidad Nacional de Colombia sede Medellín-Colombia, 2007

Zhou, M. *Comparing ladder logic diagrams and Petri nets for sequence controller design.* IEEE transactions on industry applications. 41, (6), 1994.

Zhou, M. *Design of industrial automated systems via relay ladder logic programming and Petri nets.* IEEE transactions on systems, man and cybernetics. 28, (1), 1998.

Zhou, M. *Discrete event control design methods*. 13th IFAC Word Congress, 9, 1996.

Zhou, M. *Petri nets in flexible and agile automation*. NJIT, 1995.