Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 39, pp. 6-10, Oct 2017, Chihuahua, Chih. , México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2017.zip
ISSN 1405-2172

APLICACIÓN DEL PARADIGMA ORIENTADO A OBJETOS EN LA REUSABILIDAD DE CÓDIGO EN CONTROLADORES LÓGICOS PROGRAMABLES.

Acosta-Cano-de-los-Rios, J.E. Rodríguez-Rodríguez David

Instituto Tecnológico de Chihuahua
División de Posgrado e Investigación
jacosta@itchihuahua.edu.mx david_rodriguez_1985@yahoo.com.mx

RESUMEN.

Se presenta una aproximación al paradigma orientado a objetos (POO) en sistemas basados en controlador lógico programable. La reusabilidad de sistemas como característica deseable en los sistemas informáticos, ha sido tema de interés en el área de ingeniería de software y propiciado el desarrollo de tecnología de soporte como librerías de enlace dinámico, incorporación dinámica de código y conceptos tales como acoplamiento débil. Sin embargo, en el área de programación de sistemas basados en PLC ha sido lento su avance y los entornos de desarrollo no han alcanzado tal nivel de madurez. Se encuentran esfuerzos dirigidos al modelo basado en lenguajes orientados a objetos tales como UML, SYSML y XML, entre otros; y herramientas de ingeniería de software para la implementación (codificación) basada en modelo de sistemas PLC. En el presente artículo se describe una solución al problema de instanciación de objetos en lenguaje de PLC, apoyado en lenguaje de alto nivel orientado a objeto. Se muestra el desarrollo basado en VB NET y archivos .L5K (Rockwell), para generar objetos a partir de la implementación de clases. Se logra instanciar objetos en programa PLC, obteniendo reusabilidad de código y acoplamiento débil entre software /hardware del PLC.

Palabras POO, Reusabilidad, Acoplamiento débil.:

ABSTRACT.

An approach to the Object Oriented Paradigm (OOP) for PLC based control systems is presented. System reusability as a desirable characteristic has been an interest topic in the area of software engineering and fostered development of support technology such as dynamic link libraries, dynamic code incorporation and loose coupling concepts. However, in the PLC area, that progress has been slow and development environments have not reached that level of maturity. Some efforts has been reported to PLC system modeling based on object oriented languages such as UML, SysML or XML, as well as software engineering tools for implementation (coding) based on PLC system model. In this article a solution is proposed for object instantiation in PLC languages, supported by high level object oriented language. A development based on VB Net and .L5K files is shown to generate (instantiate) objects from implemented classes. Object instantiation is achieved for PLC programs, obtaining code reusability and loose coupling between PLC software / hardware.

Keywords: POO, PLC, Reusability, PLC Loose Coupling.

1. INTRODUCCIÓN

Durante las últimas décadas la reusabilidad de sistemas informáticos ha sido una característica deseable en sistemas informáticos convencionales; últimamente se ha convertido en un área de interés para los desarrolladores de programas de control basados en PLC. El paradigma orientado a objeto (POO) soporta el reuso de código mediante conceptos tales como clase-instancia, herencia y polimorfismo. Sin embargo, los lenguajes de programación de PLC's actuales soportan tímidamente las características del POO. Las automatizaciones industriales están controladas principalmente por sistemas basados en PLC que estan programados en cualquiera de los lenguajes del estándar IEC61131-3 [1]. El estándar fue el primer paso para la integración posterior de características del paradigma orientado a objetos (OOP) por medio de los bloques de función(FB), sin embargo otras cualidades del OOP como el poliformismo y la herencia no fueron definidos [2]. Trabajos recientes han aplicado UML, SYSML y XML para crear modelos de la automatización y encapsular partes del código que sean fieles al concepto de OOP [3 y 4]. Otro aspecto es el de usar el estándar IEC61499 donde se consideran aspectos de un OOP aplicado a los PLC, aun asi el estándar no fue usado ampliamente por los fabricantes de PLC a excepción de Rockwell por medio del entorno de desarrollo de FB [5], o bien ha sido de interés académico. . La idea se basa en operar al PLC mediante eventos cosa que se busca por medio del estándar (IEC61499) y las aproximaciones por UML, SysML y XML. Lo cual cambia la forma en la que el PLC funciona de una manera cíclica a una dirigida por eventos que ofrece un mejor tiempo de reacción [6]. De esa manera los sistemas se vuelven más reactivos hacia cambios de proceso y electromecánicos. El factor común en este tipo de reportes es una solución basada en un segundo lenguaje como C++, PLCOpen [2] y CodeSys [3] para crear una interfaz de desarrollo en la generación de código o un paquete comercial para transferir los modelos POO a un lenguaje equivalente de PLC.

En las aplicaciones de automatización industrial no es común el uso del POO para facilitar el desarrollo y actualización de programas implementados en PLC, [7]. La falta de soporte en los ambientes de desarrollo comerciales respecto a las

características del POO para el desarrollo de programas de PLC, inhibe su uso en aplicaciones industriales. El objetivo del presente trabajo es desarrollar una forma de aplicar el POO en el desarrollo de programas en PLC con apoyo de lenguaje de alto nivel orientado a objetos operando en computadora personal. La instanciación de objetos facilita la adaptación del sistema a cambios en el entorno y su reusabilidad en diferentes aplicaciones particulares. …

El presente documento se encuentra organizado de la siguiente manera. En la sección dos se describe a manera de marco de referencia una introducción al tema de programación de PLC a partir de un lenguaje de alto nivel. La sección tres muestra una manera de generar código en el programa del PLC a partir de clases implementadas en un lenguaje de alto nivel (VB Net), haciendo énfasis en el acoplamiento débil así obtenido. En la última sección se mencionan las principales conclusiones obtenidas en el desarrollo de la presente investigación.

2. MARCO TEÓRICO.

Los antiguos sistemas de desarrollo RSLogix5000 y RSLogix500 (AllenBradley), presentan lo que se conoce como "*Automation Interface"* que permiten la inserción de código en programas de PLC desde aplicaciones en PC mediante una librería de clases, [8]. Las clases permiten agregar instrucciones en forma programable desde PC, por ejemplo para añadir un renglón, Fig, 1, se hace uso de la clase "*Ladder file object"* que tiene el método "*InsertRungAsASCII"* y cuya sintaxis es:

Res = gLadderFile.InsertRungAsAscii(4, "SOR XIC B3/0 OTE B3/1 EOR")

Fig. 1. Renglón de diagrama de escalera.

Por otra parte en los PLC de Allen Bradley más modernos como el CompactLogix que cuenta con el Software Studio 5000, estas características no están presentes o al menos no se han liberado oficialmente, por lo que las clases y métodos que se encuentran en RSLogix5000 y RSLogix500 no pueden ser aplicadas. Es por ésa razón que se necesita de una estrategia diferente para poder pasar de lenguaje de alto nivel (VB) a la lógica escalera del PLC.

En Studio 5000 la forma en la que se realiza es por medio de texto plano con la extensión **".L5K"** que refleja tanto las características del controlador como son las variables, rutinas, ADDONs y hardware[9]. En este caso, agregar instrucciones al programa de PLC consiste en insertar en el archivo de texto plano rutinas, variables y bloques a una forma o estructura definida (genérica) de programa de control. La estructura genérica del programa en texto plano tipo .L5K se muestra en la Fig. 2.

Para insertar instrucciones en el programa es necesario insertar líneas de texto que representan renglones en el lenguaje de escalera. Las secciones indicadas para tal efecto son **Tag** y **Program**. La sintaxis de las líneas Tag determinan si las etiquetas son memorias o entradas/salidas de los módulos. Para el caso de las memorias se tiene el siguiente ejemplo de sintaxis:

Entrada OF Local:1:I.Data.0 (RADIX := Decimal);
Salida OF local:2:O.Data.0(RADIX:=Decimal);

.

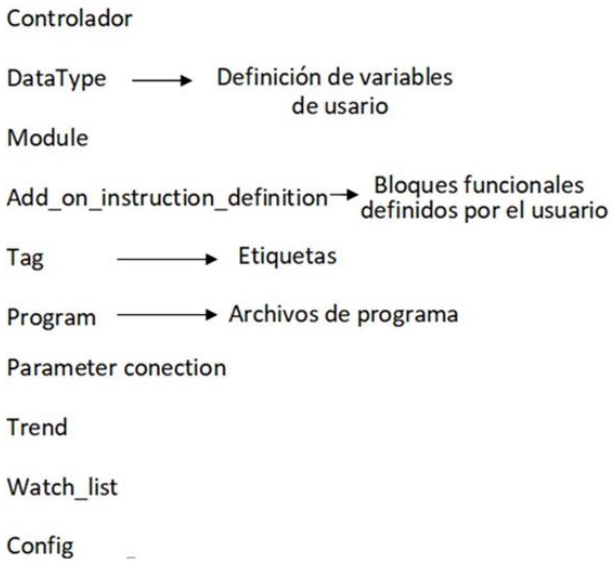

Fig 2. Estructura de texto plano .L5k

En lo que concierne a la rutina del programa se hace de una manera similar, sólo que ahora se insertan líneas que representan renglones en lenguaje escalera, dónde su sintaxis es:

N: XIC(Entrada) OTE(Salida).

A manera de ilustración se muestra a continuación el código de programa simple de encendido de un motor con arranque y paro, Fig. 3.

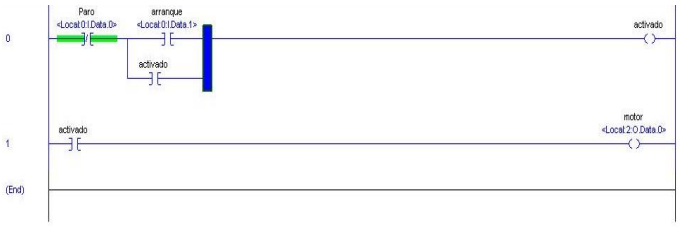

Figura 3. Ejemplo de arranque/paro en diagrama escalera

La representación en texto plano (archivo .L5k) es de la siguiente manera:

```
PROGRAM MainProgram (MAIN := "MainRoutine",
                    MODE := 0,
                    DisableFlag := 0,
                    UseAsFolder := 0)
                TAG
                        Paro OF Local:0:I.Data.0 (RADIX :=
                        Decimal);
                        arranque OF Local:0:I.Data.1 (RADIX
                        := Decimal);
                        activado : BOOL (RADIX := Decimal)
                        := 0;
                       motor OF Local:2:O.Data.0 (RADIX :=
                       Decimal);

                END_TAG

                ROUTINE MainRoutine
                        N:     XIO(Paro)   [XIC(arranque),
                       XIC(activado)]OTE(activado);
                       N:  XIC(activado)OTE(motor);
                END_ROUTINE
```

El archivo .L5K, se abre en el entorno de desarrollo Studio 5000 (Rockwell) como un programa común, para posteriormente cargarse en el PLC (en este caso en un compactlogix). El archivo contiene todas las características necesarias para operar la aplicación en el PLC, (tags, programa, hardware, etc).

3. IMPLEMENTACIÓN DEL CONCEPTO CLASE-INSTANCIA.

El concepto de clase no es soportado en los entornos de desarrollo para PLC, como lo es el caso del soporte en lenguajes de alto nivel en el área de PC. Por lo tanto, en este proyecto se recurre al apoyo de una plataforma en computadora personal (Visual Studio) para la implementación de clases y creación de objetos, y facilitar así la reusabilidad de código en el desarrollo de sistemas basados en PLC.

Retomando el ejemplo del arrancador, el texto que permite definir sus variables y estructura puede considerarse como la clase arrancador, Fig,. 4. Replicando el texto cuantas veces se necesite es posible generar diferentes objetos de tipo arrancador para diferentes entradas y salidas. De tal manera que puede existir la clase “arrancador” y sus instancias o copias “arrancador_1”, “arrancador_2”, “arrancador_3” y así sucesivamente.

3.1. Generación de instancias (objetos).

La incorporación de código en el programa de PLC puede ser apoyado por el concepto de clase presente en lenguajes orientados a objetos.

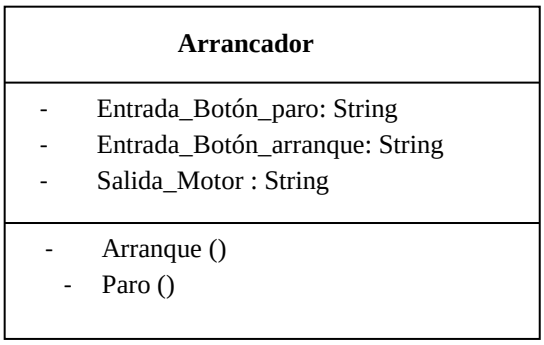

Fig. 4. Clase Arrancador

La idea básica consiste en que la clase implementada en el lenguaje de alto nivel (Vb NET) ofrezca la funcionalidad para generar el texto correspondiente a una instancia de objeto en el archivo del programa del PLC (.L5K). La aplicación en VB agregará texto que representa una lógica en el PLC. La clase tiene los atributos que corresponden a las secciones TAG y program del archivo .L5K. En el caso del arrancador requiere etiquetas (TAGS) de paro, arranque así como salida de motor, junto con lógica de escalera (program). Tales TAG cambian sus valores de variables de acuerdo a la conexión de los botones y motor con el PLC (entradas/salidas). El programa en VB identifica las secciones del archivo de texto para insertar las instancias (texto) en el programa. En la creación del objeto arrancador en VB, el propio objeto abre el archivo del programa PLC, ubica la sección donde insertar el texto respectivo a la instancia arrancador, colocando en el archivo .L5K los TAGS y Program equivalentes al objeto arrancador respectivo. En el caso del ejemplo Arrancador, el texto a insertar por la clase Arrancador sería de la siguiente manera:

TAG
Entrada_botón_paro OF Local:0:I.Data.0 (RADIX := Decimal);
Entrada_boton_arranque OF Local:0:I.Data.1 (RADIX := Decimal);
Salida_Motor OF Local:1:O.Data.0 (RADIX := Decimal);
PROGRAM
N: XIO(Entrada_botón_paro) [XIC(Entrada_boton_arranque), XIC(Salida_Motor)]OTE(Salida_Motor);

Las variables (TAGS) del objeto respectivo (tipo arrancador) son asignadas al momento de su creación, a través de una interfaz de usuario (Ventana). La clase arrancador replica en el archivo .L5K el texto equivalente al objeto arrancador con los atributos propios. De esta manera la programación en VB.Net emula la instanciación en el programa del PLC, haciendo inserciones de texto en la estructura del esquema principal del .L5K.

Suponiendo que se cuenta con un módulo DAQ cuyas entradas sean definidas como “I.0.XX” y sus salidas como “O.1.xx”, es posible asignar los atributos de la clase como

Entrada_Botón_paro, Entrada_Botón_arranque y Salida_Motor en la salida/entrada correspondientes. Por ejemplo, crear tres arrancadores (arrancador_1, arrancador_2 y arrancador_3), en tales instancian de la clase arrancador, se asignan las entradas/salidas a utilizar, como se muestra en la figura 5.

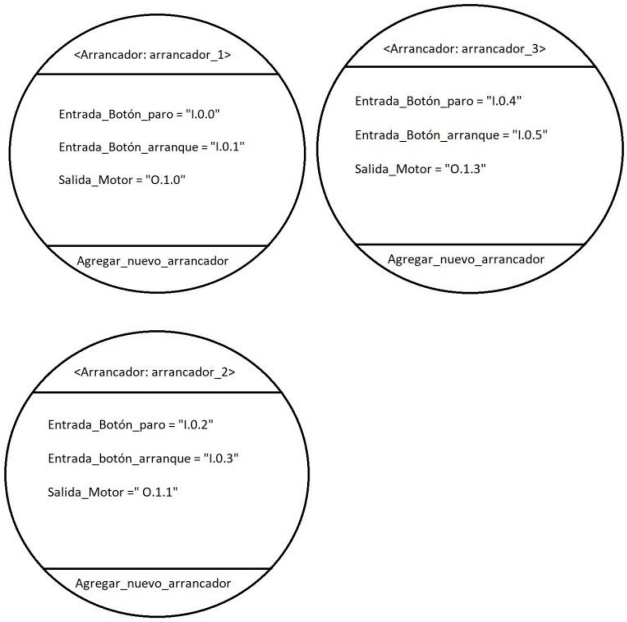

Figura 5. Instanciaciones de la clase arrancador.

3.2. Acoplamiento débil software/hardware.

Una de las principales ventajas en la aplicación del POO es la modularidad de los sistemas, en el caso de sistemas basados en PLC, tal ventaja puede ser extendida a lograr acoplamiento débil entre el software y sensores/actuadores. Tomando como ejemplo el caso del control de un elevador, enfocando el análisis en la implementación del sensor de piso, es decir, un sensor con base en codificador absoluto o bien uno con base en interruptores colocados en cada uno de los tipos. En la Fig. 6 se muestra el modelo de clases del sistema.

Utilizando un encoder absoluto como sensor de piso, el diagrama de escalera para la lectura del piso se muestra en la Fig. 7.

Donde Local:o:i.Data es donde se encuentra conectado físicamente el encoder en las entradas del PLC. El valor de la entrada (salida del encoder) es movido a la variable de memoria Piso en el PLC

El texto a manejar en la clase sensor, correspondiente al diagrama de escalera sería de la siguiente manera:

```
TAG
piso : BOOL (RADIX := Decimal) := 0;
PROGRAM
N: MOVE(  Local:0:I.Data ,piso);
```

Para utilizar diferente tipo de sensor, como pudiera ser sustituir en el sistema al encoder absoluto por un conjunto de sensores colocados en cada uno de los pisos. El detector de piso original lee en la memoria (tag PISO) el código del piso en que se encuentra el elevador. La nueva implementación del sensor debe colocar el código del piso en tal memoria. En la forma de diagrama de escalera, la operación del nuevo sensor sería como se muestra en la Fig. 8:

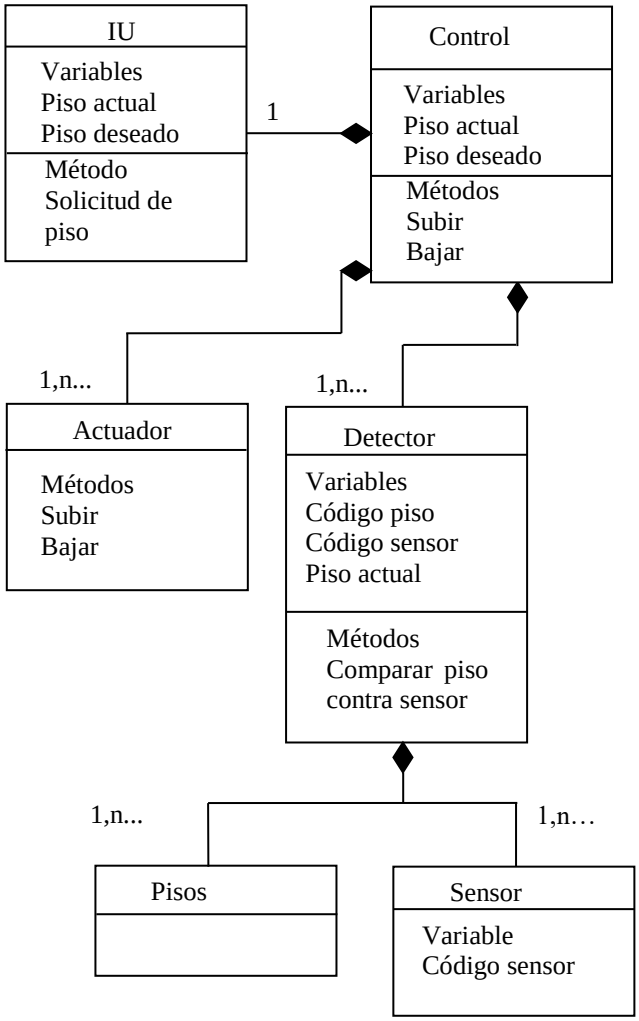

Fig. 6. Modelo de clases del sistema.

El texto a manejar en la nueva clase sensor, correspondiente al diagrama de escalera sería de la siguiente manera:

```
TAG
piso_1 OF Local:0:I.Data.0 (RADIX := Decimal);
piso_2 OF Local:0:I.Data.1 (RADIX := Decimal);
piso_3 OF Local:0:I.Data.2 (RADIX := Decimal);

PROGRAM
N:XIC (piso_1) MOVE(  1 ,piso);
```

N: XIC(piso_2) MOVE(2 ,piso);
N: XIC (piso_3) MOVE(3 ,piso);

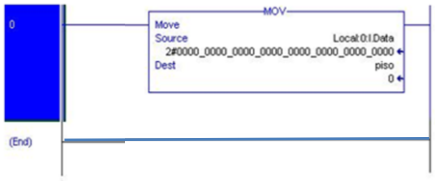

Figura 7. Diagrama escalera de clase sensor

El cambio de tipo de sensor se puede manejar mediante la instanciación del objeto de la nueva clase sensor (sensor2), el cual incorpora en el archivo .5LK el código correspondiente al nuevo diagrama de escalera, utilizando el mecanismo descrito anteriormente, sin requerir mayores cambios en el resto del programa del PLC.

Así, el uso de diferentes tipos de sensores (codificador absoluto o sensor basado en interruptores), puede ser manejado utilizando la clase correspondiente al tipo de sensor. La lógica asociada a un tipo de sensor no contemplado explícitamente en tiempo de diseño, puede ser manejada mediante el uso de la clase implementada para tal efecto. La incorporación de la nueva clase puede lograrse mediante incorporación dinámica de código, esto es, integrar en tiempo de ejecución la dll en donde se ha implementado la lógica para el manejo del nuevo tipo de sensor.

Figura 8. Diagrama escalera de la clase sensor basada en interruptores por piso.

4. CONCLUSIONES.

La adaptación del código en el programa de PLC para su aplicación en un nuevo sistema, puede ser apoyado por el concepto de clase presente en un lenguaje de alto nivel orientado a objetos. La manera aquí descrita presenta las ventajas de creación de una cantidad variable de objetos y el acoplamiento débil entre software y hardware, lo cual aporta reusabilidad al sistema de control. La generación de objetos en el lenguaje de alto nivel (VB. NET) permite sobrellevar la limitante de los lenguajes de PLC existentes, que no permiten la instanciación de objetos (o equivalente) en tiempo de ejecución.

Referencias.

[1] Francesco Basile, Pasquale Chiacchio, Diego Gerbasio, “Progress in PLC Programming for Distributed Automation Systems Control”, 2011 9th IEEE International Conference on Industrial Informatics, Caparica, Lisbon, Portugal, 26-29 July 2011 (conferencia), 06 October 2011(añadido al iee exp)

[2] Manuel Pineda-Sanchez, Marina Pérez-Vázquez, Ángel Sapena-Bañó, Javier Martinez-Román, Juan Pérez-Cruz, Rubén Puche-Panadero, Victor Pérez-Vázquez, “Programmable Logic Controllers (PLC) in the Packaging Industry: an Object Oriented Approach for Developing Control Programs”, Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT), 2014 9th International, Taipei, Taiwan, 22-24 Oct. 2014 (confe), 26 February 2015 (añadido al iee exp)

[3] Martin Obermeier, Steven Braun, and Birgit Vogel-Heuser,Senior Member, IEEE ,” A Model-Driven Approach on Object-Oriented PLC Programming for Manufacturing Systems with Regard to Usability”, IEEE Transactions on Industrial Informatics, Volume: 11, Issue: 3, June 2015, pp 790 - 800

[4] Kleanthis Thramboulidis, Georg Frey , “An MDD process for IEC 61131-based industrial automation systems” , Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, 5-9 Sept. 2011(conf), 24 October 2011(aña iee ex)

[5] D. Servat , F. Chiron , K. Kouiss , T. Meurisse , G. Souchet , “Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on”, 19-22 Sept. 2005, Catania, Italy

[6] Wenbin Dai , Valeriy Vyatkin, “Redesign Distributed PLC Control Systems Using IEC 61499 Function Blocks” , IEEE Transactions on Automation Science and Engineering ,Volume: 9, Issue: 2, April 2012, pp 390 – 401.

[7] Lorenzo Racchetti , Cesare Fantuzzi , Lorenzo Tacconi , Marcello Bonfe , “The PLC UML State-chart Design Pattern”, Emerging Technology and Factory Automation (ETFA), 2014 IEEE, 16-19 Sept. 2014

[8] Rockwell Software, “RSLogix™ Automation Interface Reference Manual”, Rockwell Software Inc, 2001, 2002, pp 14-15, 99-101.

[9] Rockwell Software ,“Reference Manual Logix5000 Controllers Import/Export Rockwell” Automation Publication 1756-RM084V-EN-P , July 2016, pp 23-24, 46