

COMPUERTAS LÓGICAS VISUALIZADAS EN COMUNICACIÓN DE PLC S7-1500 SIEMENS Y AUTOMATION A TRAVÉS DE UN OPC AL COMPUTADOR.

Autores: David Steven Galvis Arévalo 1802584, Eric Xavier Buitrago Forero 1802410, Johan Andrés Sebastián Gómez Espinosa 1802322

Co-Autores: Olga Lucía Ramos Sandoval, Ph.D.

Resumen: En este documento lleva a cabo el resultado de la implementación de las compuertas lógicas AND, OR, NOT y XOR en los distintos lenguajes de programación que permite el Simatic S7-1500, para familiarizarse con la configuración del PLC y el software TIA Portal, y luego comunicarse con el simulador Automation Studio. Esto con el fin de poder controlar un proceso desde un ordenador, ya que esta es una de las principales características de un proceso de automatización industrial y además que permite ampliar el conocimiento en la conexión del controlador con un servidor. Para llevar a cabo todo el desarrollo de la práctica fue necesario consultar en bibliografía y dentro de la opción de ayuda TIA Portal los diferentes lenguajes de programación y su respectiva estructura, declaración de variables y entre otras características propias de cada lenguaje. Posteriormente realizando la comunicación del servidor OPC que permite al PLC compartir datos con el Automation Studio. Y finalmente se llevó a cabo la programación en los distintos lenguajes y todos fueron simulados y con el resultado esperado, también se lleva la conexión entre ambos, que ofrece la gran ventaja de comunicarse a distancia desde un ordenador con el control del proceso que se está desarrollando, el cual es el penúltimo escalón de la pirámide de toda industria automatizada y que por ende es uno de los pilares importantes en la misma.

Abstract: This document shows the result of the implementation of logic gates AND, OR, NOT and XOR in the different programming languages that Simatic S7-1500 allows. This allows to know better the PLC initial configuration and the TIA Portal software and then to communicate with the Automation Studio's simulator. It is with the objective of controlling a process from a computer, this is already one of the main features of an industrial automation process and it allows to expand the knowledge of connecting the controller to a server. To make this development it was necessary to consult in bibliography and the TIAPortal's help option the differentes programming languages and their structures, declaration of variables and some other features of each language. Later setting connection between PLC and Automation Studio by an OPC server. Finally, a connection developed between both previously mentioned were simulated with a good result. It offers the great advantage of communicating with a computer and to process the control. This is the penultimate step of every automatized industry and because of this it's one of the most important pillars of itself.

Introducción

Un controlador lógico programable es un dispositivo electrónico digital que usa una memoria programable para guardar instrucciones y llevar a cabo funciones lógicas, de configuración de secuencia, de sincronización, de conteo y aritméticas, para el control de maquinaria y procesos, en este principalmente se implementan funciones lógicas como primer paso para la aprender la programación [1].

Los PLCs de la familia SIMATIC S7 son basados en el sistema de automatización de Siemens. Estos son la solución a procesos de automatización complejos, su principio, es que cada PLC consiste en una CPU y módulo de entradas y salidas. La CPU contiene el programa creado por el usuario mientras que los módulos I/O permiten el procesos de comunicación con los equipos de campo [2].

En principio, el software estándar STEP 7 solo permitía solo programar los sistemas SIMATIC en solo 3 lenguajes, KOP, FUP y AWL, pero luego se incorporó otro más a la lista, el SCL. El lenguaje KOP consta de un esquema de contactos al igual que un diagrama eléctrico, el lenguaje FUP es basado en funciones lógicas representada en bloques, el lenguaje AWL son listas de instrucciones en escalera, donde la “A” representa una conexión en serie y la letra “O” representa una conexión en paralelo. Y por último el lenguaje SCL está basado en sentencias condicionales “if - else”[3].

Un servidor OPC consiste en un conjunto estándar de interfaces, propiedades y métodos para su uso en aplicaciones de control de procesos y manufactura. Está basado en las tecnologías de Microsoft OLE (Object Linking and Embedding) y COM(Component Object Model). Su principal objetivo es conseguir uniformizar el acceso a los

datos que existen en la industria, de forma que varios clientes puedan acceder a los datos gestionados por un servidor OPC. De esta forma se reduce el desarrollo de varios driver a un solo driver [4]. La forma de crear una nueva base de datos en un OPC IBH [5].

Objetivos

- General:

*Implementar compuertas lógicas en todos los lenguajes de programación del TIA Portal y demostrar su funcionamiento a través de la comunicación con el PLC y Automation Studio.

- Específicos:

*Conocer la configuración de los módulos del PLC dependiendo de su modelo.

*Diseñar las compuertas AND,OR,NOT y XOR en los lenguajes del TIA Portal.

*Comunicar el TIA Portal con el PLC y Automation Studio mediante un OPC.

Desarrollo

Primero se debe conocer el funcionamiento de cada compuerta lógica a implementar y cómo funciona en cada uno de los 4 lenguajes de programación del PLC Simatic S7-1500. Además de la utilización de un OPC como enlace entre el PLC y Automation Studio y cómo configurar el servidor de Automation para que permita la lectura de estas variables a través del bloque de datos.

Materiales:

- PLC Siemens S7-1500
- Realtek RTL8188EU Wireless LAN 802.11n USB 2.0 Network Adapter.
- TIA Portal V13
- OPC IBH

- Automation Studio 6.1

Procedimiento

Primero se crea un nuevo proyecto en el TIA Portal V13 y luego se debe agregar un dispositivo, para este caso se decide usar el controlador PLC S7-1500 en versión de firmware 1.0, la cual es la que poseen los PLC's que se encuentran a disposición en el laboratorio de la universidad.

Ya abierto el TIA Portal se procede a configurar los módulos que acompañan a la CPU, estos se deben configurar en orden y todos se deben encontrar en la misma versión. Se configuran entonces los siguientes módulos:

- Potencia:

PM 70W 120/230VAC

- Módulo de comunicación:

PROFIBUS CM 1542-5

- Módulo de Entradas digitales:

DI 16x24VDC HF

- Módulo de salidas digitales:

DQ 16x24VDC/0.5A ST

- Módulo de entradas análogas:

AI 8xU/I/RTD/TC ST

- Módulo de salidas análogas:

AQ 4xU/I ST

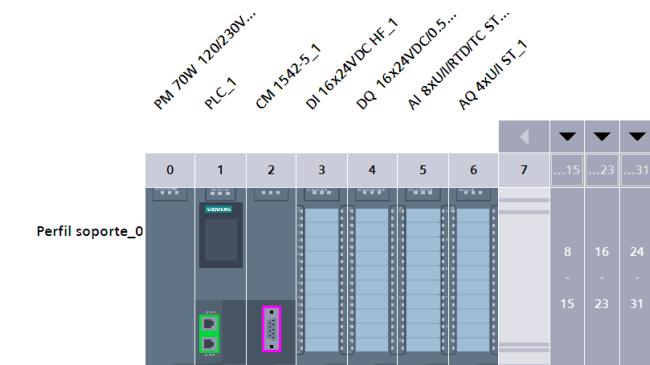


Figura 1: CPU y módulos del PLC.

El dispositivo queda configurado como se observa en la figura 1, donde están todos los módulos con sus respectiva versión de firmware, que para este caso fue 1.0.

Ahora se declaran las variables que se quieren programar en PLC, en el panel que se encuentra en la parte izquierda de la pantalla se selecciona el PLC que estamos usando, la pestaña de “Variables PLC” y luego en “Mostrar todas las variables”. Justo como se demuestra en la figura 2.

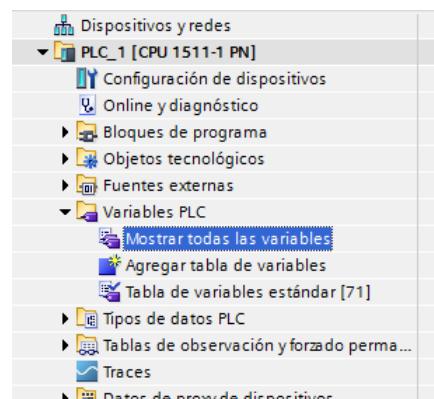


Figura 2: Variables PLC.

Aparece una tabla de variables en blanco, y en la cual se deben declarar las variables y seleccionar un tipo de variable y la dirección de la misma. Al ser compuertas lógicas lo que se quiere probar, se selecciona variable tipo Bool y el identificador de operando, que depende de si la variable es entrada I, salida Q, o un espacio en memoria M. Luego la

dirección y número de bit que refiere a cuál de las 16 entradas o salidas digitales del S7-1500 se quiere asignar la variable recién declarada. Para este caso se declaran solo 2 entradas y 16 salidas digitales para poder observar los 4 lenguajes y las 4 compuertas al mismo tiempo como se observa en la figura 3..

	Nombre	Tabla de variables	Tipo de datos	Dirección	Rrema...	Visibl...	Acces...
1	entrada1	Tabla de variables...	Bool	%I0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	entrada2	Tabla de variables...	Bool	%I0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	KOPAND	Tabla de variables...	Bool	%Q0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	KOPOR	Tabla de variables...	Bool	%Q0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	KOPNOT	Tabla de variables...	Bool	%Q0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	KOPXOR	Tabla de variables...	Bool	%Q0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	FUPAND	Tabla de variables...	Bool	%Q0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	FUPOR	Tabla de variables...	Bool	%Q0.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	FUPNOT	Tabla de variables...	Bool	%Q0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	FUXOR	Tabla de variables...	Bool	%Q0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	AWLAND	Tabla de variables...	Bool	%Q1.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	AMLOR	Tabla de variables...	Bool	%Q1.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	AWLNOT	Tabla de variables...	Bool	%Q1.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	AMLXOR	Tabla de variables...	Bool	%Q1.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
15	SCLAND	Tabla de variables...	Bool	%Q1.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
16	SCLOR	Tabla de variables...	Bool	%Q1.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
17	SCLNOT	Tabla de variables...	Bool	%Q1.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
18	SCLXOR	Tabla de variables...	Bool	%Q1.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 3: Variables asignadas en el PLC.

Teniendo esta lista de variables declaradas, se procede a crear en el main las compuertas AND, OR, NOT y XOR usando los 4 lenguajes de programación del PLC S7-1500 que son KOP, FUP, AWL y SCL.

Primero se quieren simular las 4 compuertas en los 4 lenguajes antes de ser simuladas y comunicadas al Automation Studio.

En el panel de dispositivo del TIA Portal ubicado en la zona izquierda de la pantalla se debe agregar en “Bloques de programa” se debe “Agregar un nuevo bloque” por cada lenguaje que se quiera usar. Donde se puede seleccionar además si el bloque es de orientación, de función, una función o un bloque de datos, como muestra la figura 4.

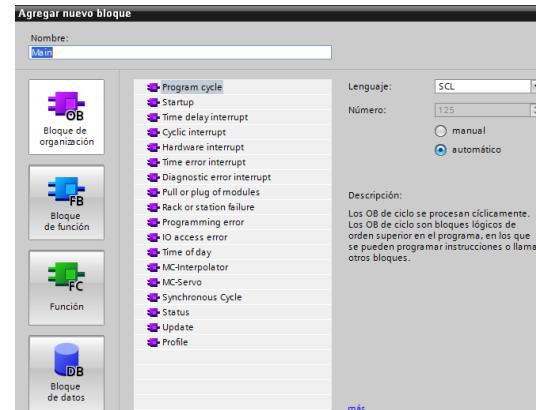


Figura 4: Menú de nuevo bloque.

Para el primer lenguaje KOP, la compuerta AND corresponde a dos contactos normalmente abiertos conectados en serie, como se observa en la figura 5:

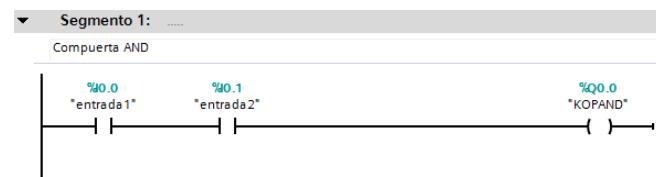


Figura 5: Compuerta AND en lenguaje KOP.

En este mismo lenguaje, para la compuerta OR, se conectan dos contactos normalmente abiertos en paralelo, como se muestra en la figura 6.

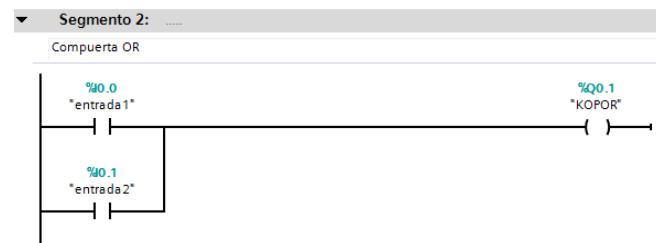


Figura 6: Compuerta OR en lenguaje KOP.

Siguiendo con este lenguaje de programación, la compuerta NOT consta de la entrada asociada a un contacto normalmente cerrado, de manera que el contacto siempre se abra cuando la entrada digital sea 1, lo que salida será 0 y estará negada. Se observa el diagrama en la figura 7.



Figura 7: Compuerta NOT en lenguaje KOP

Y la última compuerta que corresponde a este lenguaje es la XOR, que consta de una entrada 1 como un contacto normalmente abierto en serie con una entrada 2 como un contacto normalmente cerrado, estos en paralelo a un contacto normalmente cerrado de la entrada 1, en serie con un contacto normalmente abierto de la entrada 2. Como se muestra en la figura 8.

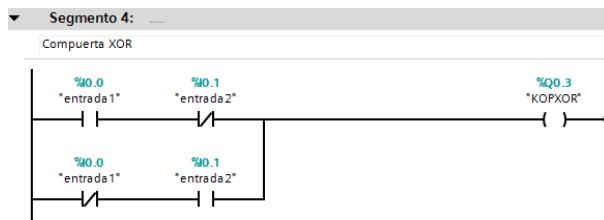


Figura 8: Compuerta XOR en lenguaje KOP

Ahora para el lenguaje FUP, se utilizan los bloques de función, las compuertas AND y OR se muestran en la figura 9.

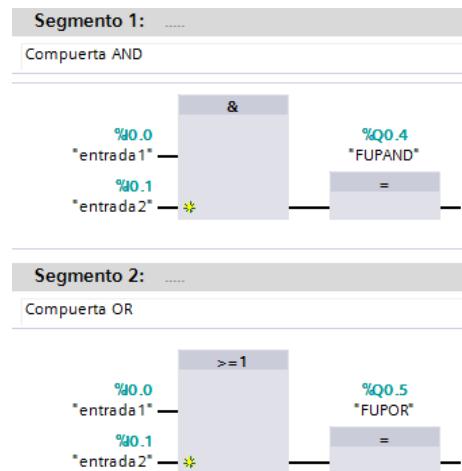


Figura 9: Compuertas AND y OR en lenguaje FUP

Y las compuertas NOT y XOR en lenguaje FUP se observan en la figura 10.

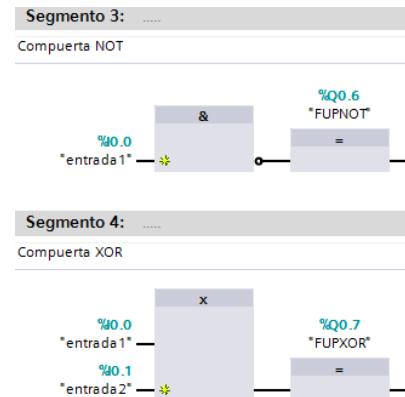


Figura 10: Compuertas NOT y OR en lenguaje FUP.

Ahora para el lenguaje AWL, que son una lista de instrucciones, se recuerda que la letra A representa una conexión serie y O una conexión en paralelo. Se programan las 4 compuertas lógicas nuevamente, entonces: las compuertas AND y OR se expresan en la figura 11.



Figura 11: Compuertas AND y OR en lenguaje AWL.

Y las compuertas NOT y XOR, donde para la primera se niega la entrada 1 mediante la instrucción “not” y en el caso de la compuerta XOR se escribe la letra X, que presenta una compuerta OR exclusiva: se encuentran en la figura 12.



Figura 12: Compuertas NOT y XOR en lenguaje AWL

Para el lenguaje final, que es SCL, es mucho más sencilla su programación, ya que solo requiere una línea de código que consta en igualar la salida a un comando entre entradas, se muestran presentan entonces las compuertas AND,OR,NOT y XOR en este lenguaje, en la figura 13.

```

1 "SCLAND" := "entrada1" AND "entrada2";
2 "SCLROR" := "entrada1" OR "entrada2";
3 IF "entrada1" THEN
4   "SCLNOT" := 0;
5 ELSE
6   "SCLNOT" := 1;
7 END_IF;
8 "SCLXOR" := "entrada1" XOR "entrada2";

```

Figura 13: Compuertas en lenguaje SCL

Ya teniendo todas las compuertas programadas, se compila y luego se carga en el dispositivo, que en este caso al ser primero una simulación, se carga en el dispositivo que permite visualizarse por el PLCSim.

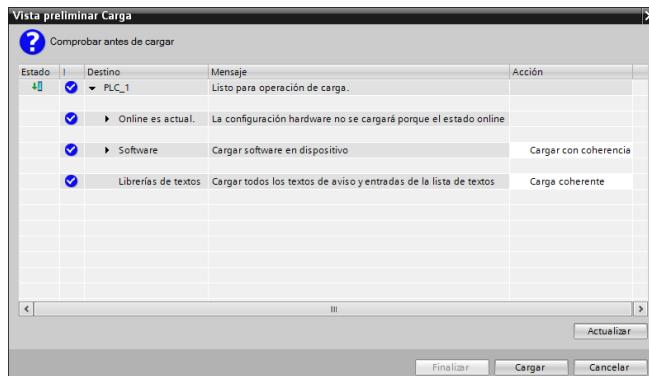


Figura 14: Vista preliminar de cargar programa al dispositivo.

Se establece la conexión online mediante el tipo de interfaz PN/IE e Interfaz que provee el PLCSIM, la cual es PLCSIM S7-1200/S7-1500 y se comprueba la conexión. Como se observa en la figura 15.

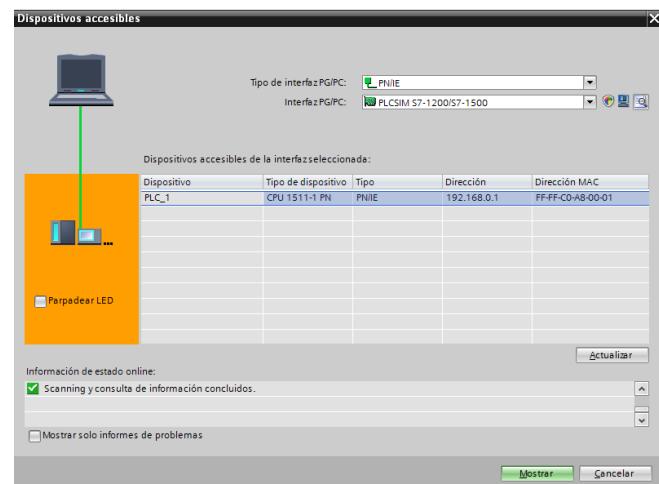


Figura 15: Dispositivos accesibles al TIA Portal

Ahora se puede observar que la simulación corre perfectamente. Ya que en el panel de dispositivo del TIA Portal se encuentran todos los ítems en estado OK, como se ve en la figura 16.

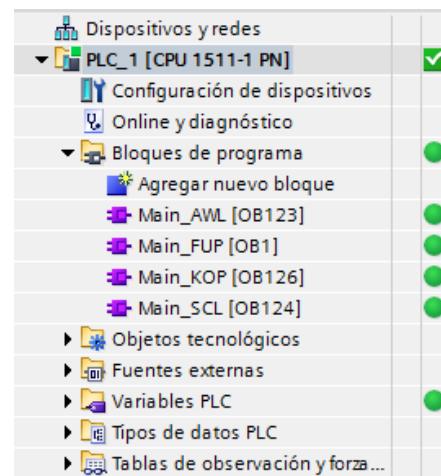


Figura 16: Dispositivos funcionando correctamente en la conexión online.

En la interfaz de PLC SIM se importan las variables programadas del PLC y que se encuentran en el TIA, esto con el fin de observar el pleno funcionamiento del programa desarrollado. Ahora, se prueban todas las compuertas usando solo las 2 entradas creadas.

Cuando ambas entradas se encuentran en estado 0, al ser solo la entrada la que está asociada a la compuerta NOT, esta salida tiene un estado 1 mientras el resto están en estado 0. Como se observa en la figura 17.

Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...	⚡
"entrada1"	%I0.0	B...	FALSE	FALSE		<input checked="" type="checkbox"/>	FALSE
"entrada2"	%I0.1	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPAND"	%Q0.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPOR"	%Q0.1	Bool	FALSE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPNOT"	%Q0.2	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPXOR"	%Q0.3	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPAND"	%Q0.4	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPOR"	%Q0.5	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPNOT"	%Q0.6	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPXOR"	%Q0.7	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLAND"	%Q1.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLOR"	%Q1.1	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLNOT"	%Q1.2	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLXOR"	%Q1.3	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLAND"	%Q1.4	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLR"	%Q1.5	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLNOT"	%Q1.6	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLXOR"	%Q1.7	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE

Figura 17: Simulación en PLCsim cuando ambas entradas son 0.

Ahora, para cuando la “entrada 1” está en estado 1 y la “entrada 2” en estado 0, se muestra el resultado en la figura 18.

Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...	⚡
"entrada1"	%I0.0	B...	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"entrada2"	%I0.1	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPAND"	%Q0.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPNOT"	%Q0.2	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPXOR"	%Q0.3	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPAND"	%Q0.4	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPNOT"	%Q0.6	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPXOR"	%Q0.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLAND"	%Q1.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLOR"	%Q1.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLNOT"	%Q1.2	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLXOR"	%Q1.3	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLAND"	%Q1.4	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLR"	%Q1.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLNOT"	%Q1.6	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLXOR"	%Q1.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE

Figura 18: Simulación en PLCsim cuando entrada 1 = 1 y entrada 2 = 0

Para cuando la “entrada 1” se encuentra en estado 0 y la “entrada 2” en estado 1. Se encuentra el resultado en la figura 19.

Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...	⚡
"entrada1"	%I0.0	B...	FALSE	FALSE		<input type="checkbox"/>	FALSE
"entrada2"	%I0.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPAND"	%Q0.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPNOT"	%Q0.2	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPXOR"	%Q0.3	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPAND"	%Q0.4	Bool	FALSE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPNOT"	%Q0.6	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPXOR"	%Q0.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLAND"	%Q1.0	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLOR"	%Q1.1	Bool	FALSE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLNOT"	%Q1.2	Bool	TRUE	FALSE		<input type="checkbox"/>	FALSE
"AWLXOR"	%Q1.3	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLAND"	%Q1.4	Bool	FALSE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLR"	%Q1.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLNOT"	%Q1.6	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"SCLXOR"	%Q1.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE

Figura 19: Simulación en PLCsim cuando entrada 1=0 y entrada 2 = 1.

Y por último, cuando ambas entradas están en estado 1. Se muestra en la figura 20.

Nombre	Dirección	Form...	Valor...	Forz...	Bits	Forz...	⚡
"entrada1"	%I0.0	B...	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"entrada2"	%I0.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPAND"	%Q0.0	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPOR"	%Q0.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"KOPNOT"	%Q0.2	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"KOPXOR"	%Q0.3	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPAND"	%Q0.4	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPOR"	%Q0.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"FUPNOT"	%Q0.6	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"FUPXOR"	%Q0.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLAND"	%Q1.0	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLOR"	%Q1.1	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"AWLNOT"	%Q1.2	Bool	FALSE	FALSE		<input type="checkbox"/>	FALSE
"AWLXOR"	%Q1.3	Bool	TRUE	FALSE		<input type="checkbox"/>	FALSE
"SCLAND"	%Q1.4	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLR"	%Q1.5	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE
"SCLNOT"	%Q1.6	Bool	TRUE	FALSE		<input type="checkbox"/>	FALSE
"SCLXOR"	%Q1.7	Bool	TRUE	FALSE		<input checked="" type="checkbox"/>	FALSE

Figura 20: Simulación en PLCsim cuando ambas entradas son iguales a 1.

Ya al comprobar que la programación en todos los lenguajes de todas las compuertas fue exitosa, se procede ahora a probar la comunicación del PLC con Automation Studio mediante un OPC.

Simulación de todas las compuertas en todos los lenguajes con variables en bloque de datos.

El procedimiento anterior fue con el fin de comprobar que cada lenguaje fue implementado de la manera correcta y sin errores en su sintaxis.

Ahora se hace un programa donde se prueba el pleno funcionamiento de todas las compuertas y en todos los lenguajes en un bloque de datos, y que permite conectarse con Automation Studio. y en el cual a su vez se pueda controlar desde el TIA Portal o desde el PLC directamente.

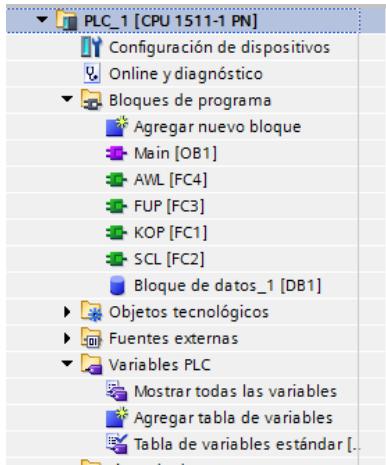


Figura 21: Configuración de 4 funciones y bloque de datos

En el main principal se conecta cada lenguaje como una función, y se declaran las variables como se observa en la figura 22 y las variables del bloque de datos como se observa en la figura 23.

	Nombre	Tabla de variables	Tipo de datos	Dirección	Reman...	Visibl...	Acces...
1	INT 0	Tabla de variable	Bool	%Q0.0		✓	✓
2	INT 1	Tabla de variables e.	Bool	%Q0.1		✓	✓
3	OUT_0_AND	Tabla de variables e.	Bool	%Q0.0		✓	✓
4	OUT_0_OR	Tabla de variables e.	Bool	%Q0.1		✓	✓
5	OUT_0_XOR	Tabla de variables e.	Bool	%Q0.2		✓	✓
6	OUT_0_NOT	Tabla de variables e.	Bool	%Q0.3		✓	✓
7	OUT_1_AND	Tabla de variables e.	Bool	%Q0.4		✓	✓
8	OUT_1_OR	Tabla de variables e.	Bool	%Q0.5		✓	✓
9	OUT_1_XOR	Tabla de variables e.	Bool	%Q0.6		✓	✓
10	OUT_1_NOT	Tabla de variables e.	Bool	%Q0.7		✓	✓
11	OUT_2_AND	Tabla de variables e.	Bool	%Q1.0		✓	✓
12	OUT_2_OR	Tabla de variables e.	Bool	%Q1.1		✓	✓
13	OUT_2_XOR	Tabla de variables e.	Bool	%Q1.2		✓	✓
14	OUT_2_NOT	Tabla de variables e.	Bool	%Q1.3		✓	✓
15	OUT_3_AND	Tabla de variables e.	Bool	%Q1.4		✓	✓
16	OUT_3_OR	Tabla de variables e.	Bool	%Q1.5		✓	✓
17	OUT_3_XOR	Tabla de variables e.	Bool	%Q1.6		✓	✓
18	OUT_3_NOT	Tabla de variables e.	Bool	%Q1.7		✓	✓
19	M_Q_2_NOT_1	Tabla de variables e.	Bool	%MD.0		✓	✓
20	M_Q_2_NOT_2	Tabla de variables e.	Bool	%MD.1		✓	✓

Figura 22: Variables del PLC main.

Bloque de datos_1							
	Nombre	Tipo de datos	Offset	Valor de arranq...	Reman...	Accesible d...	Visible en ...
1	Static						
2	INT_0	Bool	0.0	false		✓	✓
3	INT_1	Bool	0.1	false		✓	✓
4	OUT_0_AND	Bool	0.2	false		✓	✓
5	OUT_0_OR	Bool	0.3	false		✓	✓
6	OUT_0_XOR	Bool	0.4	false		✓	✓
7	OUT_0_NOT	Bool	0.5	false		✓	✓
8	OUT_1_AND	Bool	0.6	false		✓	✓
9	OUT_1_OR	Bool	0.7	false		✓	✓
10	OUT_1_XOR	Bool	1.0	false		✓	✓
11	OUT_1_NOT	Bool	1.1	false		✓	✓
12	OUT_2_AND	Bool	1.2	false		✓	✓
13	OUT_2_OR	Bool	1.3	false		✓	✓
14	OUT_2_XOR	Bool	1.4	false		✓	✓
15	OUT_2_NOT	Bool	1.5	false		✓	✓
16	OUT_3_AND	Bool	1.6	false		✓	✓
17	OUT_3_OR	Bool	1.7	false		✓	✓
18	OUT_3_XOR	Bool	2.0	false		✓	✓
19	OUT_3_NOT	Bool	2.1	false		✓	✓

Figura 23: Variables del bloque de datos.

El main del programa se encuentra a continuación en la figura 24.

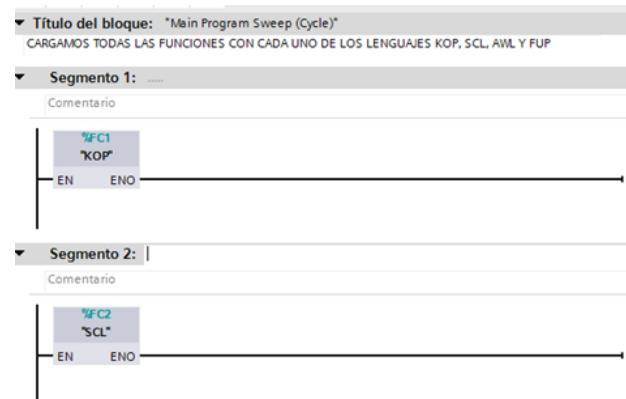


Figura 24: Funciones del bloque en main

Para el lenguaje AWL, la implementación de cada compuerta se da de la siguiente manera como se expresa en las figuras siguientes

Segmento 1:	
AND	
1	A "INT_0"
2	A "INT_1"
3	O
4	A "Bloque de datos_1".INT_0
5	A "Bloque de datos_1".INT_1
6	= "OUT_2_AND"
7	= "Bloque de datos_1".OUT_2_AND
	\$I0.0
	\$I0.1
	\$DB1.DBX0...
	\$DB1.DBX0...
	\$Q1.0
	\$DB1.DBX1...

Figura 25: Compuertas AND en AWL.

Segmento 2:	
OR	
1	O "INT_0"
2	O "INT_1"
3	O "Bloque de datos_1".INT_0
4	O "Bloque de datos_1".INT_1
5	= "OUT_2_OR"
6	= "Bloque de datos_1".OUT_2_OR
	\$I0.0
	\$I0.1
	\$DB1.DBX0...
	\$DB1.DBX0...
	\$Q1.1
	\$DB1.DBX1...

Figura 26: Compuertas OR en AWL.

```
Segmento 4: ....
NOT
1 O "INT 0"
2 NOT
3 = "M_Q_2_NOT_1"
4 //
5 O "Bloque de datos_1".INT_0
6 NOT
7 = "M_Q_2_NOT_2"
8 //
9 A "M_Q_2_NOT_1"
10 A "M_Q_2_NOT_2"
11 = "Bloque de datos_1".OUT_2_NOT
12 = "OUT_2_NOT"
```

		\$I0.0
1	O "INT 0"	\$M0.0
2	NOT	\$DB1.DBX0...
3	= "M_Q_2_NOT_1"	\$M0.1
4	//	\$M0.0
5	O "Bloque de datos_1".INT_0	\$DB1.DBX0...
6	NOT	\$M0.1
7	= "M_Q_2_NOT_2"	\$DB1.DBX1...
8	//	\$Q1.3
9	A "M_Q_2_NOT_1"	
10	A "M_Q_2_NOT_2"	
11	= "Bloque de datos_1".OUT_2_NOT	
12	= "OUT_2_NOT"	

Figura 27: Compuertas NOT en AWL

```
Segmento 3: ....
XOR
1 X "INT 0"
2 X "INT 1"
3 O
4 X "Bloque de datos_1".INT_0
5 X "Bloque de datos_1".INT_1
6 =
7 = "OUT_2_XOR"
8 =
9 = "Bloque de datos_1".OUT_2_XOR
```

		\$I0.0
1	X "INT 0"	\$I0.1
2	X "INT 1"	
3	O	
4	X "Bloque de datos_1".INT_0	\$DB1.DBX0...
5	X "Bloque de datos_1".INT_1	\$DB1.DBX0...
6	=	\$Q1.2
7	= "OUT_2_XOR"	\$DB1.DBX1...
8	=	
9	= "Bloque de datos_1".OUT_2_XOR	

Figura 28: Compuertas XOR en AWL

Luego, la implementación de las compuertas en el lenguaje FUP se muestran en las siguientes figuras.



Figura 29: Compuerta AND en FUP.

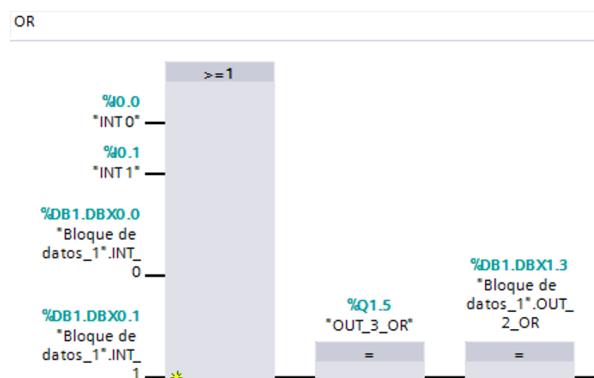


Figura 30: Compuertas OR en FUP.

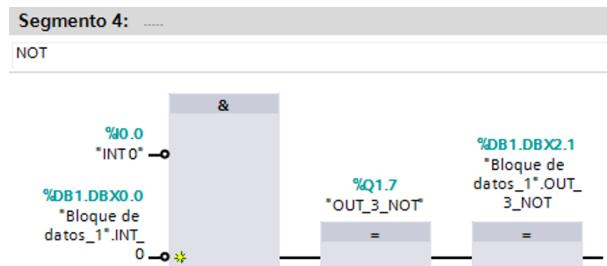


Figura 31: Compuertas NOT en FUP.



Figura 32: Compuerta XOR en FUP

La implementación del lenguaje KOP, se muestra a continuación.

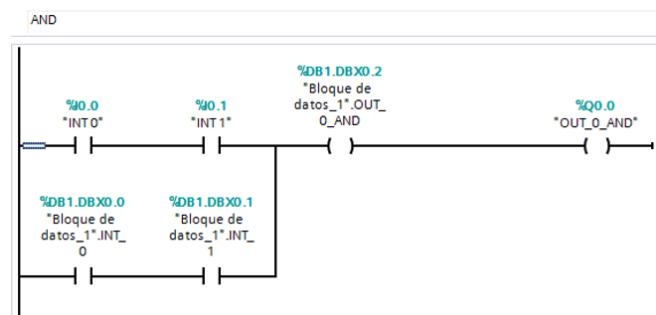


Figura 33: Compuertas AND en KOP.

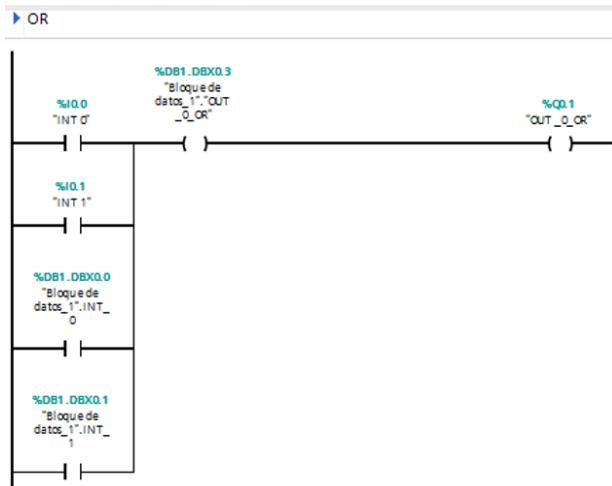


Figura 34: Compuertas OR en KOP.

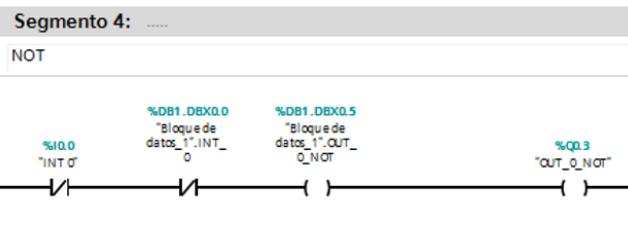


Figura 35: Compuertas NOT en KOP.

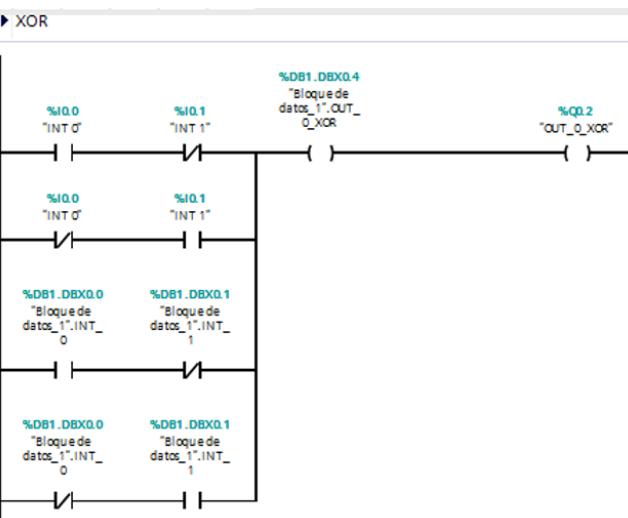


Figura 36: Compuertas XOR en KOP.

Y por último, las 4 compuertas implementadas en el lenguaje SCL, se muestran en la figura 37.

```

1 //AND
2 "OUT_1_AND" :=( "INT_0" AND "INT_1" ) OR ("Bloque de datos_1".INT_0 AND "Bloque de datos_1".INT_1);
3 "Bloque de datos_1".OUT_1_AND := "OUT_1_AND";
4 // OR
5 "OUT_1_OR" := "Bloque de datos_1".INT_0 OR "Bloque de datos_1".INT_1 OR "INT_0" OR "INT_1";
6 "Bloque de datos_1".OUT_1_OR := "OUT_1_OR";
7 //XOR
8 "OUT_1_XOR" := ("INT_0" XOR "INT_1") OR ("Bloque de datos_1".INT_0 XOR "Bloque de datos_1".INT_1);
9 "Bloque de datos_1".OUT_1_XOR := "OUT_1_XOR";
10 //NOT
11 "OUT_1_NOT" := NOT ("INT_0") AND NOT ("Bloque de datos_1".INT_0);
12 "Bloque de datos_1".OUT_1_NOT := "OUT_1_NOT";

```

Figura 37: Compuertas en SCL.

Comunicación con Automation Studio.

Ahora en un nuevo caso se prueba una compuerta AND en lenguaje KOP, se deben crear las variables en un bloque de datos que serán las que se comuniquen al Automation Studio. Para que esta comunicación sea exitosa se debe permitir a la CPU comunicarse con medios externos, desactivando la protección. Esto permite leer y escribir sobre el PLC.

Para realizar este procedimiento se debe ir a “Configuración de dispositivos” y seleccionar la CPU, se debe desplazar hasta la pestaña de protección, y en la parte inferior seleccionar y activar la última casilla. Como se observa en la figura 38.

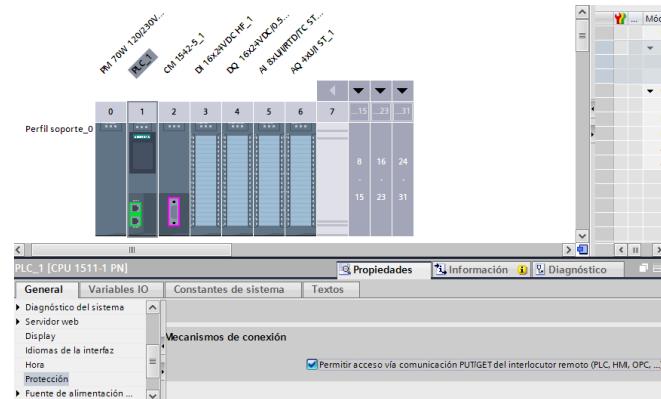


Figura 38: Permitir acceso de comunicación con el OPC.

Luego de esto, se crea un bloque de datos y se configuran las variables las cuales interactúan con el Automation Studio.

Bloque de datos_1								
	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Visible en ...	Valor de au...
1	Static							
2	dbin1	Bool	0.0	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	dbin2	Bool	0.1	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	dbout	Bool	0.2	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figura 39: Variables del bloque de datos.

Teniendo las variables declaradas se crea un bloque de función que nos va a permitir ubicarlo en el bloque “main” posteriormente. Esto con el fin de organizar mejor la estructura de programación del PLC. Dentro de la función se va a encontrar el programa en lenguaje KOP, esto se observa en la figura 40.

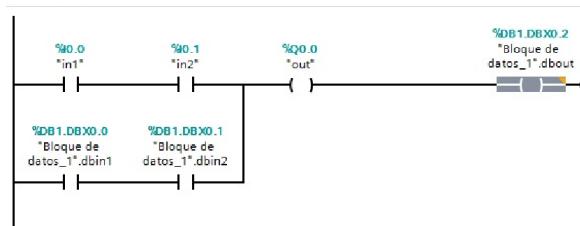


Figura 40: Compuertas AND en KOP

Se programa una compuerta AND la cual permite que, al activar las dos entradas en el Automation, se pueda observar un estado 1 en la salida digital del PLC, y viceversa, al activar las 2 entradas en el TIA Portal o en el módulo físico de entradas digitales del PLC, se pueda observar en el Automation que la salida está en estado 1.

En las propiedades del bloque de datos, se debe también permitir la comunicación de los datos con el OPC, para esto se hace click derecho en el “Bloque de datos” y se selecciona “Propiedades”. En la pestaña de atributos se debe desactivar la casilla de “Acceso optimizado al bloque”. Como se observa en la figura 41.

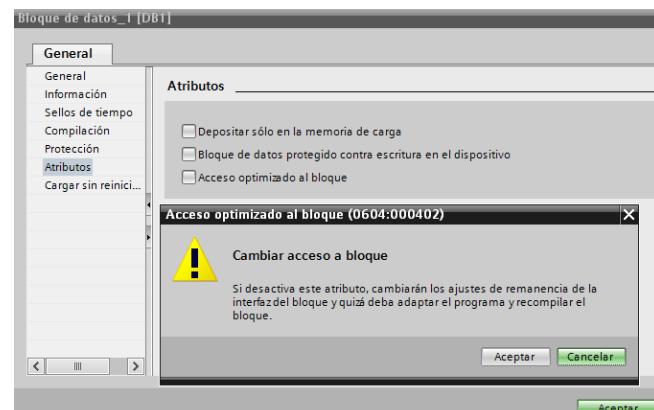


Figura 41: Desactivar la protección del bloque de datos.

Al desactivar la casilla, se debe aceptar cambiar el acceso al bloque, lo que permite que se pueda comunicar sin problemas.

Ahora se debe abrir el OPC[6], y crear un nuevo proyecto.

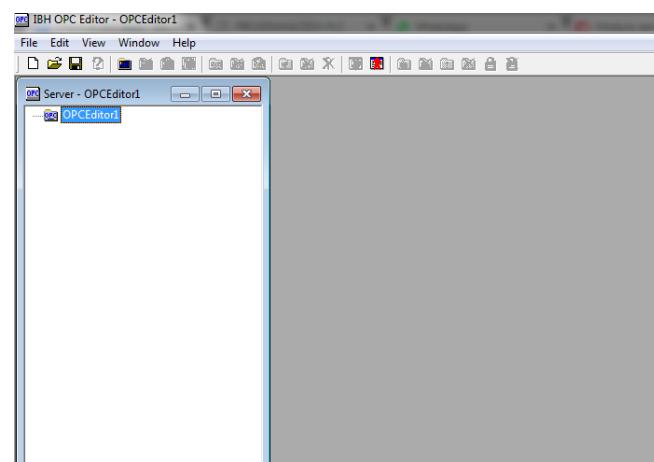


Figura 42: Crear nuevo archivo en el OPC.

Se debe hacer click derecho en OPC Editor e insertar un nuevo PLC, se inserta entonces el S7-1500.

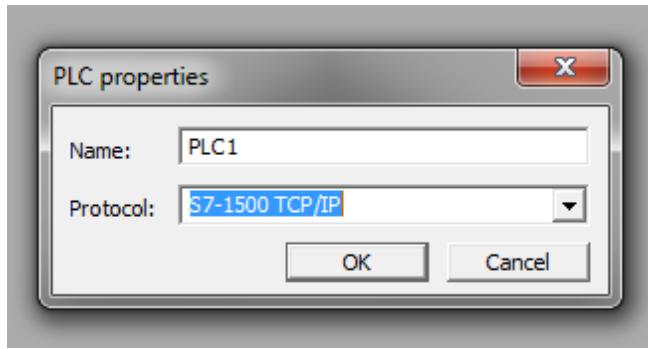


Figura 43: Insertar un nuevo PLC.

Se hace ahora click derecho en el PLC creado y se selecciona Connection Settings.

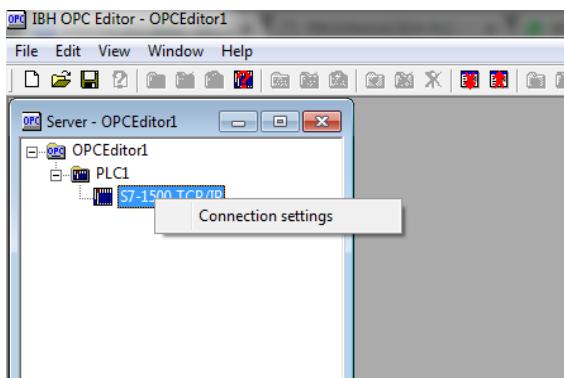


Figura 44: Ajustes de conexión del PLC.

Luego de esto, se selecciona la IP del PLC, en la ventana emergente. Como se muestra a continuación.

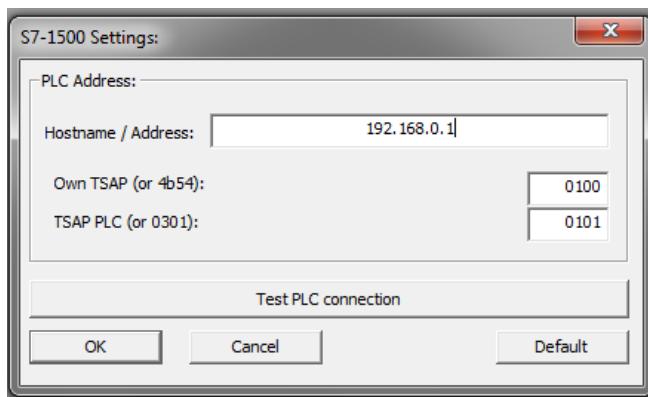


Figura 45: Selección de IP y prueba de la conexión al PLC.

Ahora se deben definir las variables que se quieren comunicar, para esto se selecciona la opción “define variable”, como se muestra en la figura 46.

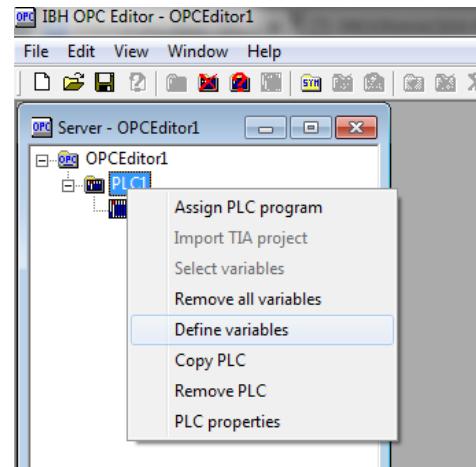


Figura 46: Definir variables.

Se declara un nombre a la variable y luego de esto se selecciona el tipo de variable, que será Booleana y luego la dirección y bit al que corresponda, dependerá de la variable del bloque de datos declarada en el TIA Portal.

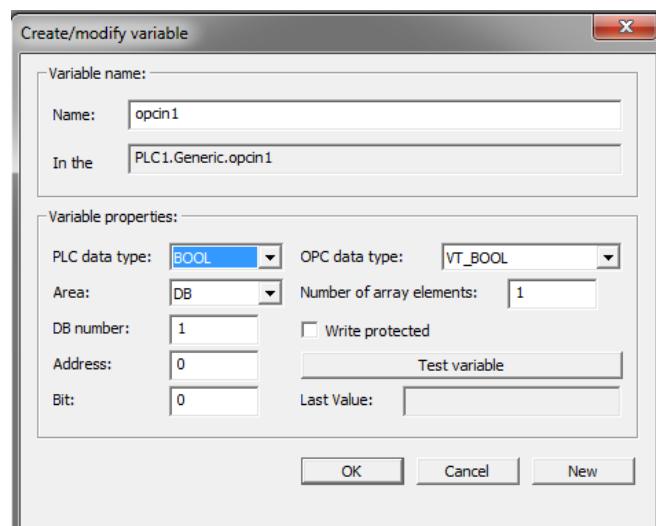


Figura 47: Declarar variables y configurar dirección y bit.

Luego de declarar las 3 variables, 2 entradas y 1 salida, se transfiere el servidor OPC, en el ícono que se señala en la figura 48.

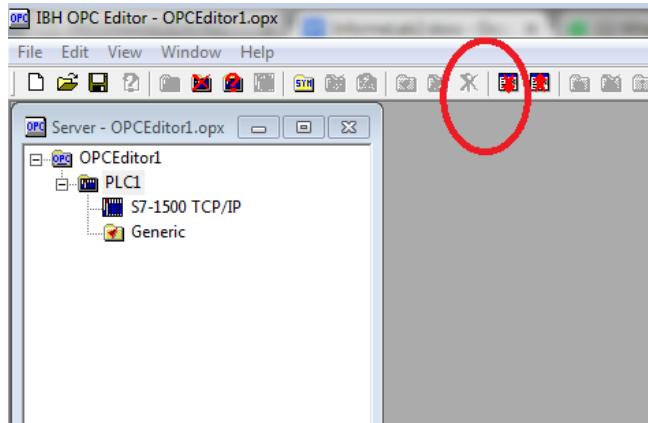


Figura 48: Transferir servidor de OPC.

Se guarda y luego de esto, se selecciona el IBHSoftec.

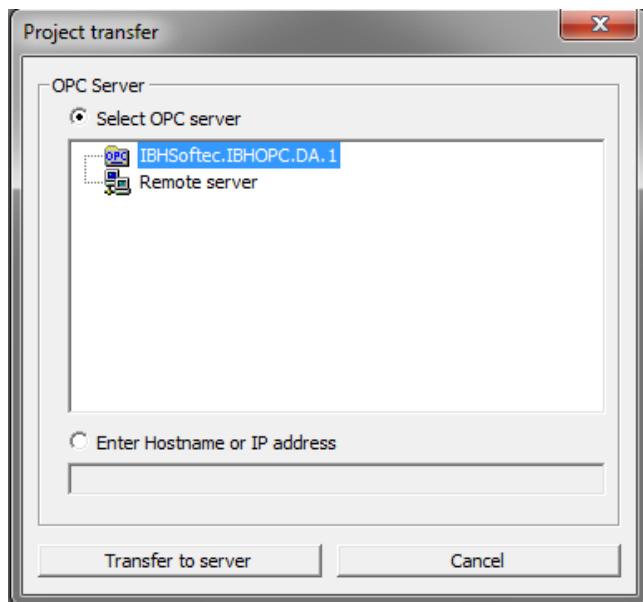


Figura 49: Transferir a IBH.

Cuando aparece esta pestaña, significa que la transferencia fue exitosa y que ahora procede a configurar el OPC Server en Automation Studio.

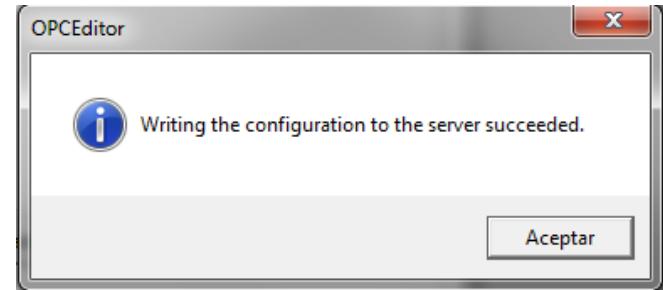


Figura 50: Mensaje de configuración exitosa.

Luego de ya estar todas las configuraciones realizadas en el OPC se abre el Automation Studio. Para la primera entrada se crea un interruptor en Automation, para esto se selecciona como se muestra en la imagen.

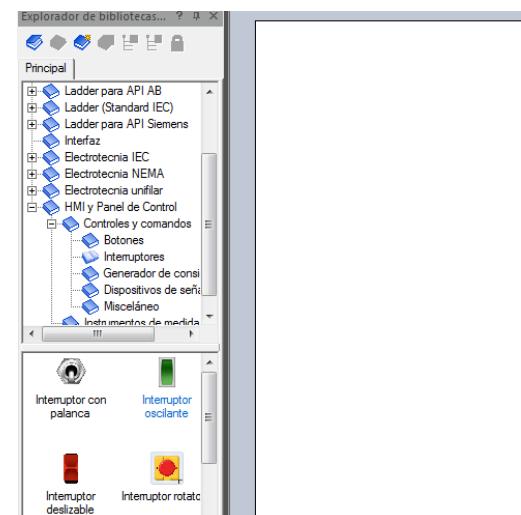


Figura 51: Menú principal Automation Studio.

En el panel de la izquierda se sigue la siguiente dirección: HMI y Panel de control -> Controles y comandos -> Interruptores y selecciona el interruptor con palanca.

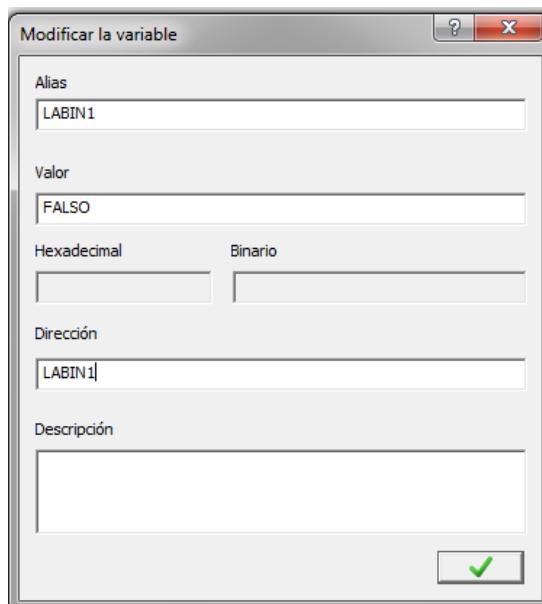


Figura 52: Panel para modificar variables.

Aparece la anterior ventana emergente, donde le damos nombre al interruptor de la interfaz gráfica en el Alias y Dirección. Se da el nombre de “labin1” al interruptor 1 y “labin2” al interruptor 2.

Al hacer doble click sobre cualquiera de los interruptores, se debe abrir una ventana, en la cual se da clic en External Links como se muestra en la figura 53.

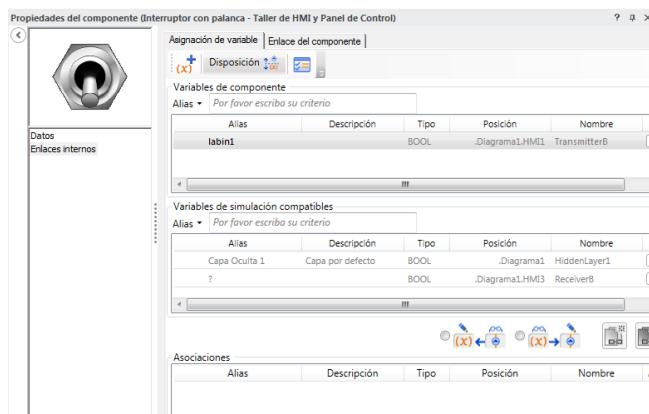


Figura 53: Panel de configuración general de los componentes.

Ahora se crea una variable externa, haciendo click en el botón “[+ (x)]” y en la nueva ventana

se selecciona el tipo de variable “bool” y se da un nombre a la variable, en este caso se llama “AUTIN1”.

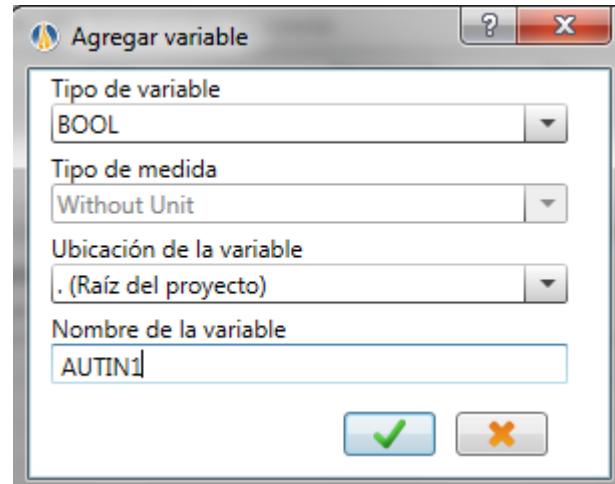


Figura 54: Panel para agregar variable.

Luego de esto, se asocia la variable externa creada con la variable de la interfaz (interruptor o luz) y se crea el enlace realizando el siguiente procedimiento.

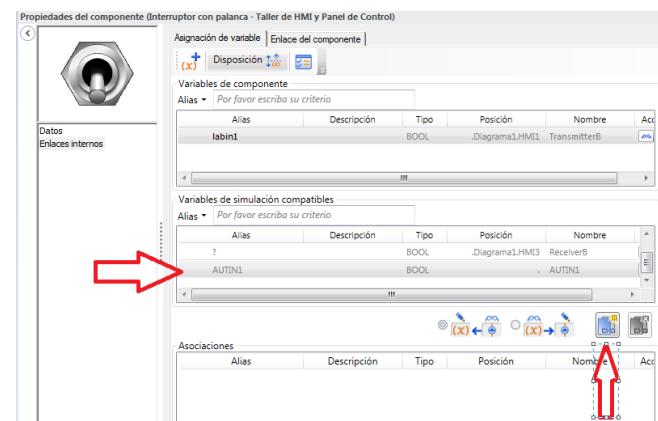


Figura 55: Asignar enlace

Se confirma que el enlace está creado cuando en la parte inferior de la ventana aparece lo siguiente.

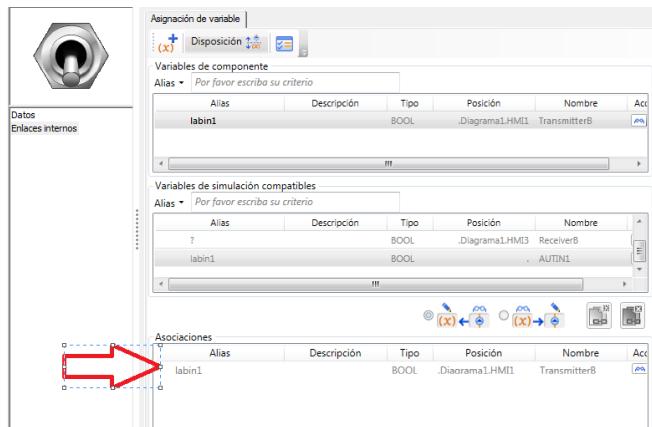


Figura 56: Enlaces asignados.

Este mismo procedimiento se realiza con el segundo interruptor y con la luz LED.

Luego se selecciona la opción “Variable Manager” ubicada en la pestaña Tools, como se muestra en la figura 57.

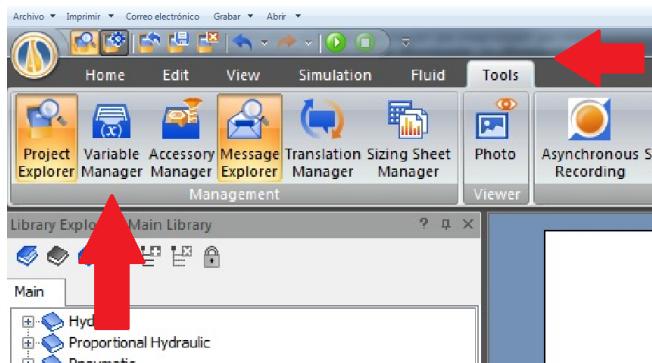


Figura 57: Panel Tools.

Al abrirse la ventana, se hace click en la última opción que es “OPC Client” como se observa a continuación en la figura 58.

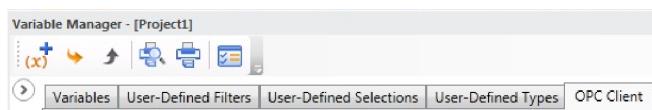


Figura 58: Interfaz de Variable Manager.

Ahora, en la nueva ventana en la pestaña OPC Servers se agrega un nuevo Server, que es el IBH, fabricante del OPC usado. Se añade además un grupo que será el que provenga del OPC IBH,

luego en ítems se selecciona el grupo ya creado y finalmente en la pestaña Links o enlaces se enlazan las variables del Automation Studio con las creadas en el OPC. Se deben enlazar como variables de lectura y escritura y crear el enlace, como se muestra en la figura 59.

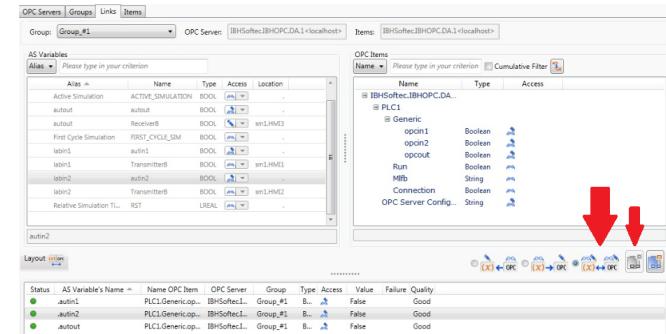


Figura 59: Ventana de enlace de variables.

Finalmente al realizar este procedimiento, se tiene la comunicación del PLC con el Automation Studio, y se interactúa para comprobarla. Dejando ambas entradas en estado 0, el bombillo no enciende y luego al activar ambas, este se coloca en estado 1, como se observa en la figura 60.

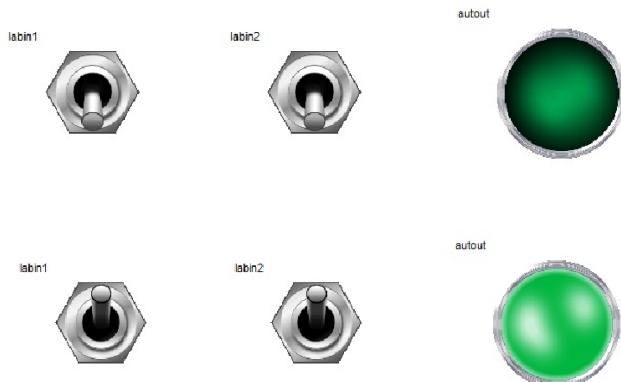


Figura 60: Entradas y salidas de Automation Studio.

En la figuras 61 y 62 se observa que la salida digital Q0.0 del PLC se activa pese a que sus

entradas digitales estén en 0, esto es debido a que se activaron desde el Automation Studio.



Figura 61: Interruptores del PLC.

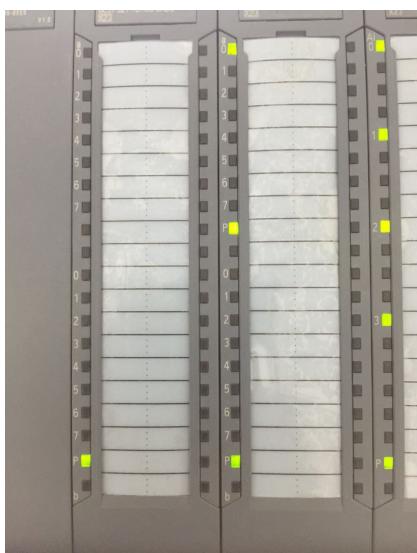


Figura 62: Entradas y salidas digitales del PLC.

Mostrando finalmente la comunicación entre el PLC y el Automation Studio.

Conclusiones

*Se puede concluir que para realizar una comunicación completa, es decir con visualización en tiempo real, se deben incluir todos los parámetros relacionados con el PLC, ya

sean sus módulos, su protocolo de comunicación o su versión del firmware.

*Para conocer la versión de cada uno de los módulos es necesario saber cual es la versión en la que se encuentra configurado el PLC.

*El componente de OPC dentro de la comunicación ya es estandarizado lo cual da una facilidad para la comunicación entre la parte MES de una compañía y su red de control y campo.

*La implementación de las diferentes compuertas lógicas se desarrolló sin esfuerzo, debido a que cada lenguaje ya cuenta con un comando predeterminado para cada una de ellas.

Referencias:

[1]W. Bolton, Mecatrónica: sistemas de control electrónico en ingeniería mecánica y eléctrica, Editorial Alfaomega, 2 ed, pp 423 - 431.

[2]C. Dinis, G. Popa, A. Iagar, “Control of mechanism for pushing trucks using Siemens PLC”, 2016 International conference on applied and theoretical electricity (ICATE), 2016.

[3]P. Mengual, STEP 7: Una manera fácil de programar PLC de Siemens. Editorial Marcombo, pp 72-84.

[4]R. Alves, J. Normey-Rico, A. Merino, C. de Prada, “Un SCADA vía OPC aplicado a una planta piloto”, 2do congresso brasileiro de P&D em Petróleo & Gás, 2003.

[5]http://ftp.softwaretoolbox.com/support/IBH/SiemensUSBMPI-IBHOPC_0208.pdf

[6]https://www.ibhsoftec.com/epages/63444704.sf/en_GB/?ObjectPath=/Shops/63444704/Products/3204