PREPRÁCTICA

Tema: Instrucciones Add-On y subrutinas

1. Objetivos

1.1. Objetivo general

Elaborar programación ladder, bloques de instrucciones y texto estructurado en subrutinas utilizando el controlador micro850 y el software Connected Component Workbench para la realización de aplicaciones industriales.

1.2. Objetivos específicos

1. Realizar funciones de bloques definidas por el usuario (UDFB) mediante programación de texto estructurado de instrucciones de comparadores y sentencias booleanas.
2. Utilizar funciones definidas por el usuario (UDF) mediante programación de bloques de instrucciones.
3. Integrar UDFB y UDF en el programa principal del proyecto para la realización de aplicaciones industriales.

¿Qué actividades realizaran?

1. Integrar al menos cuatros UDFB/UDF en el programa principal de cualesquiera de los proyectos presentados en las prepráctica anteriores utilizando lenguaje de programación escalera, diagrama de bloques de funciones o texto estructurado con todos los compañeros del paralelo.

2. Elaborar y simular una HMI en FactoryTalk View Studio.

¿Como lo realizaran?

Programas por utilizar:

- Connected Components Workbench -CCW (se recomienda versión 13)
- RsLinx Classic
- Simulador micro850
- Factory I/O
- FactoryTalk View

¿Cuáles son los entregables de la prepráctica?

Realizar una infografía de manera individual/grupal (pareja) de las actividades a desarrollar en formato A3 (horizontal o vertical), el cual debe tener los siguientes elementos:

- Un tema claro y conciso: Es decir, comunicar un tema específico de manera clara y fácil de entender.
- Datos relevantes y precisos: La información presentada en la infografía debe ser precisa y relevantes a su tema.
- Diseño atractivo y llamativo: La infografía debe ser visualmente atractiva y llamar la atención del espectador.
- Estructura fácil de seguir: La información debe presentarse en una estructura clara y fácil de seguir para que el espectador no tenga problemas para entender su contenido.
- Fuentes y referencias: La infografía debe incluir las fuentes y referencias utilizadas para obtener la información presentada en ella.

Nota: Todas las prepráctícas serán entregados en el aula virtual en PDF, hasta la semana de la práctica, las faltas ortográficas serán penalizadas, así como la copia ya sea con otros reportes o de internet..

Material de apoyo

Seguir el siguiente orden para la realización de la prepráctica.

- Video: Realizar subrutinas en CCW
- Anexo: Solución del ejercicio de subrutina
- Anexo: Procedimiento de la creación de UDFB en CCW

Anexo - Marco teórico

User-Defined Function

Funciones definidas por el usuario (UDF) son útiles para reutilizar la lógica del programa y hacer que sus programas sean más legibles. Es importante saber que se utiliza para un cálculo simple que requiere solo una salida.

Las UDF son similares a una subrutina ya que una UDF tiene parámetros de entrada y un único parámetro de salida. Las UDF no pueden acceder a las variables locales en el programa de llamada. Las variables locales en el programa de llamada deben pasarse a la UDF como parámetros de entrada.

Características de UDF

- Un parámetro de entrada definido por el usuario no se puede usar para habilitar o deshabilitar las UDF porque el parámetro de entrada solo puede habilitar o deshabilitar las instrucciones dentro de la UDF.
- Para habilitar o deshabilitar la ejecución de un UDF, seleccione la casilla de verificación EN / ENO en la ventana Selector de bloque de instrucciones para el UDF especificado. Por ejemplo, cuando EN es FALSO, el UDF no se ejecuta y los parámetros de salida no se sobrescriben.
- Si el programa que realiza la llamada ejecuta un UDF más de una vez por exploración del programa, no se recomienda utilizar una instrucción que requiera más de una exploración del programa para finalizar la ejecución. Esto incluye instrucciones que retienen el estado entre los escaneos del programa, tales como temporizador, movimiento, mensaje e instrucciones de contador.

User-Defined Function Block

Es un programa definido por el usuario que puede ser empaquetado dentro de un bloque de instrucciones. Este bloque puede reutilizarse dentro de un proyecto. Además, cuando se añade una UDFB se selecciona el lenguaje de programación vasado en el tipo de aplicación que se está desarrollando. Por ejemplo, el diagrama escalera para ejecutar lógicas booleanas simple, temporizadores y contadores. Diagrama de bloques y texto estructurado pueden ser más eficiente para procesos con instrucciones más avanzadas en esos lenguajes.

Ventajas de utilizar user-defined function blocks (UDFBs)

- *Reusar código*

  Usar UDFBs para algoritmos que se usan varias veces en el mismo proyecto. Añadir el código dentro de un UDFB para hacerlo modular y fácil de reutilizar.

- *Usar UDFB en lugar de UDF*

  Para cálculos complejos que tienen múltiples salidas.

Al guardar valores de variables locales de ejecución en ejecución (se requiere guardar estado).

Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que un UDF porque un UDFB en un proyecto no existe en un programa hasta que se instancia como una variable.

- *Proporcionar una interfaz más fácil de entender*

  Coloque algoritmos complicados dentro de un UDFB y luego proporciona una interfaz que sea más fácil de entender al mostrar solo los parámetros esenciales o requeridos.

  Reduzca el tiempo de desarrollo de la documentación insertando comentarios en cualquier lugar del UDFB. Los comentarios son solo para fines de documentación y el programa no actúa sobre ellos.

- *Simplificar el mantenimiento*

  Simplifica el mantenimiento del código porque la lógica de UDFB monitoreada en Connected Components Workbench muestra los valores de entrada y salida relativos a la instancia específica del UDFB.

- *Fácil restablecer valores iniciales de instancias*
  Use la ventana Refactorización para restablecer los valores iniciales para instancias de bloques de funciones definidas por el usuario

Utilizar un User-Defined Function en vez de un User-Defined Function Block

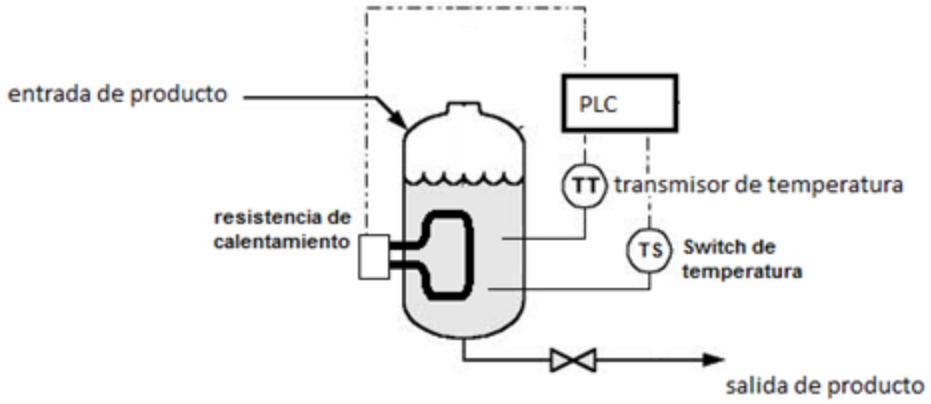
- Para un cálculo simple que requiere solo una salida como Y = MX + B
- Para instrucciones sin estado que no requieren guardar valores de variables locales de ejecución. Ejemplo: SIN es una instrucción sin estado.
- Cuando el parámetro de salida no requiere una matriz o tipo de datos estructurados.
- Siempre que sea posible porque un UDF consume menos memoria.
- Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que un UDF porque un UDFB no existe en un programa hasta que se instancia como una variable.

Utilice un bloque de funciones definido por el usuario en lugar de una función definida por el usuario

- Para cálculos complejos que tienen múltiples salidas.
- Al guardar valores de variables locales de ejecución en ejecución (se requiere guardar estado).
- Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que un UDF porque un UDFB en un proyecto no existe en un programa hasta que se instancia como una variable.
- Cuando el parámetro de salida requiere una matriz o un tipo de datos estructurados.
- Cuando se envía más de un mensaje simultáneamente, un UDFB podría ser una mejor opción que un UDF. Cuando un UDF contiene una instrucción de mensaje como MSG_CIPGENERIC, el UDF solo envía un mensaje a la vez, incluso si se llama al UDF varias veces por exploración de programa.

Anexo - Ejercicio Propuesto

Diseñe la programación de un sistema de control de temperatura con histéresis (ON-OFF) utilizando el simulador del Micro850, con una salida y tres entradas analógicas de 0 -10V, que cumpla con las siguientes características:



a. Se dispone de solo un pulsador para marcha y paro, el mismo que permite encender y apagar al sistema.

b. Se cuenta con una luz piloto que indica si el sistema está en marcha o no.

c. Utilice una entrada del PLC para conectar el transmisor de temperatura, que entrega un rango de lectura de (60-200°C) que corresponde (2-10V).

d. Utilice otra entrada del PLC para configurar el Set-point y la Histéresis a través de un potenciómetro, útiles para el control de la resistencia de calentamiento, y use dos interruptores para su selección (01→Set-point; 11→histéresis).

e. Proponga usted los valores a criterio sabiendo que: los valores del Set-point (60-180°C) que corresponde a (0-10V); y la histéresis (2-10°C) corresponde (0-5V).

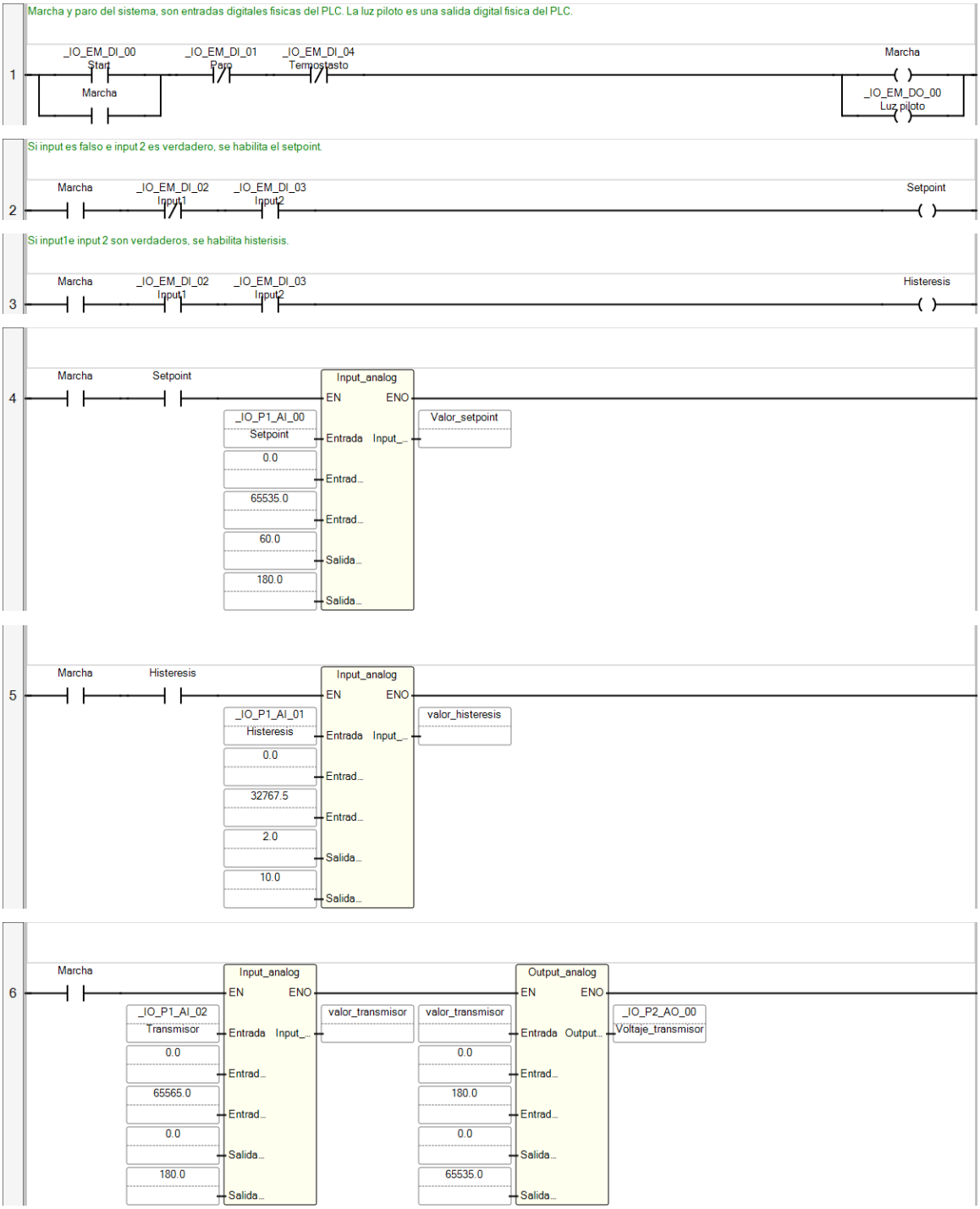
f. El transmisor de temperatura es de 0-180°C que corresponde a 0 -10 V y también dicha lectura será reflejada en una salida analógica.

g. Para protección del proceso se cuenta con un termostato (switch de temperatura) que apagará el sistema de forma inmediata.

h. La salida del PLC Resistencia de Calentamiento se enciende cuando el transmisor de temperatura tiene una lectura menor a: (Set point de temperatura – Histéresis/2) y se mantiene encendida hasta que la lectura del transmisor de temperatura sea mayor a: (Set point de temperatura + Histéresis/2).

Solución del ejercicio propuesto

Controller.Micro850.Micro850.Practice6_JC



POU Practice6_JC

The POU defines 7 variable(s).

Variable Marcha

(* *)

Direction: Var

Data type: BOOL

Attribute: Read/Write

Variable Setpoint

(* *)

Direction: Var

Data type: BOOL

Attribute: Read/Write

Variable Histeresis

(* *)

Direction: Var

Data type: BOOL

Attribute: Read/Write

Variable Valor_setpoint

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Variable valor_histeresis

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Variable Formula_1

(* *)

Direction: Var

Data type: Formula

Attribute: Read/Write

Variable valor_transmisor

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Controller.Micro850.Micro850.Formula

```
H:=(Histeresis/2.0);
valor1 := (Setpoint - H) ;
valor2 := (Setpoint + H);
```

```
if (transmisor< valor1) then
        resistor :=TRUE;
elsif (transmisor >=valor1 and (transmisor<valor2)) then
        resistor :=TRUE;
else
        resistor := False;
END_IF;
```

POU Formula

The POU defines 7 variable(s).

Variable Transmisor

(* *)

Direction: VarInput

Data type: REAL

Attribute: Read

Variable Setpoint

(* *)

Direction: VarInput

Data type: REAL

Attribute: Read

Variable Histeresis

(* *)

Direction: VarInput

Data type: REAL

Attribute: Read

Variable resistor

(* *)

Direction: VarOutput

Data type: BOOL

Attribute: Write

Variable valor1

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Variable valor2

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Variable H

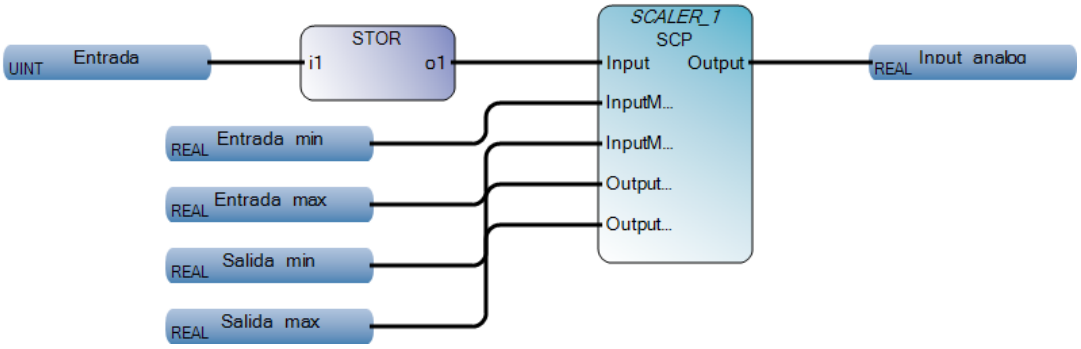
(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Controller.Micro850.Micro850.Input_analog



POU Input_analog

The POU defines 8 variable(s).

Variable SCALER_1

(* *)

Direction: Var

Data type: SCALER

Attribute: Read/Write

Variable variable

(* *)

Direction: Var

Data type: REAL

Attribute: Read/Write

Variable Entrada

(* *)

Direction: VarInput

Data type: UINT
Attribute: Read

Variable Entrada_min

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Entrada_max

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Salida_min

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Salida_max

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

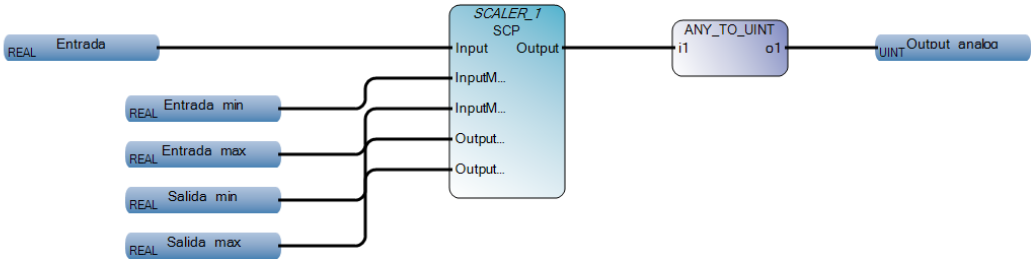
Variable Input_analog

(* *)
Direction: VarOutput

Data type: REAL

Attribute: Write

Controller.Micro850.Micro850.Output_analog



POU Output_analog

The POU defines 7 variable(s).

Variable SCALER_1

(* *)

Direction: Var

Data type: SCALER

Attribute: Read/Write

Variable Entrada

(* *)

Direction: VarInput

Data type: REAL

Attribute: Read

Variable Entrada_min

(* *)

Direction: VarInput

Data type: REAL
Attribute: Read

Variable Entrada_max

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Salida_min

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Salida_max

(* *)
Direction: VarInput
Data type: REAL
Attribute: Read

Variable Output_analog

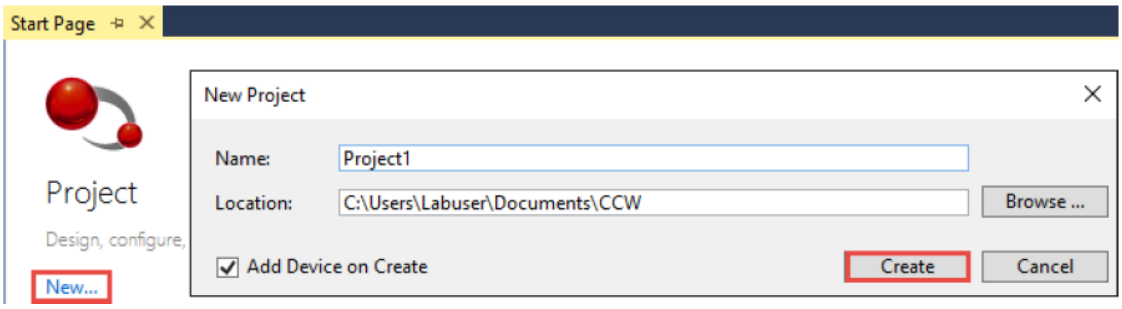
(* *)
Direction: VarOutput
Data type: UINT
Attribute: Write

Anexo

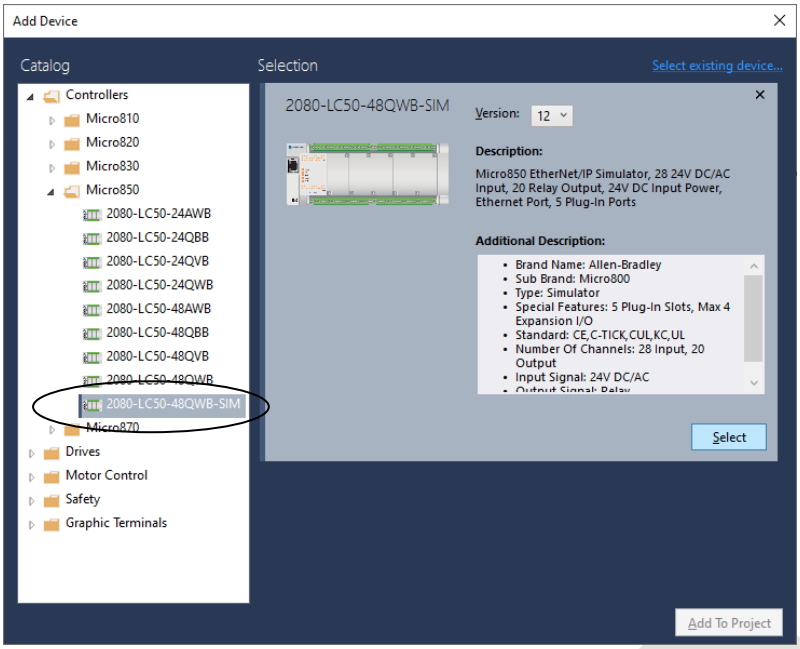
Procedimiento de la creación de UDFB en CCW

Crear un proyecto en Connected Component Workbench y agregar un controlador Micro850

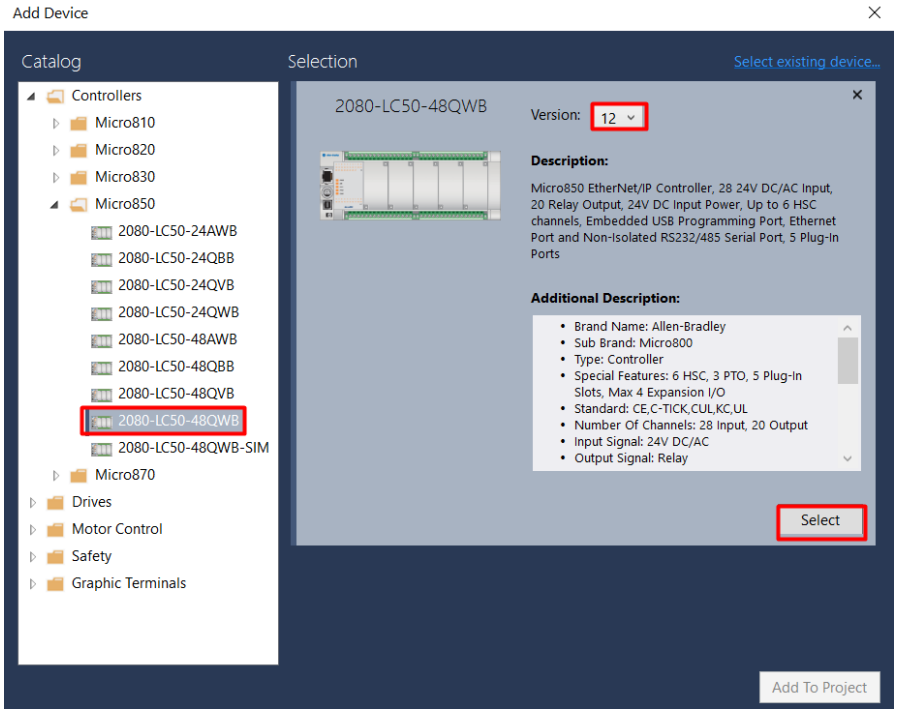
1. Abrir el Software **Connected Component Workbench** (CCW), doble clic al icono en el escritorio.
2. Clic en **New** de la pestaña **Start Page.** Escoger un nombre y localización del proyecto, luego dar clic en **Create**.



3. Escoger el dispositivo con el cual se trabajará en el proyecto, en este caso se expande la carpeta **Controllers** y luego la serie **Micro850** seleccionando la opción del PLC simulado. Luego, dar clic en “Seleccionar” y procedemos a “Agregar al proyecto”.

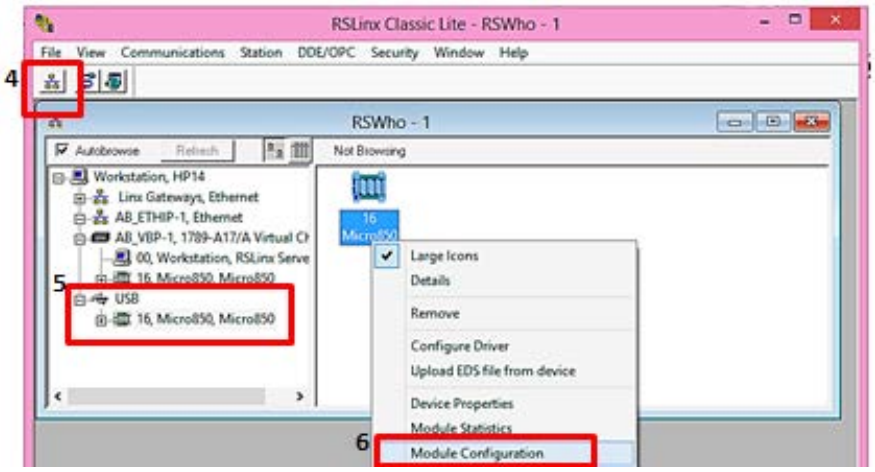


En caso de realizarse la práctica en el laboratorio, se expande la carpeta **Controlllers**, luego la carpeta **Micro850** y se escoge el número de catálogo **2080-LC50-48QWB** cuya versión de firmware del dispositivo debe coincidir con el controlador físico.

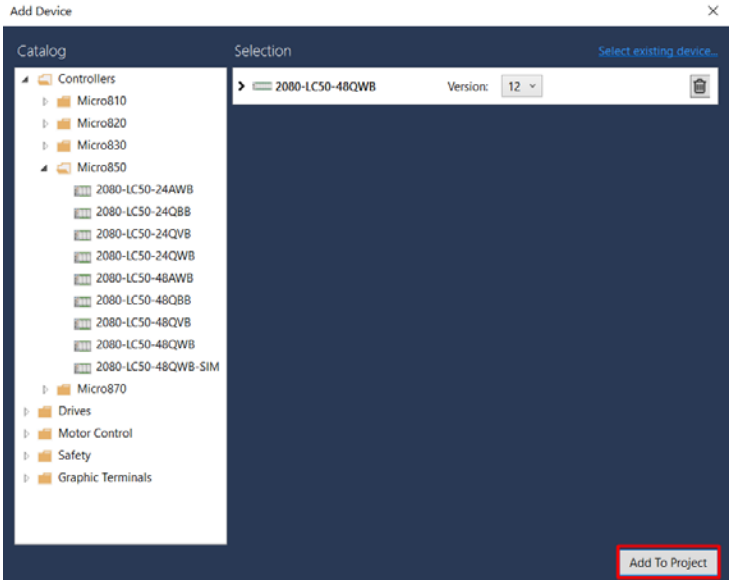


Si se desea verificar por software la versión del equipo, abrir **RSLinx Classic**, seleccionar **RSWho** donde aparecerán los dispositivos conectados a la computadora.

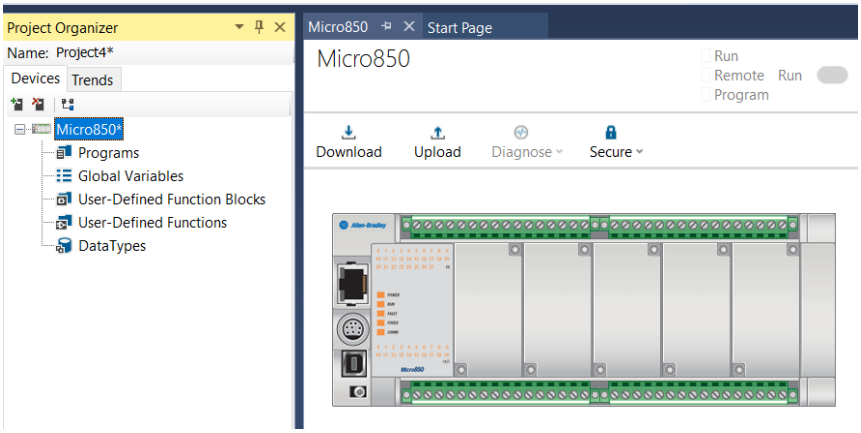
En este caso a través del puerto USB se ha conectado el equipo, se hace clic derecho al dispositivo. A continuación, se escoge **Module Configuration**, en la siguiente ventana que aparezca se puede visualizar algunas características relevantes del controlador, como por ejemplo el número de catálogo o la revisión de este.



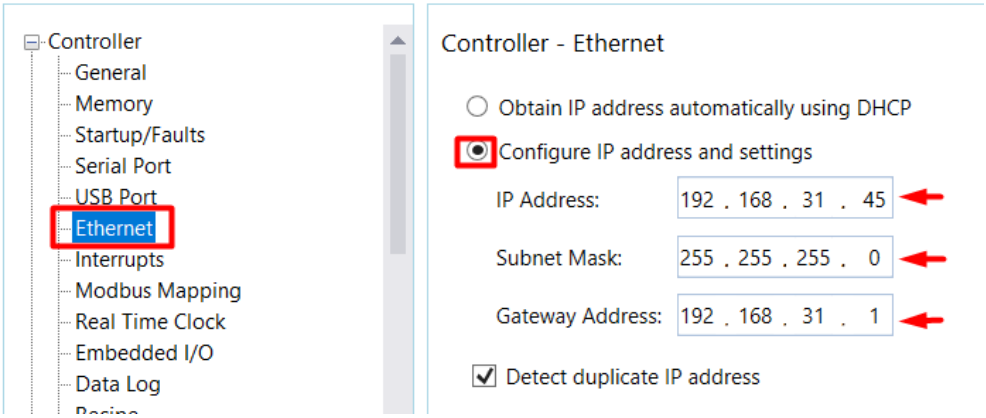
Finalmente, luego de verificar que tanto el número de catálogo como la revisión sean las correctas, dar clic **Add To Project.**



4. Si se creó correctamente el proyecto, el micro850 se muestra en la pestaña **Project Organizer** y además una imagen del controlador en la ventana.

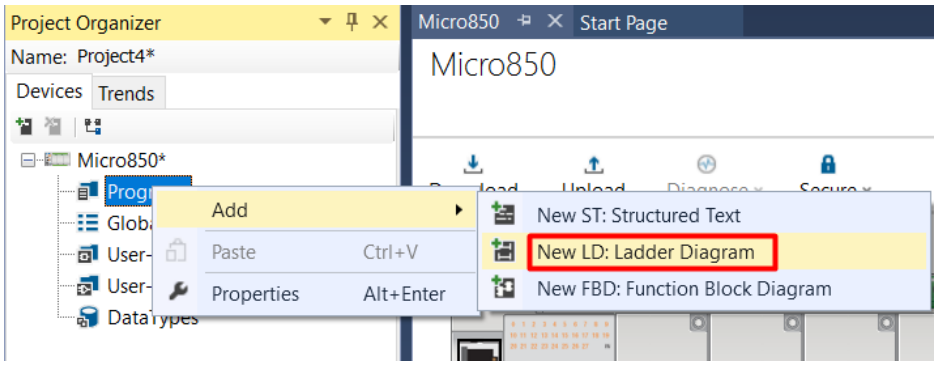


En caso de realizarse la práctica en el laboratorio se requiere configurar **Ethernet** del controlador, donde se deben escribir los siguientes parámetros: **IP Address, Subnet Mask y Gateway Address** del controlador correspondiente.

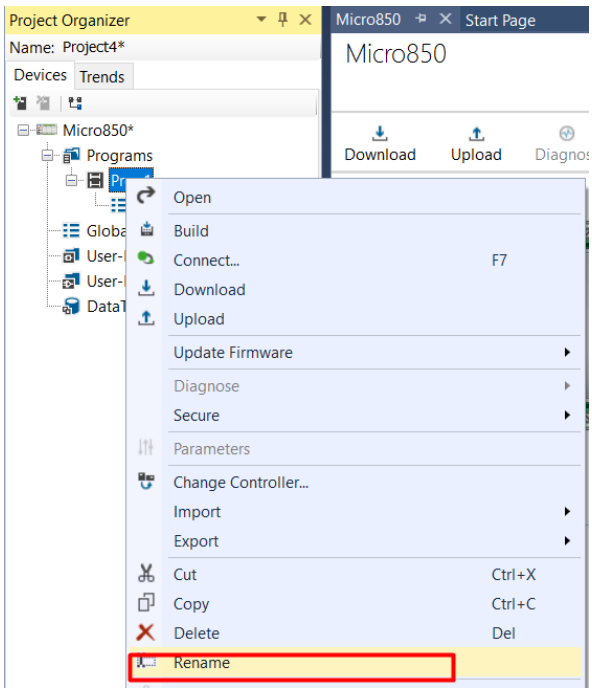


Añadir un programa en lenguaje escalera

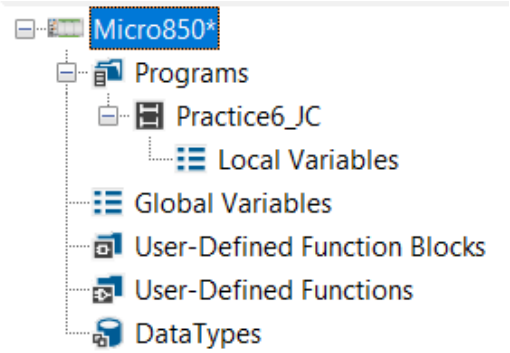
1. Agregar un nuevo programa, el cual puede ser programado en distintos lenguajes de programación, en este caso seleccionaremos **Ladder diagram**. Además, se debe considerar lo siguiente:
   - No se puede cambiar el lenguaje de programación cuando el programa ya fue creado.
   - El proyecto puede contener hasta 256 programas.
   - Cada programa debe tener un nombre diferente. Estos nombres pueden tener hasta 128 caracteres y deben comenzar por una letra.
   - Los controladores Micro800 permiten múltiples programas, así como el uso de diferentes lenguajes de programación (como Structured Text o Function Block Diagram) en la aplicación.



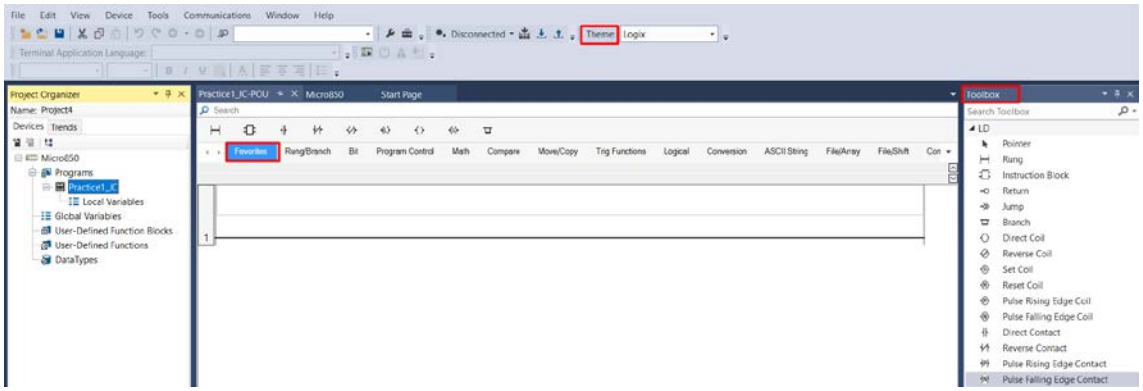
2. Clic derecho en el icono del programa llamado **Prog1** y seleccionar **Rename.**



3. Cambiar el nombre del programa de acuerdo con el **número de practica** y las iniciales del practicante tanto el **nombre** como **apellido**.

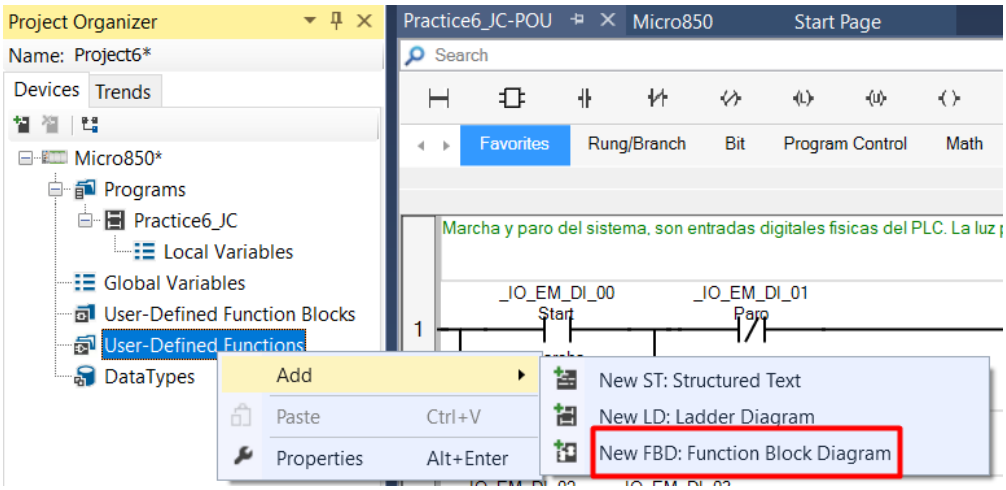


4. Doble clic en el icono del programa **Practice6_JC.** Luego, el editor del diagrama escalera aparece en el espacio de trabajo del proyecto principal con un peldaño vacio. Por último, la opción **Theme** escoger **Logix**, con este tema las instrucciones de los símbolos y terminología será más familiares cuando se utilice el software de programación Studio 5000 Logix Designer.

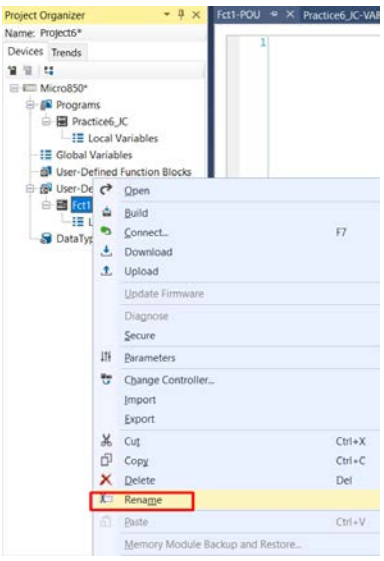


Añadir un user-defined function

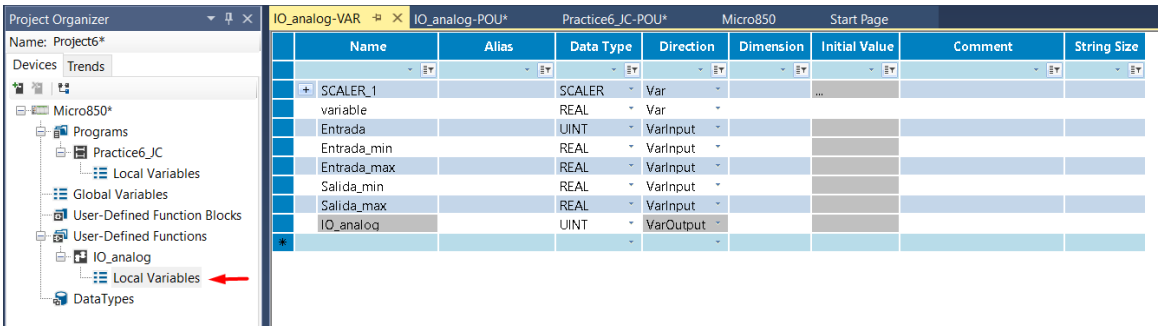
En **Project Organizer,** clic derecho en **User-Defined Functions (UDF).** Luego, seleccionar **Add** y entonces seleccionar el tipo de lenguaje de programación.



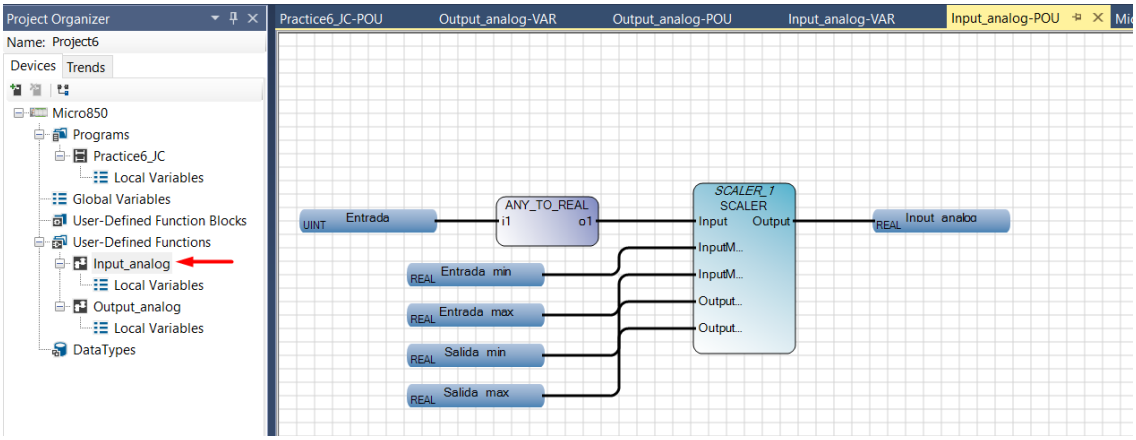
2. Clic derecho en el icono del programa llamado **Fct1** y seleccionar **Rename** para rescribir el nombre de la función acorde a la programación.



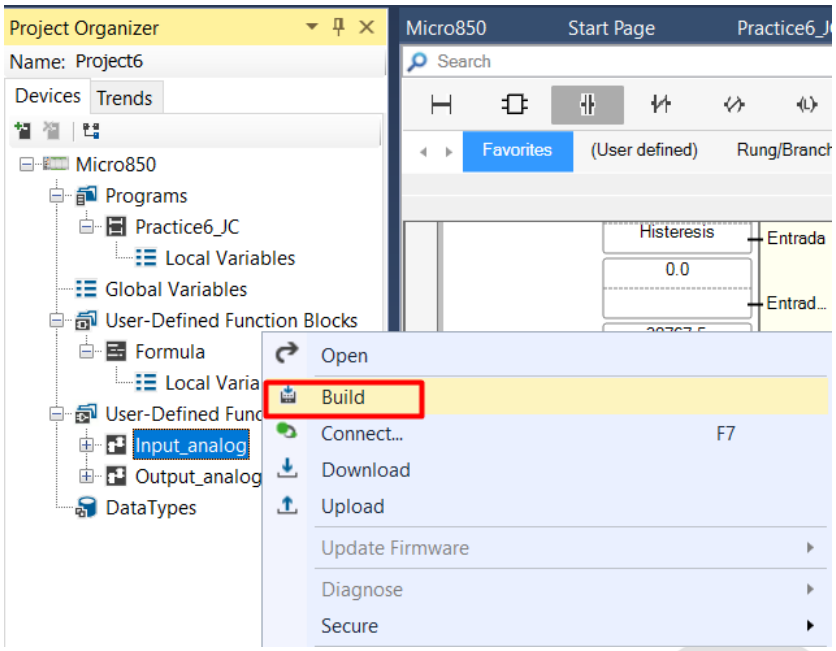
3. Doble clic en **Local Variables**. Luego, escribir los nombres y tipos de variables. Además, en cada variable se debe especificar la dirección; es decir si es una variable interna, se escoge **Var** o si son variables de entrada seleccionar **VarInput**. Cabe mencionar que UDF solo permite una variable de salida **VarOuput**.



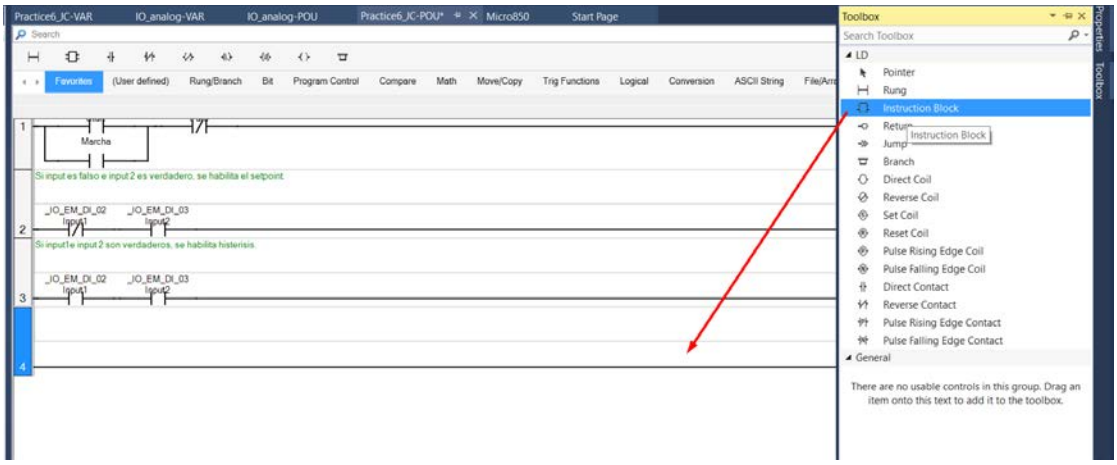
4. Doble clic a una función para escribir la programación respectiva, en este caso es **Input_analog.**



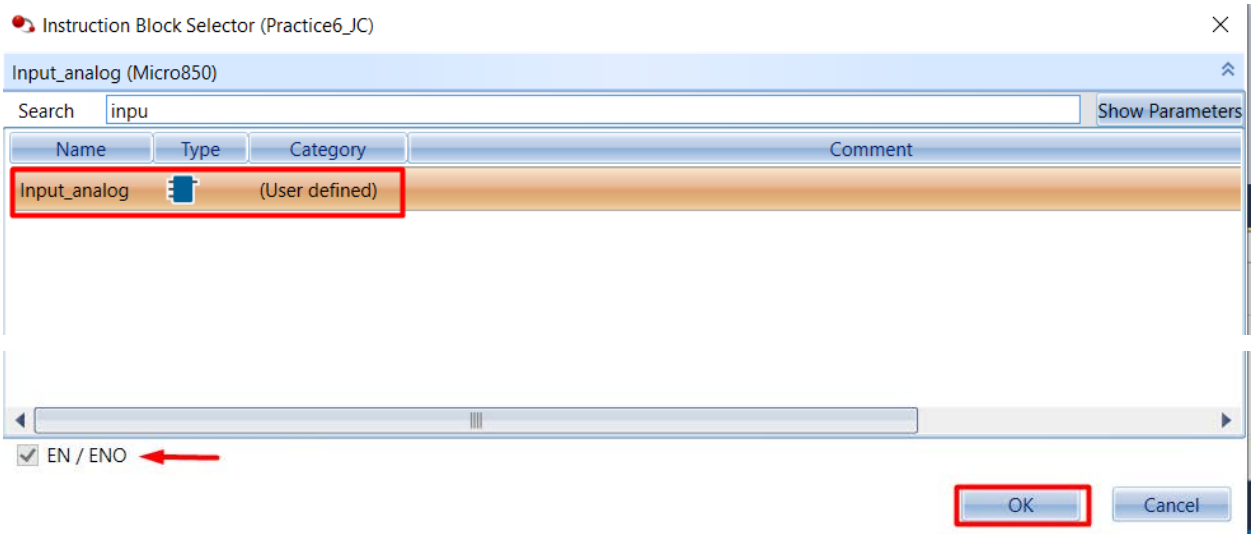
5. Clic derecho a **Input_analog** y seleccionar **Build** para compilar el código y si está libre de errores, continuar con los siguientes pasos.



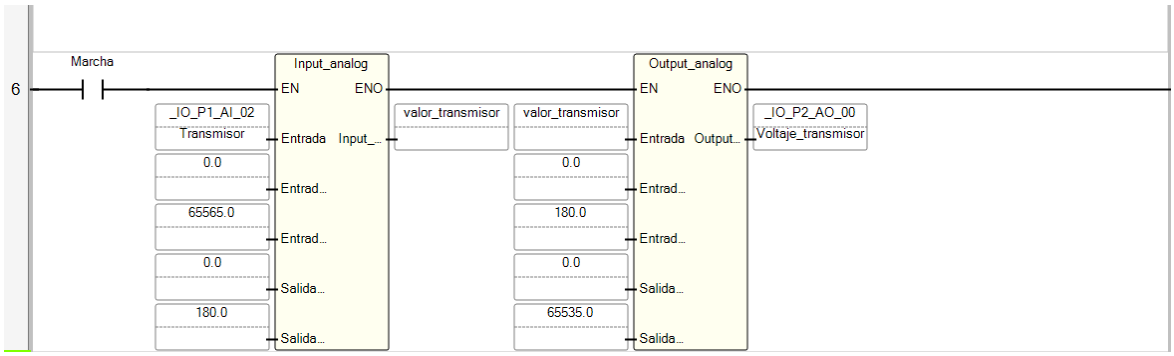
6. En el programa principal, escoger **Instrucción Block** desde **Toolbox** de CCW.



7. Escribir el nombre del UDF en la sección **search** de la ventana **Instruction Block Selector**. Luego, seleccionar el bloque y clic en **OK**.

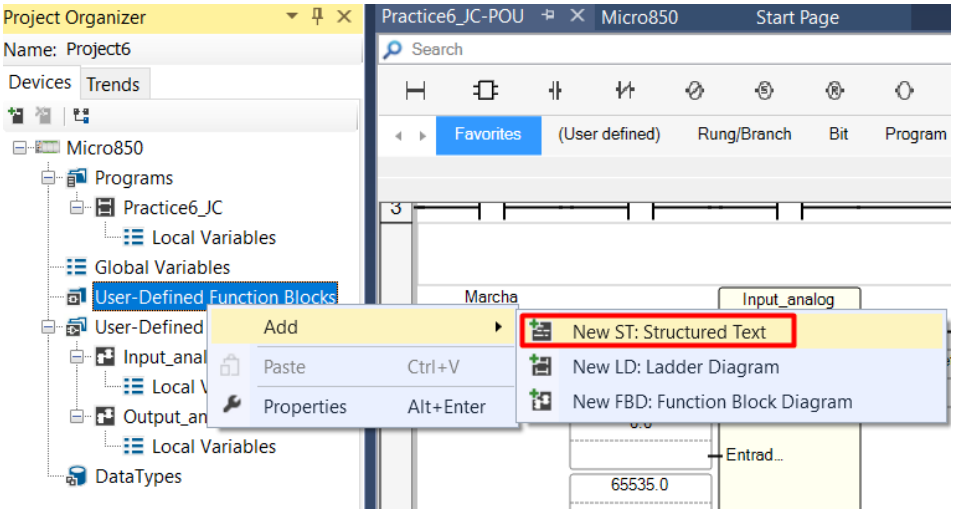


8. Finalmente, escribir los nombres de las entradas y salidas correspondientes con el mismo tipo de dato el cual fue creada el UDF.

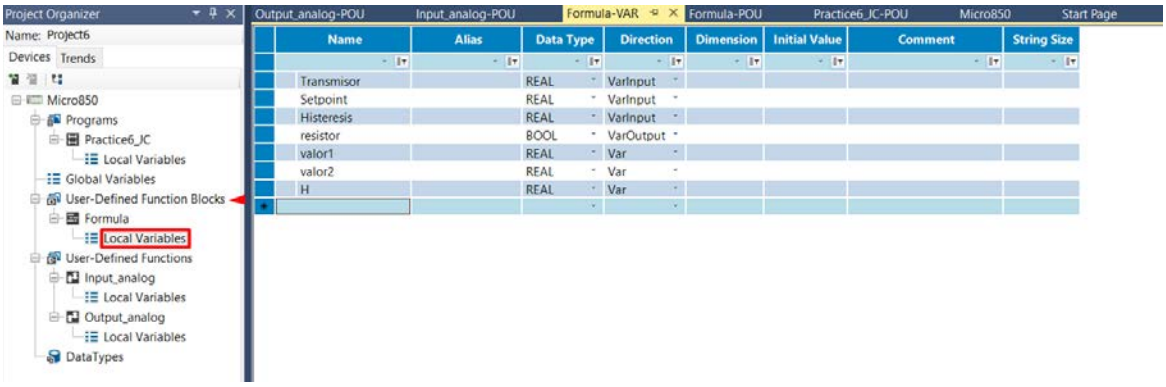


Añadir un user-defined function block

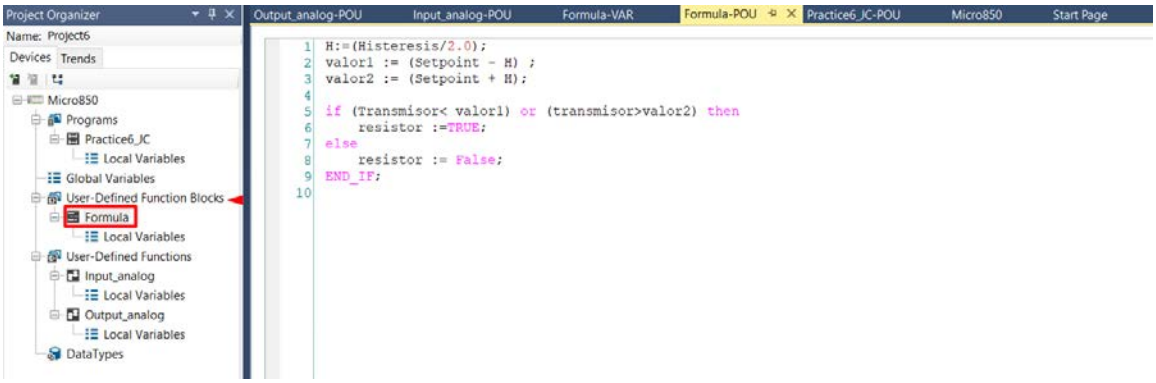
1. En **Project Organizer,** clic derecho en **User-Defined Function Blocks (UDFB).** Luego, seleccionar **Add** y entonces seleccionar el tipo de lenguaje de programación.



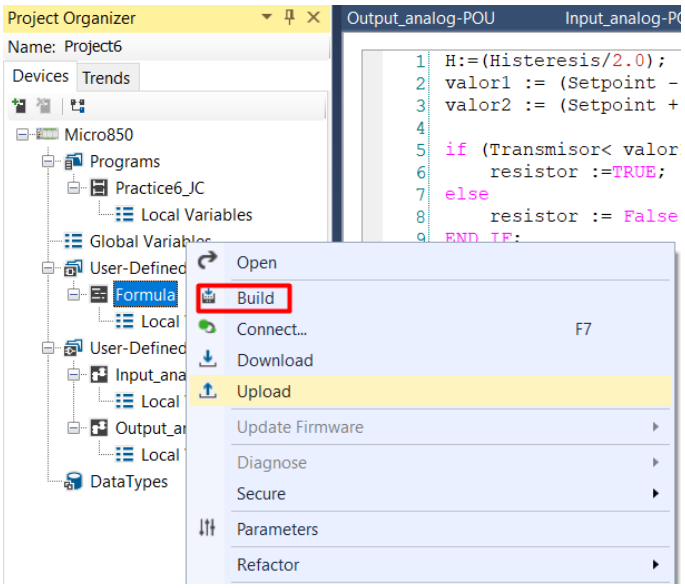
2. Clic derecho en el icono del programa llamado **Fct1** y seleccionar **Rename** para rescribir el nombre de la función acorde a la programación. Doble clic en **Local Variables**. Luego, escribir los nombres y tipos de variables. Además, en cada variable se debe especificar la dirección; es decir si es una variable interna, se escoge **Var** o si son variables de entrada seleccionar **VarInput**. Cabe mencionar que UDF permite más de una variable de salida **VarOuput**.



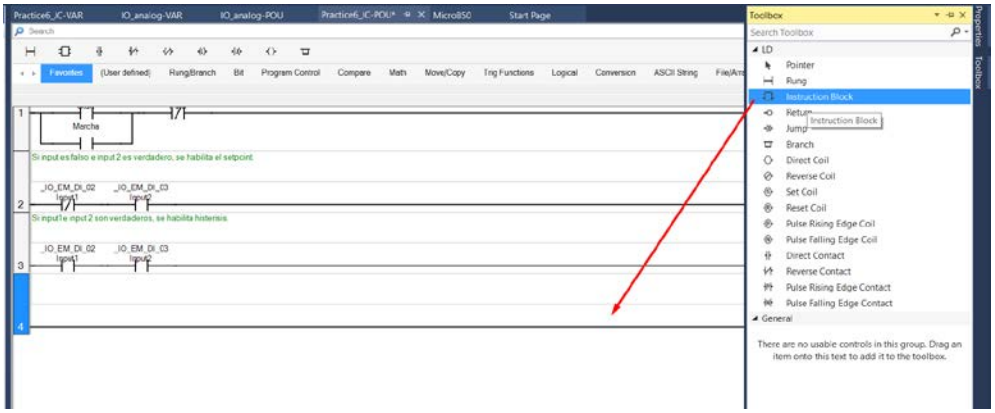
3. Doble clic en el bloque de función para escribir la programación respectiva, en este caso es **Formula.**



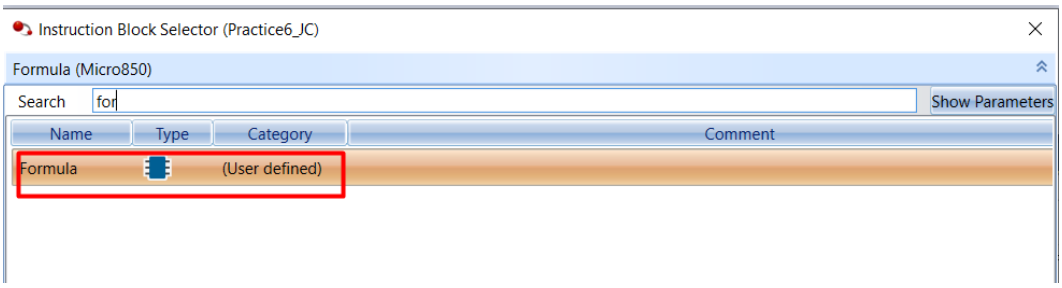
4. Clic derecho en **Formula** y seleccionar **Build** para compilar el código y si está libre de errores, continuar con los siguientes pasos.



5. En el programa principal, escoger **Instrucción Block** desde **Toolbox** de CCW.



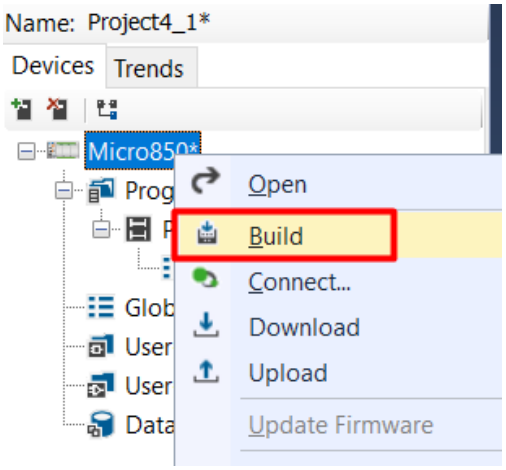
6. Escribir el nombre del UDF en la sección **search** de la ventana **Instruction Block Selector**. Luego, seleccionar el bloque y clic en **OK**.



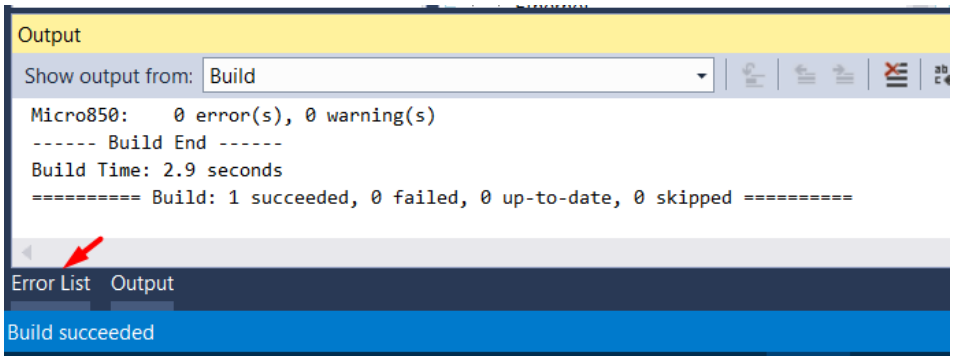
9. Finalmente, escribir los nombres de las entradas y salidas correspondientes con el mismo tipo de dato el cual fue creada el UDFB.

Compilar, descargar y comprobar un proyecto de CCW

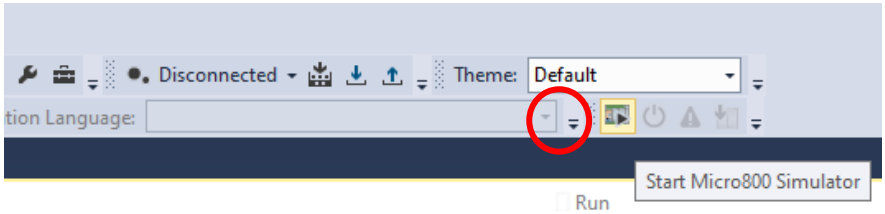
1. Compilar la aplicación dando clic derecho al controlador Micro850 en **Project Organizer**, y seleccionar **Build**.



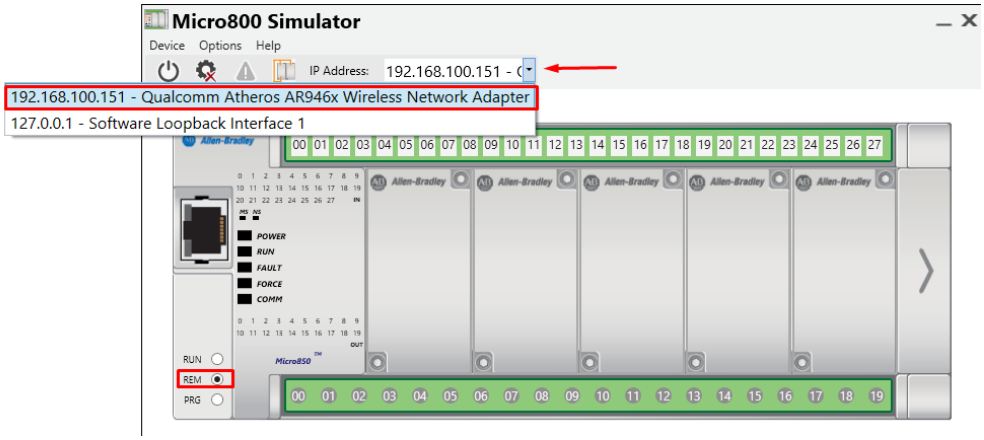
Cuando la compilación es completada, se observará un mensaje en la esquina inferior izquierda afirmando si la compilación ha sido exitosa. Si hay errores en la programación, entonces en el panel **Error List** dar clic en el error donde automáticamente se direccionará al error en el proyecto. Caso contrario, descargar el proyecto al controlador.



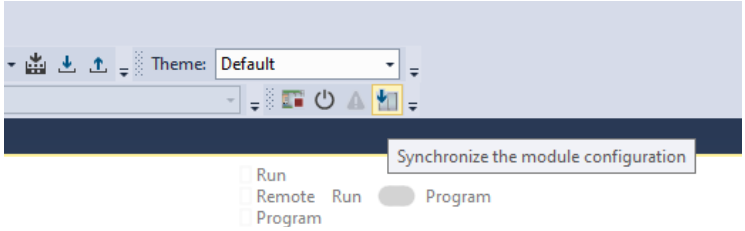
2. En la barra de herramientas de Connected Components, dar clic en “Micro800 Simulator” que permitirá la ejecución del simulador.



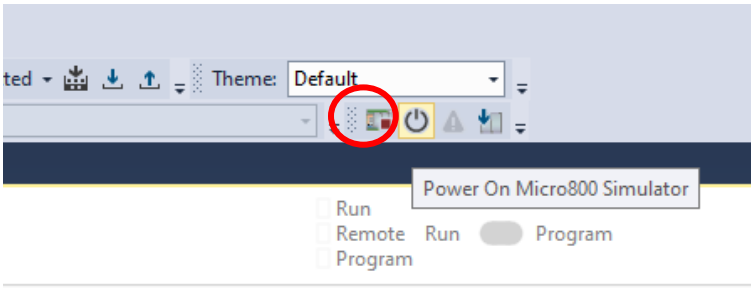
3. En la ventana del simulador seleccionamos la dirección IP del controlador para la comunicación ethernet; posteriormente se procede a sincronizar la configuración del controlador con el simulador dando clic en el botón mostrado.

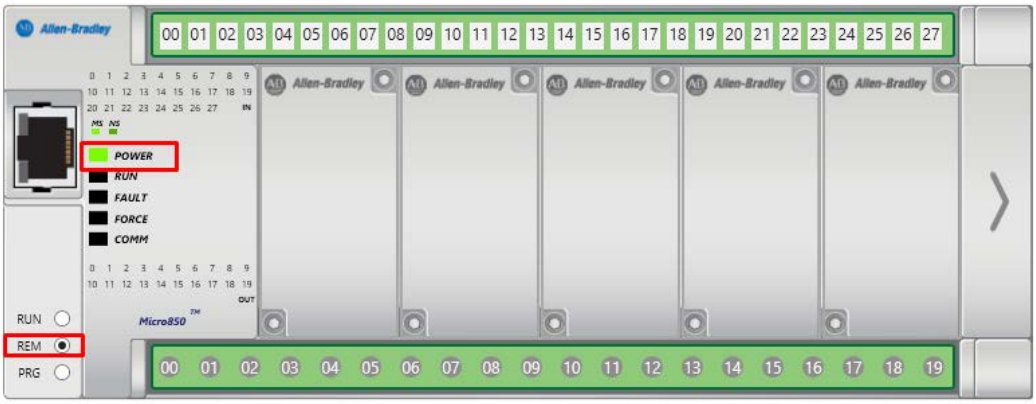


4. Luego, se procede a sincronizar la configuración del módulo con el simulador Micro 800 dando clic en “Sincronizar la configuración del módulo”.

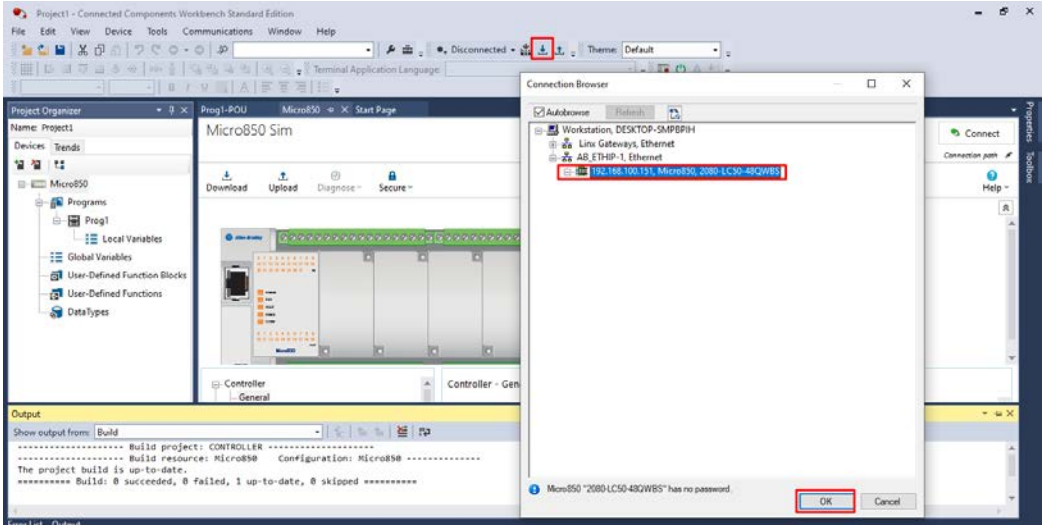


5. A continuación, se debe encender el controlador con la ventana del simulador abierta.

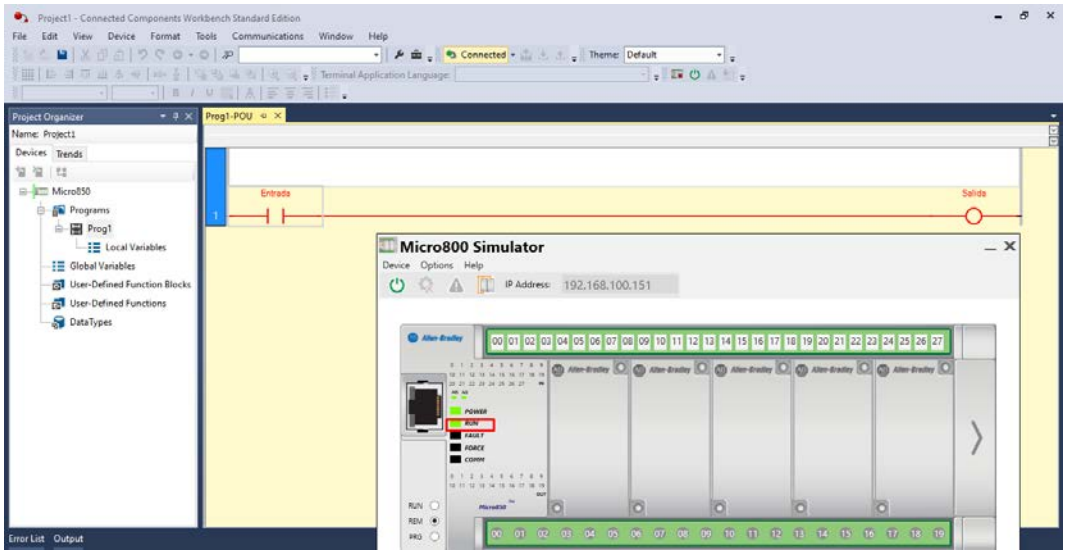




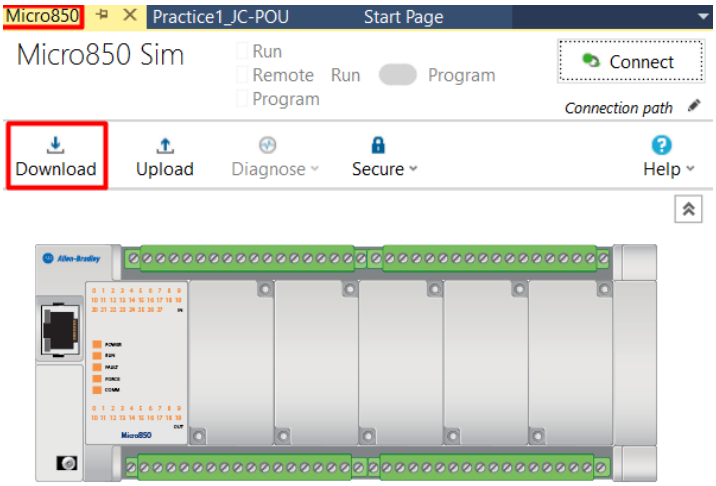
6. Clic en “Download” en la barra de herramienta del software para descargar el programa del computador hacia el simulador Micro800. Luego, seleccionamos el PLC con la dirección IP asignada con anterioridad. Por último, dar clic en “OK”.



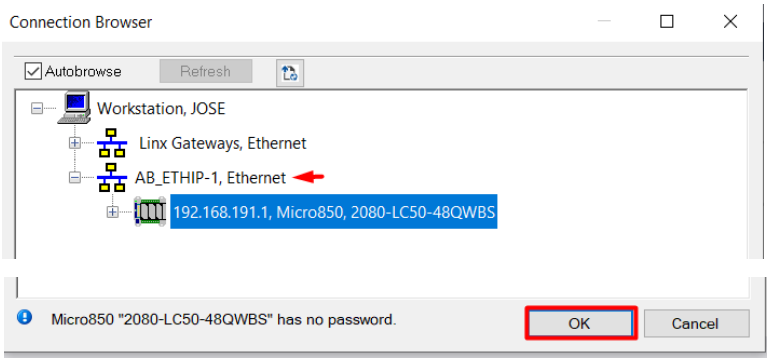
7. El controlador se encuentra “Conectado”, si deseáramos desconectarlo, en este mismo ícono podemos realizarlo. El color azul muestra que la línea no está energizada y el color rojo muestra que la línea está energizada.



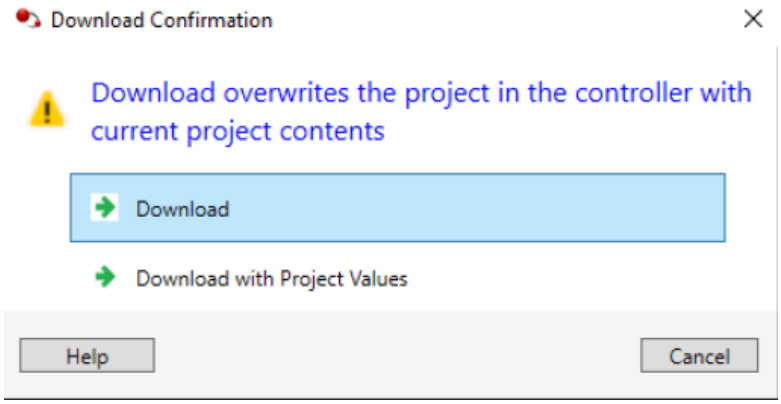
Si la descarga del programa se hiciera en un controlador real, se hace clic en **Download** situado debajo de la ventana **Micro850**.



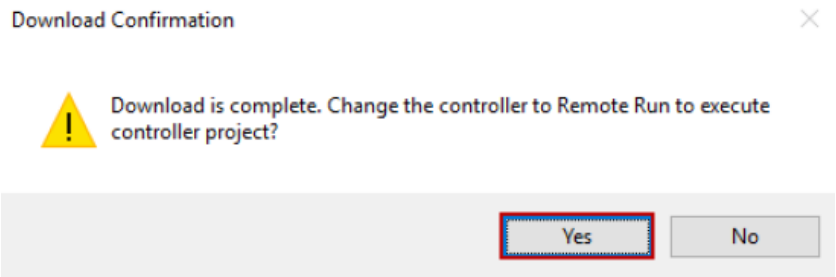
La ventana **Connection Browser** emergerá, luego buscar el controlador correspondiente expandiendo **AB_ETHIP-1** y escoger el controlador con la respectiva **dirección IP**. Por último, dar clic en **OK.**



La ventana **Download Confirmation** aparecerá para sobrescribir el proyecto en el controlador. En este caso, el proyecto no tiene valores asignado todavía por lo tanto dar clic en **Download**.



La descarga continuará, sin embargo, se presenta el anuncio para colocar el controlador en modo Run. Clic **Yes**.



El controlador está en modo **Connected**. Si se quiere desconectar, desplegar las opciones del mismo icono y seleccionar Disconnected. Además, el color gris como fondo en las líneas de código muestra que se encuentra en línea.