ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

COMUNICACIONES INDUSTRIALES Y SISTEMAS SCADA

Práctica

Comunicación Modbus TCP/IP con PLCs Logix5000

Autores

Arturo Freire Veliz

Paul Steven Martillo Pozo

Profesor

Ing. José Cueva Tumbaco

GUAYAQUIL-ECUADOR

I PAO 2025

Table of Contents

1. Objetivos

1.1. Objetivo General

Implementar una arquitectura de comunicación industrial basada en el protocolo Modbus TCP/IP utilizando el Add-On Instruction (AOI) oficial de Rockwell Automation, con el propósito de establecer un intercambio de datos fiable entre controladores Allen-Bradley de la familia Logix5000, la plataforma Node-RED y un analizador de red Sentron PAC3200.

1.2. Objetivos Específicos

- Configurar e implementar el AOI de Modbus TCP/IP en los controladores CompactLogix y ControlLogix para establecer comunicación entre ellos, actuando el primero como cliente y el segundo como servidor.
- Integrar a Node-RED como segundo cliente en la arquitectura de comunicación, validando su capacidad para leer datos directamente desde el ControlLogix mediante Modbus TCP/IP.
- Establecer la comunicación entre el CompactLogix y el analizador Sentron PAC3200 de Siemens, permitiendo al controlador leer datos de medición del dispositivo.
- Diseñar un flujo lógico en el CompactLogix que permita almacenar y reenviar los datos adquiridos del SentronPac hacia el ControlLogix, y garantizar que Node-RED reciba dicha información desde este último.
- Verificar la integridad, consistencia y sincronización del intercambio de datos entre todos los dispositivos involucrados en el sistema de comunicación.

2. Equipos y herramientas

- Studio 5000 Logix Designer.
- RSLinx Classic.
- Cables Ethernet.
- Stratix 5700 / 8300.
- PCs.
- CompactLogix 1769-L33ERM.
- ControlLogix L73, Módulo EN2TR.
- Node-RED.
- Sentron PAC3200.
- Manuales de Programación: Modbus TCP Client Add-On Instruction based code for ControlLogix and CompactLogix controller, Multímetro SENTRON PAC3200 Manual de usuario.

3. Marco Teórico

3.1. Modelo Cliente-Servidor

Es una arquitectura de comunicación, en donde se establece la relación que existe entre los procesos de dispositivos distintos. El cliente es el dispositivo *que envía una solicitud a un servidor*. El cliente espera una respuesta del servidor. Por otra parte, el servidor es el dispositivo *que recibe una solicitud de un cliente*. Se espera que el servidor dé una respuesta al cliente.

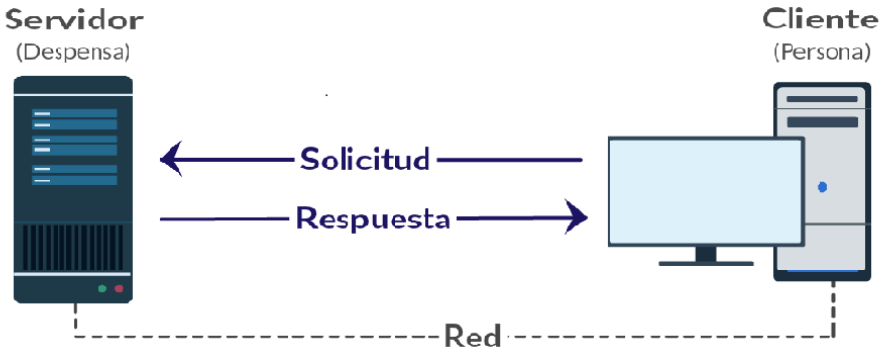


Figura 1. Modelo Cliente-Servidor.

3.2. Protocolo Modbus TCP/IP

Modbus es un protocolo de comunicación industrial desarrollado originalmente por Modicon (hoy Schneider Electric) en 1979. Su versión Modbus TCP/IP se ejecuta sobre redes Ethernet estándar, permitiendo la comunicación entre dispositivos industriales como PLCs, variadores y HMIs mediante el modelo cliente-servidor. En este protocolo, el cliente solicita datos o acciones, y el servidor responde.

Modbus TCP/IP utiliza el *puerto 502 por defecto* y transmite datos en formato de registros: entradas/salidas digitales (coils), entradas analógicas y registros de retención (Holding Registers). Es ampliamente utilizado por su simplicidad, apertura y compatibilidad entre dispositivos de distintos fabricantes.

Figura 2. Protocolo Modbus TCP/IP.

3.2.1. Objetos, Códigos de Función y Direccionamiento

Comandos a nivel de Bit

Código de Función	Nombre	Descripción	Valores Soportados	Rango Modbus
01	Leer Bobinas	Este código de función se utiliza para leer el estado contiguo de bobinas en un dispositivo remoto (direcciones 0xxxx). Las bobinas en el mensaje de respuesta se empaquetan como una bobina por bit del campo de datos.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 bobinas	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536
02	Leer Entradas Discretas	Este código de función se utiliza para leer el estado contiguo de entradas discretas en un dispositivo remoto (direcciones 1xxxx). Las entradas en el mensaje de respuesta se empaquetan como una bobina por bit del campo de datos.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 Entradas	**Dirección Local:** 10001-11024 **Dirección del Servidor:** 10001-165536
05	Escribir Bobina Simple	Este código de función se utiliza para escribir una sola bobina a ON o OFF en un dispositivo remoto (direcciones 0xxxx).	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536
15	Escribir Múltiples Bobinas	Este código de función se utiliza para escribir en una o más bobinas en una secuencia de bobinas a ON o OFF en un dispositivo remoto (direcciones 0xxxx).	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 256 bobinas	**Dirección Local:** 00001-01024 **Dirección del Servidor:** 000001-065536

Comandos a nivel de Palabra

Código de Función	Nombre	Descripción	Valores Soportados	Rango Modbus
03	Leer Registros de Retención	Este código de función se utiliza para leer el contenido de un bloque contiguo de registros de retención (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536
04	Leer Registros de Entrada	Este código de función se utiliza para leer el contenido de un bloque contiguo de registros de entrada (direcciones 3xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros de entrada	**Dirección Local:** 30001-31024 **Dirección del Servidor:** 300001-365536
06	Escribir un Solo Registro de Retención	Este código de función se utiliza para escribir en un solo registro de retención (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536
16	Escribir Múltiples Registros de Retención	Este código de función se utiliza para escribir en registros de retención contiguos (direcciones 4xxxx) en un dispositivo remoto.	**Dirección Local:** 0 a 1023 **Dirección del Servidor:** 0 a 65535 **Longitud:** 1 a 120 registros	**Dirección Local:** 40001-41024 **Dirección del Servidor:** 400001-465536

Nota: Un error común es considerar que las direcciones o canales de Modbus equivalen a los espacios de memoria en el controlador. Por ejemplo, a nivel de bits, esta suposición se cumple, ya que cada variable booleana ocupa un bit de memoria (Data[0].0, Data[0].1), y por tanto a cada una se le asignaría una dirección Modbus correspondiente (00001, 00002). Pero, en el caso de variables analógicas (16 bits, 32 bits, etcétera), no se cumple aquello. La dirección Modbus 40001 equivale a 1 HR, por lo que, si tenemos dos variables analógicas de 16 bits, se les asignaría las direcciones 40001 y 40002 respectivamente sin ningún problema, **y no 40001 y 40017** si se pensaran como espacios de memoria. Si se tuviera una variable de tipo Double-Word (32 bits), esta ocuparía 2 HR, por lo que, si se le asigna la dirección 40003, esta también ocuparía la dirección 40004, por lo que la siguiente dirección disponible sería la 40005. Esto también se cumple para los Registros de Entrada.

3.2.2. Trama de Datos

El comando Modbus TCP consta de parte del mensaje Modbus RTU y un encabezado especial. La dirección SlaveID al principio y la suma de comprobación CRC al final se eliminan del mensaje Modbus RTU, creando la PDU (Unidad de datos de protocolo).

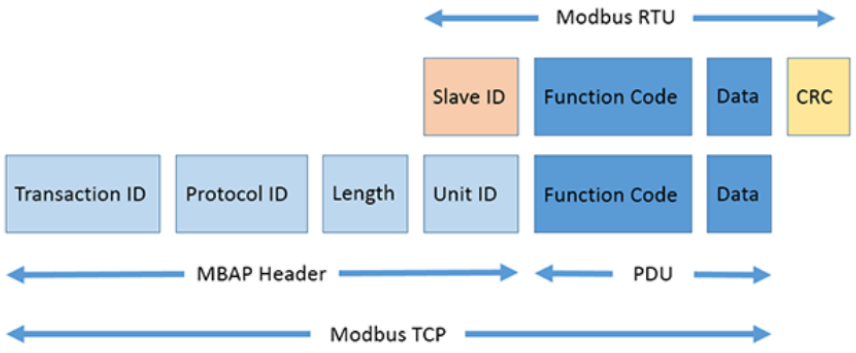


Figura 3. Trama del protocolo Modbus TCP/IP a partir de Modbus RTU.

Se añade un nuevo encabezado de 7 bytes, denominado encabezado MBAP (encabezado de aplicación Modbus), al inicio del mensaje PDU recibido. Este encabezado contiene los siguientes datos:

Identificador de Transacción: 2 bytes definidos por el maestro para identificar de forma única cada solicitud. Puede ser cualquier número. El dispositivo esclavo repite estos bytes en la respuesta, ya que sus respuestas no siempre se reciben en el mismo orden que las solicitudes.

Identificador de Protocolo: 2 bytes definidos por el maestro, siempre 00 00, que corresponde al protocolo Modbus.

Longitud: 2 bytes especificados por el maestro, que indican la cantidad de bytes del mensaje posterior. Se cuenta desde el identificador de unidad hasta el final del mensaje.

Identificador de Unidad: 1 byte especificado por el maestro. El dispositivo esclavo lo repite para identificarlo de forma única.

0001	**Identificador de transacción**	Identificador de transacción
0000	**Identificador de protocolo**	Identificador de protocolo
0006	**Longitud** (siguen 6 bytes)	Longitud del mensaje
11	**Dirección del dispositivo** (17 = 11 hexadecimales)	Identificador de unidad
03	**Código de función** (lectura de los registros de retención de salida analógica)	Código de función
006B	**Dirección del primer registro** (107 = 40108-40001 = 6B hex)	Dirección de datos del primer registro
0003	**Número de registros necesarios** (lectura de 3 registros del 40108 al 40110)	El número total de registros

Figura 4. Solicitud de un Cliente Modbus TCP/IP.

0001	**Identificador de transacción**	Identificador de transacción
0000	**Identificador de protocolo**	Identificador de protocolo
0009	**Longitud** (siguen 9 bytes)	Longitud del mensaje
11	**Dirección del dispositivo** (17 = 11 hexadecimales)	Identificador de unidad
03	**Código de función** (lectura de los registros de retención de salida analógica)	Código de función
06	**Número de bytes siguientes** (siguen 6 bytes)	Recuento de bytes
02	**Valor de registro Hi (AO0)** (02 hex)	Valor de registro Hi (AO0)
2B	**Valor de registro Lo (AO0)** (2B hexadecimal)	Valor de registro Lo (AO0)
00	**Valor de registro Hi (AO1)** (00 hexadecimal)	Valor de registro Hi (AO1)
64	**Valor de registro Lo (AO1)** (64 hex)	Valor de registro Lo (AO1)
00	**Valor de registro Hi (AO2)** (00 hexadecimal)	Valor de registro Hi (AO2)
7F	**Valor de registro Lo (AO2)** (7F hexadecimal)	Valor de registro Lo (AO2)

Figura 5. Respuesta de un Servidor Modbus TCP/IP.

3.3. Relación y Comparación entre EtherNet/IP y Modbus TCP/IP

Aunque Ethernet/IP y Modbus TCP/IP pueden coexistir sobre la misma red física (Ethernet), son protocolos diferentes. Sin embargo, los controladores de Allen-Bradley pueden ser configurados para comunicarse también usando Modbus TCP/IP mediante funciones adicionales, como *Add-On Instructions (AOIs)* o archivos de configuración específicos.

3.4. Endianismo (Endianness)

El endianismo *es el orden en el que los bytes dentro de una palabra de datos digitales se transmiten a través de un medio de comunicación de datos* o se direccionan (por direcciones ascendentes) en la memoria de la computadora, contando solo la *significancia del byte en comparación con la anticipación*. El endianismo se expresa principalmente como Big-Endian (BE) o Little-Endian (LE).

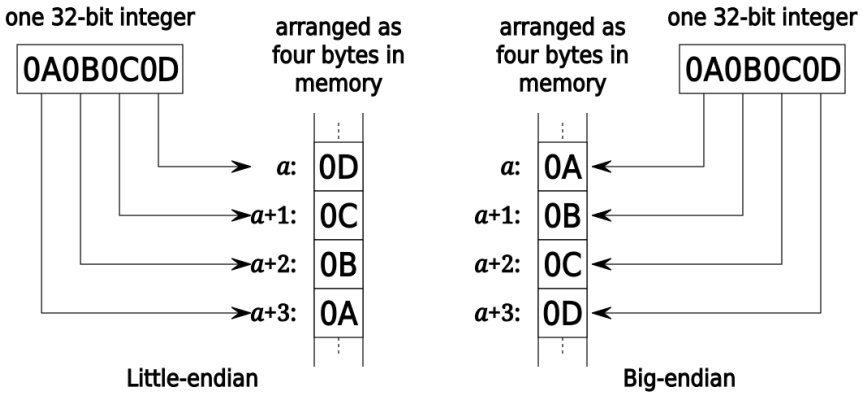


Figura 6. Concepto de Endianismo.

Por ejemplo, en la Figura 1, observamos un caso concreto donde acontece el endianismo. Se tiene un dato de tipo DINT (32 bits), que en hexadecimal tiene un valor de 0x0A0B0C0D, donde 0A es el byte más significativo, y 0D el menos significativo. Podemos representar a este dato con el formato de ABCD para mayor simplicidad. Ahora, si nuestro ordenador funciona con orden Little-Endian, el byte menos significativo se almacenará en el espacio de memoria [0], mientras que el más significativo en el [3] (se leería ABCD). Pero, si por el contrario funcionara con orden Big-Endian, el menos significativo se almacenaría en el espacio [3], y el más significativo en el [0] (se leería DCBA).

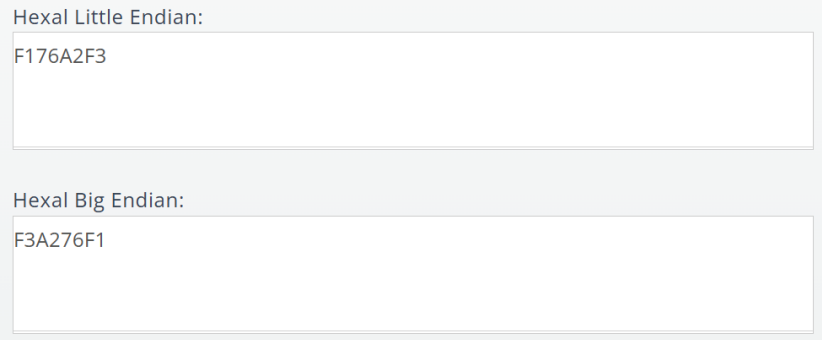


Figura 7. Ejemplo Hexadecimal de Endianismo.

Por ejemplo, en la Figura 2, tenemos un dato de 4 bytes, el cual se lee en orden Little-Endian como 0xF176A2F3. Por simplicidad, F1 será A, 76 será B, A2 será C y F3 será D, formando así ABCD. Si ahora cambiamos el orden a Big-Endian, obtendríamos DCBA, lo cual se leería como 0XF3A276F1.

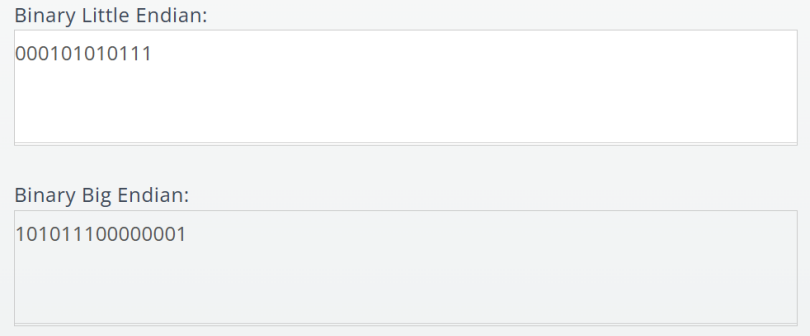


Figura 8. Ejemplo Binario de Endianismo.

Por ejemplo, en la Figura 3, tenemos un dato de 12 bits, el cual se lee en orden Little-Endian como 000101010111. Debido a que el ordenamiento se realiza a nivel de bytes, se completa el número a 16 bits (2 bytes) de la siguiente forma: 0000000101010111 (se puede leer como AB). Si ahora cambiamos el orden a Big-Endian, obtendríamos 1010111000000001 (se puede leer como BA).

3.4.1. Endianismo en Modbus

Modbus es un *protocolo Big-Endian*, es decir, el byte más significativo de un valor de 16 bits se envía antes que el byte menos significativo. Parece obvio que los valores de 32 y 64 bits también deben transferirse en orden Big-Endian. Sin embargo, algunos fabricantes *han optado por tratar los valores de 32 y 64 bits como si estuvieran compuestos por palabras de 16 bits y transferirlas en orden Little-Endian*. Por ejemplo, el valor de 32 bits 0x12345678, donde 0x1234 sería la Word "A" y 0x5678 sería la Word "B" (formando AB), se leería en orden Little-Endian como 0x56781234, o BA.

Nota: Esto será de mucha importancia cuando se realice la lectura del Sentron PAC3200.

4. Arquitectura de Comunicación

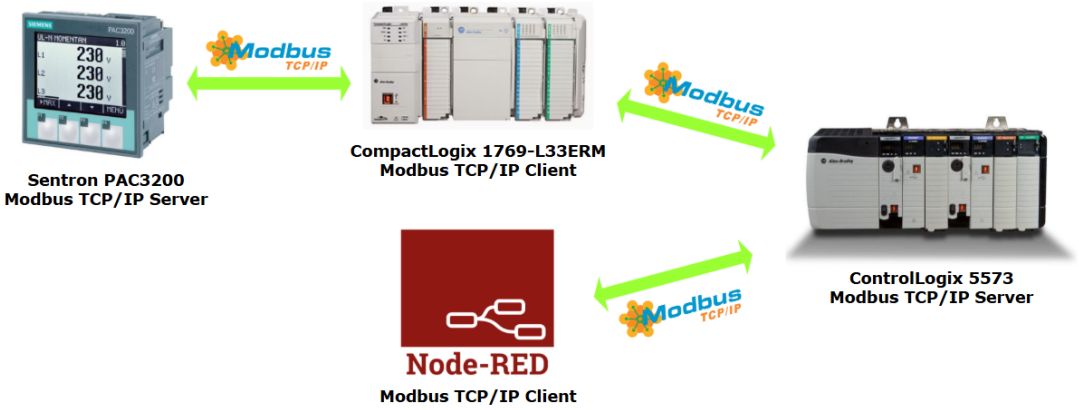


Figura 9. Arquitectura de Comunicación Modbus TCP/IP.

5. Procedimiento

En este apartado se explica a detalle la configuración para la comunicación Modbus TCP/IP entre los distintos dispositivos y servicios que forman parte de la arquitectura. Inicialmente, se explicarán las comunicaciones puntuales (Compact y Control, Control y Node-RED, Compact y Sentron), para luego implementar la arquitectura completa.

5.1. Comunicación Modbus TCP/IP entre CompactLogix y ControlLogix

5.1.1. Configuración Modbus Client en el Compactlogix

- Inicialmente, se debe crear un proyecto en Studio 5000 Logix Designer para configurar el cliente de la comunicación Modbus, el cual es un CompactLogix con CPU modelo 1769-L33ERM. Considerar la revisión de Firmware.

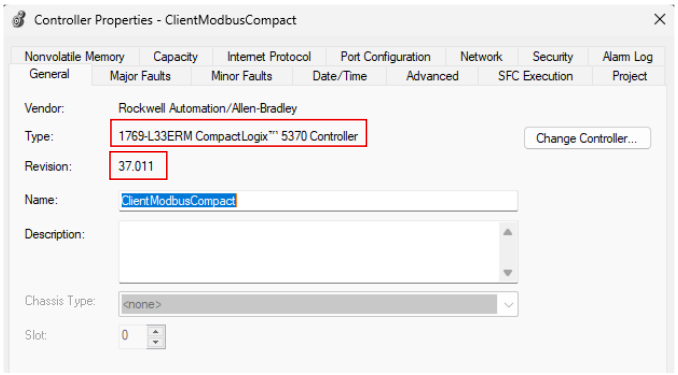


Figura 10. Configuración del Controlador.

Nota: Antes de importar el Modbus TCP Client AOI, se recomienda crear una *tarea periódica (Periodic Task)* con un intervalo de 10 ms o superior, para garantizar una ejecución estable y cíclica del AOI.

- Dirigirse a MainRoutine, dar clic derecho en el rung y seleccionar la opción *Import Rung*. Seleccionar el archivo *raC_Opr_NetModbusTCPClient_Rung.L5*.

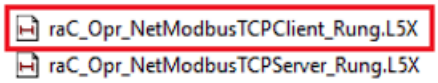


Figura 11. Rung a importar.

- En el Dialogo de importación, seleccionar *Tags*.

Figura 12. Seleccionar la opción de Tags.

Opcional: Usar la opción *Find/Replace* para cambiar nombres predeterminados como "*Client_01"* a un prefijo personalizado para la aplicación (esto en el caso que se quiera identificar de mejor el o los clientes).

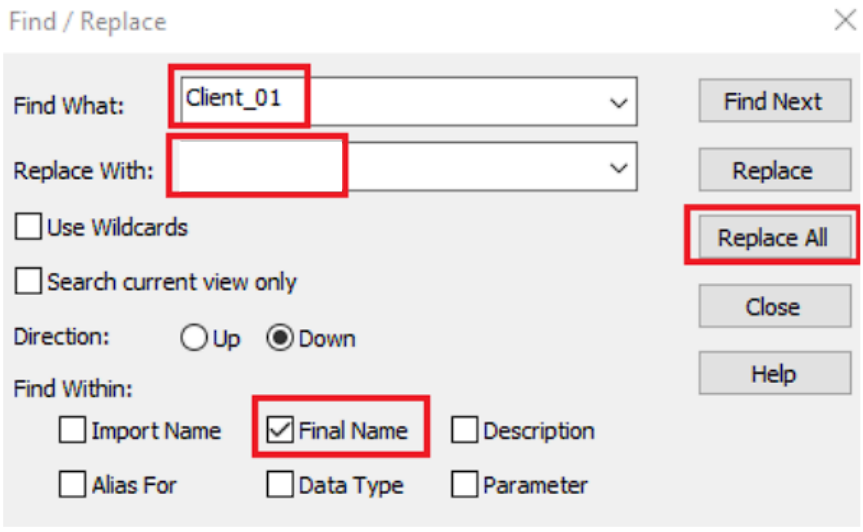


Figura 13. Uso de la opción Find/Replace.

- Hacer clic en Ok para completar la importación.

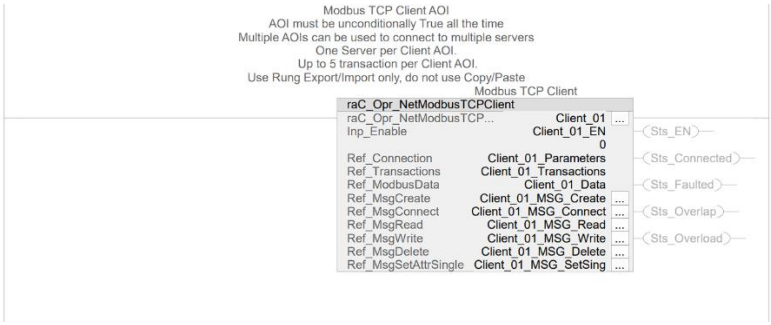


Figura 14. Rung correctamente importado.

- El nuevo rung debe verse como la figura mostrada sin ningún error.

Configurar Parámetros de red local

- Localizar el parámetro *Ref_Connection* del AOI importado, dar clic derecho y seleccionar la opción *Monitor "..."*.

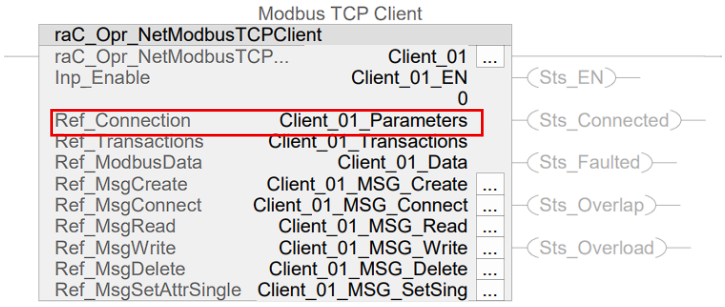


Figura 15. Monitorear el parámetro Ref_Connection.

- Expandir la variable *Client_01_Parameters*.

Especificar el slot del módulo local EtherNet/IP

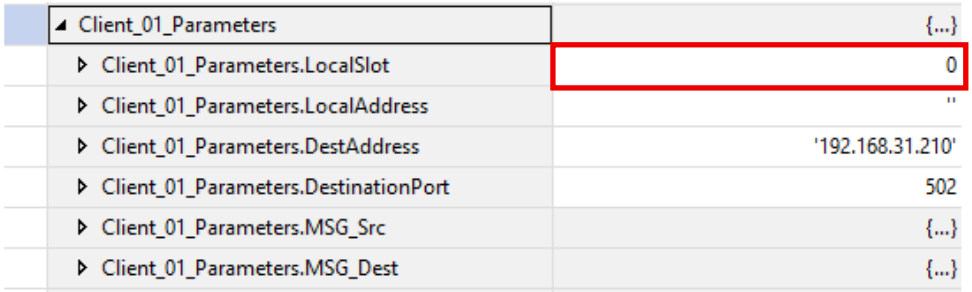


Figura 16. Especificar el slot del módulo local EtherNet/IP.

- Para los procesadores 1756 ControlLogix, especifica el slot real del módulo 1756-EN2T(R) deseado. Para los controladores CompactLogix 5370, 5380 y 5480, dejar el parámetro *LocalSlot* en 0.

Especificar la dirección local del módulo EtherNet/IP

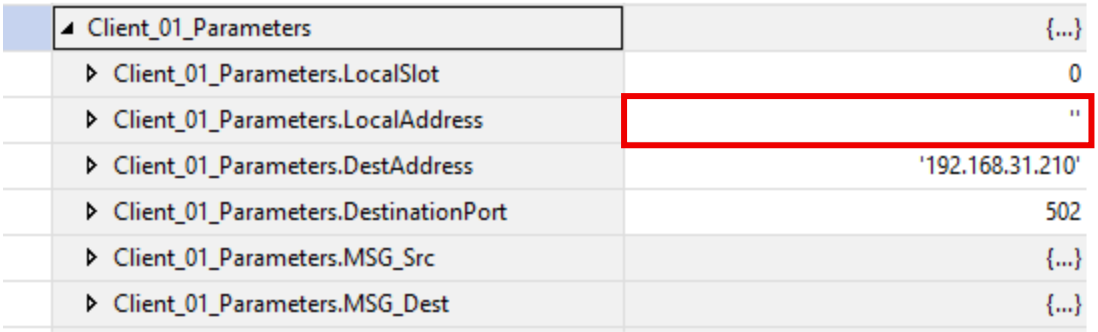


Figura 17. Especificar la dirección del módulo local EtherNet/IP.

- Solo para los CompactLogix 5380 y 5480 en modo Dual IP, especifica la dirección IP de la conexión Ethernet local utilizada para las comunicaciones Modbus TCP. Dejar este campo en blanco en todos los demás casos.

Especificar la dirección IP del dispositivo servidor Modbus

- Se debe asignar la dirección IP del dispositivo servidor al cual se desea comunicar. Esta dirección debe ser indicada y *no puede quedar en blanco*.

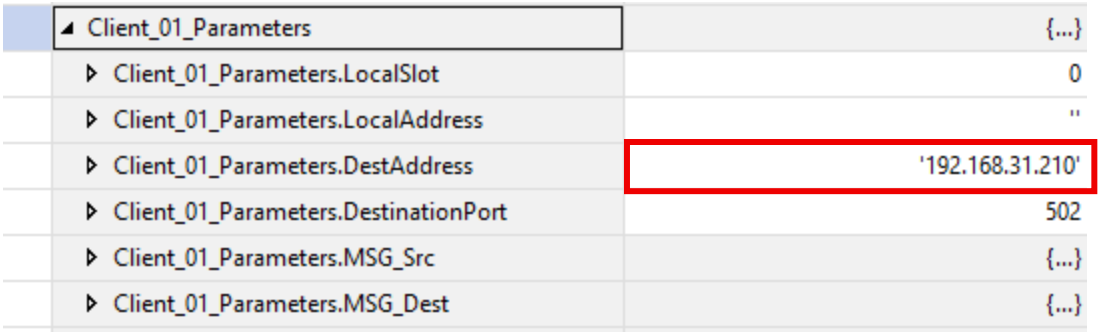


Figura 18. Especificar la dirección del dispositivo servidor Modbus.

- Deja el puerto Modbus TCP predeterminado en 502. Este valor es el *estándar del protocolo Modbus TCP/IP*.

- Iniciar el cliente Modbus TCP configurando en 1 la etiqueta asociada al parámetro *Inp_Enable*.

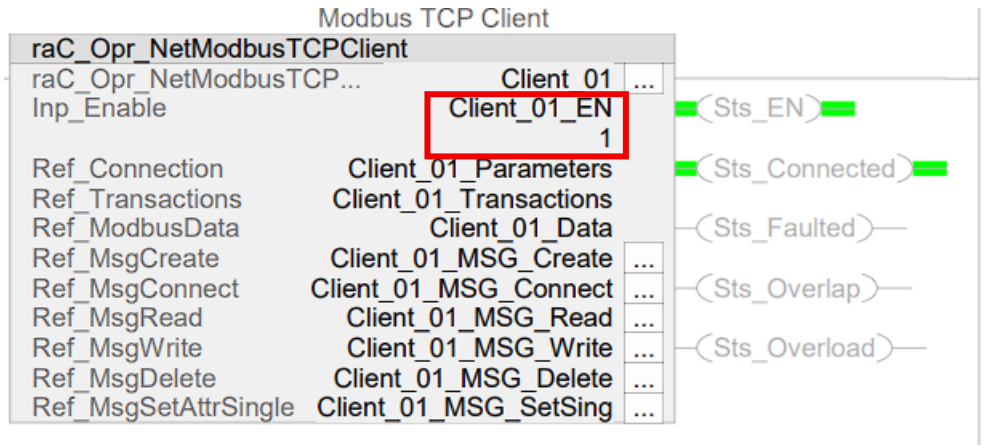


Figura 19. Iniciar el Cliente Modbus.

Configurar las Transacciones de Datos

- Expandir la estructura de etiquetas *Transaction[]* en el AOI para visualizar los campos de configuración del miembro *Transaction[0]* (se lo puede encontrar en los tags del controlador).

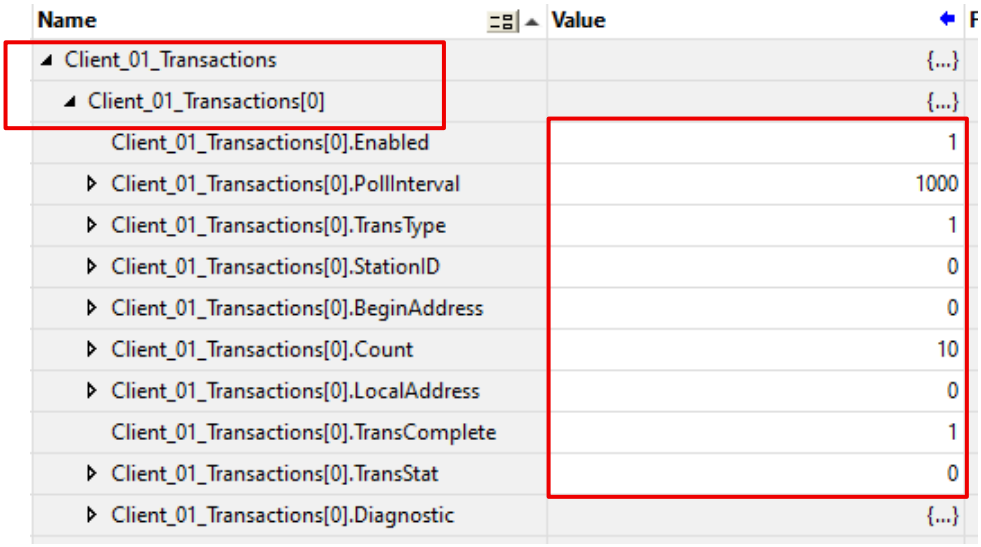
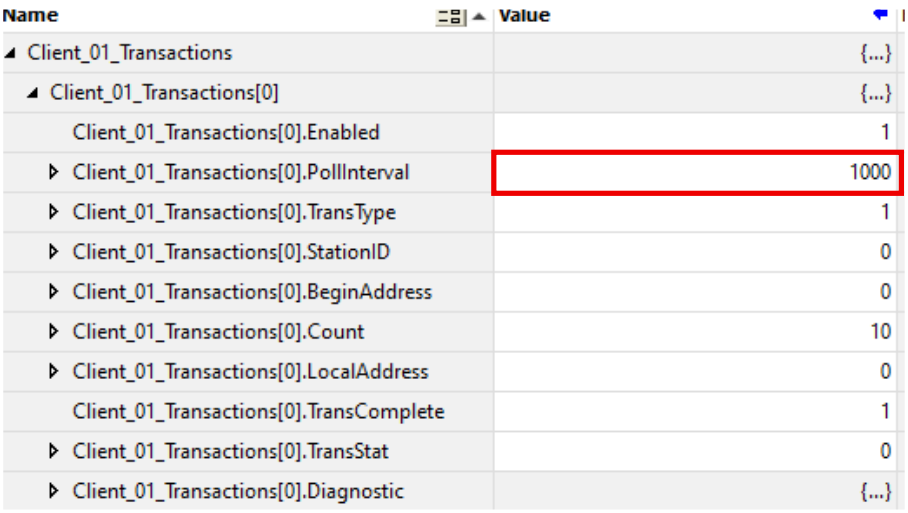


Figura 20. Transcación del Cliente.

Configura el Polling Interval en milisegundos

- Valor por defecto: **1000 ms** (1 segundo).
- Valor mínimo: **80 ms**.

Nota: Si se establece un valor inferior a 80 ms, el sistema forzará el intervalo a 1000 ms *automáticamente*.

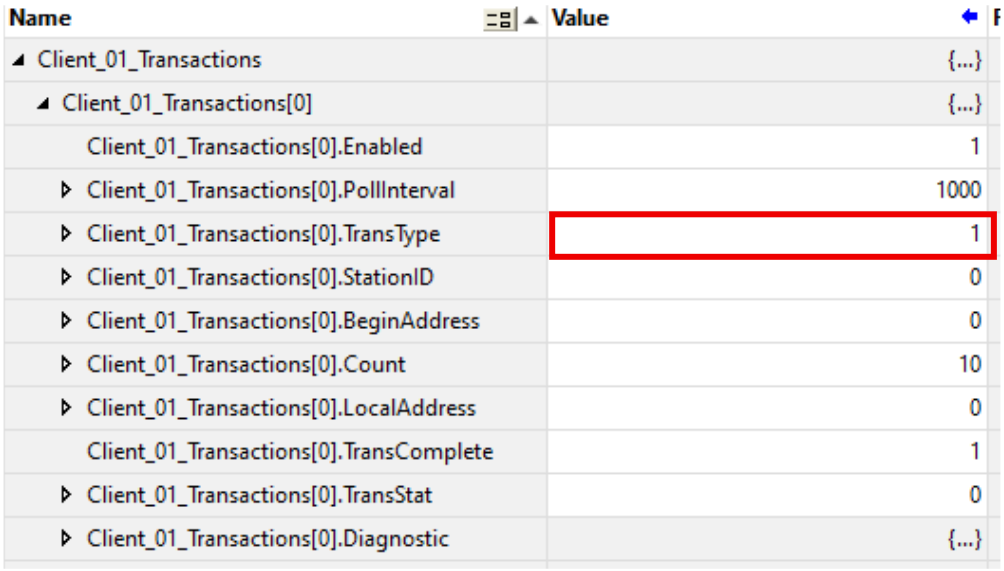


Name	Value
Client_01_Transactions	{...}
Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
Client_01_Transactions[0].PollInterval	1000
Client_01_Transactions[0].TransType	1
Client_01_Transactions[0].StationID	0
Client_01_Transactions[0].BeginAddress	0
Client_01_Transactions[0].Count	10
Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	1
Client_01_Transactions[0].TransStat	0
Client_01_Transactions[0].Diagnostic	{...}

Figura 21. Polling Interval.

Configuración TransType

- Establecer el código de función Modbus en la etiqueta TransType. Consulte la sección de códigos Modbus soportados para más opciones.

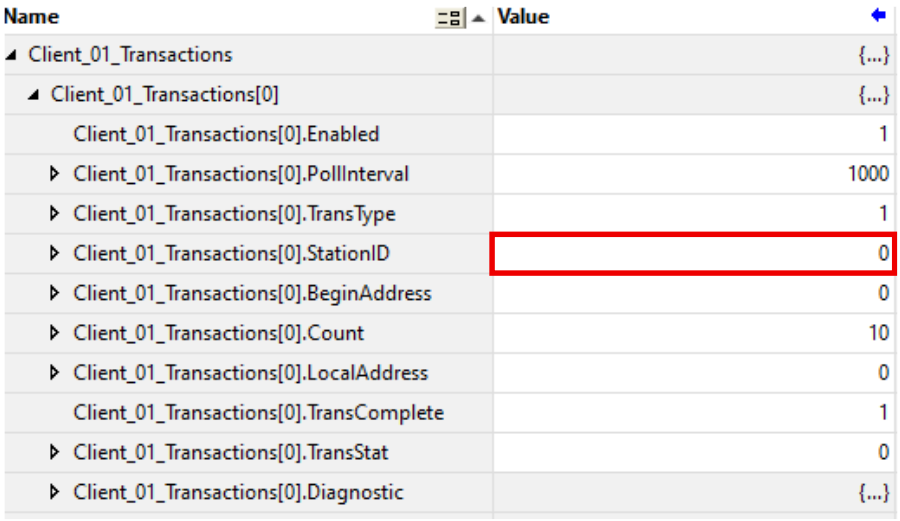


Name	Value
Client_01_Transactions	{...}
Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
Client_01_Transactions[0].PollInterval	1000
Client_01_Transactions[0].TransType	1
Client_01_Transactions[0].StationID	0
Client_01_Transactions[0].BeginAddress	0
Client_01_Transactions[0].Count	10
Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	1
Client_01_Transactions[0].TransStat	0
Client_01_Transactions[0].Diagnostic	{...}

Figura 22. Transaction Type.

Configuración del valor de StationID

- Sólo es necesario si el servidor Modbus lo requiere (en la mayoría de los casos se puede dejar en 0).

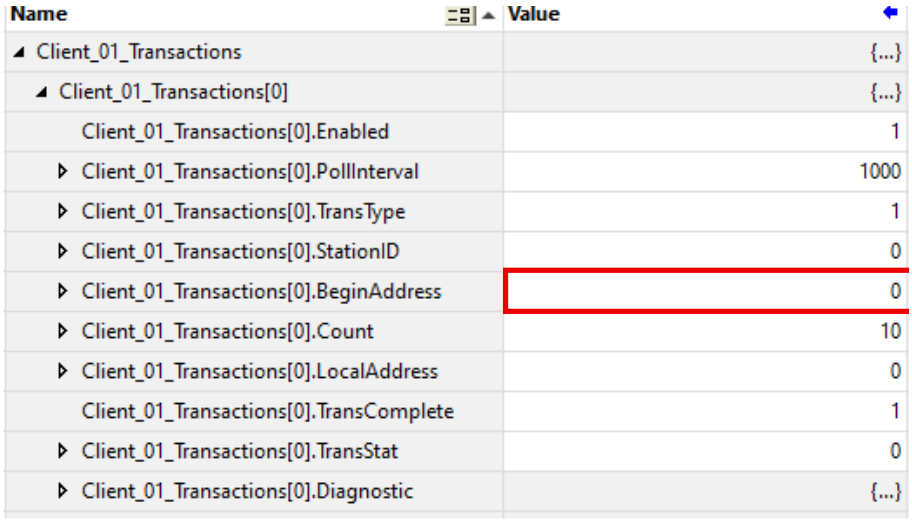


Name	Value
Client_01_Transactions	{...}
Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
Client_01_Transactions[0].PollInterval	1000
Client_01_Transactions[0].TransType	1
Client_01_Transactions[0].StationID	0
Client_01_Transactions[0].BeginAddress	0
Client_01_Transactions[0].Count	10
Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	1
Client_01_Transactions[0].TransStat	0
Client_01_Transactions[0].Diagnostic	{...}

Figura 23. Station ID.

Definir BeginAddress

- Representa la dirección inicial Modbus en el dispositivo servidor (Modbus TCP Server).



Name	Value
Client_01_Transactions	{...}
Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
Client_01_Transactions[0].PollInterval	1000
Client_01_Transactions[0].TransType	1
Client_01_Transactions[0].StationID	0
Client_01_Transactions[0].BeginAddress	0
Client_01_Transactions[0].Count	10
Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	1
Client_01_Transactions[0].TransStat	0
Client_01_Transactions[0].Diagnostic	{...}

Figura 24. Dirección Inicial Modbus.

Nota: No es necesario escribir direcciones como 40001 o 30001 con el AOI proporcionado por Rocwell Automation. Ya con definir que el TransType es 3 (leer un HR) y que la dirección inicial Modbus es 0, se entenderá que nos referimos a la dirección 40001.

Configuración Count

- Representa la cantidad de ítems que se desean leer o escribir en el dispositivo remoto.

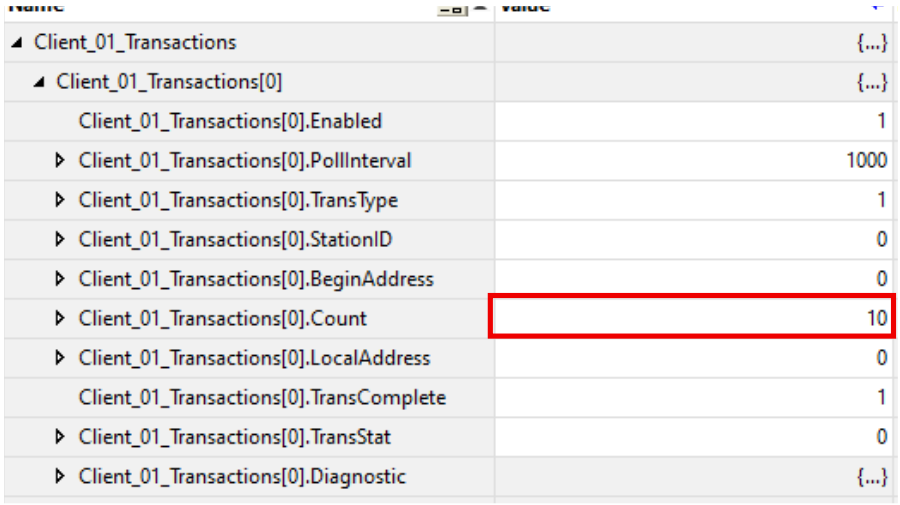


Figura 25. Count.

Nota: Para los códigos de función 1 y 3, permite leer más de una bobina o registro de retención. Sin embargo, para los códigos 5 y 6, sólo permite escribir a una bobina y a un registro. Si se desea escribir a varios, utilizar los códigos 15 y 16 respectivamente.

Configuración LocalAddress

- Representa la dirección inicial dentro de los arrays locales *Data[]* donde se almacenarán (lectura) o desde donde se tomarán los datos (escritura).

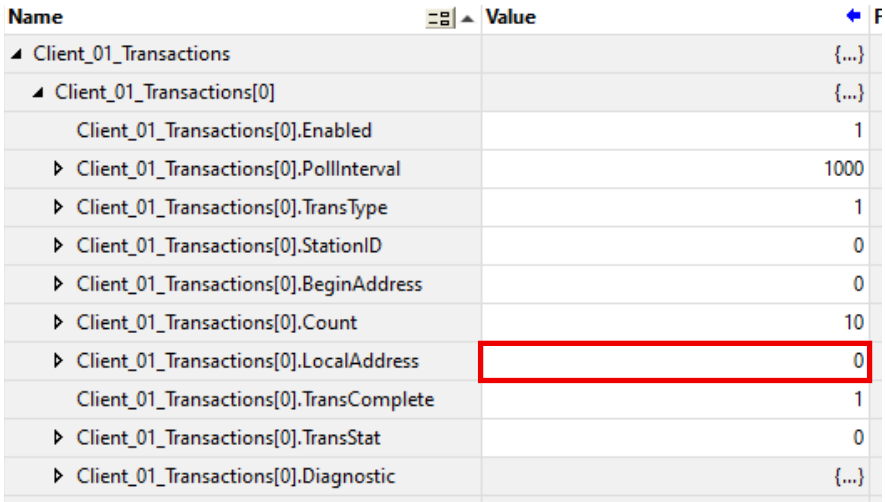


Figura 26. Local Address.

- Iniciar la transacción configurando el bit Enabled en 1.

Supervisión de las operaciones del cliente Modbus TCP

- Verificar que el Client AOI esté correctamente configurado.

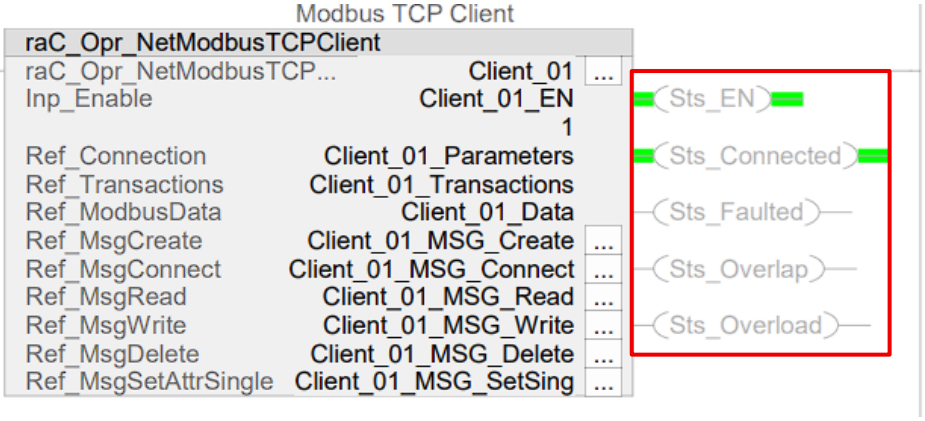


Figura 27. Banderas del Modbus TCP Client.

- La bandera *Sts_EN* indica que la funcionalidad del cliente Modbus TCP está habilitada.
- La bandera *Sts_Connected* indica que la solicitud de conexión del cliente fue aceptada por el servidor. No indica un flujo de datos activo. Se debe verificar el estado de las transacciones individuales para confirmar el intercambio de datos.
- La salida *Sts_Faulted* indica que una de las instrucciones de mensaje tiene una falla.
- La salida *Sts_Overlap* indica que una o más transacciones no se completan antes del siguiente disparo.
- La salida *Sts_Overload* indica un exceso de superposiciones en una o más transacciones.

5.1.2. Configuración Modbus Server en el Contrologix

- Crear un nuevo proyecto en Studio 5000 para configurar el Modbus Server, seleccionar 1756-L73 ControlLogix.

Nota: La revisión del controlador corresponde al Firmware del dispositivo, la revisión se puede observar en el software RSLinx Classic, al dar clic derecho en el equipo y seleccionando sus propiedades.

- En la carpeta I/O Configuration, dar clic derecho en el bus del controlador y seleccionar New Module.

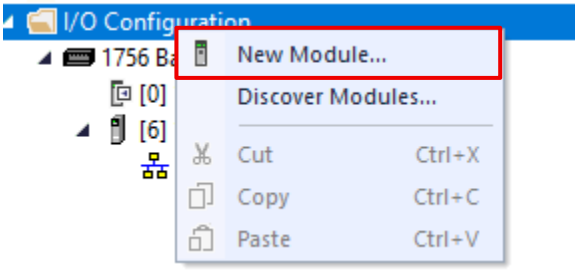


Figura 28. Añadir nuevo módulo.

- La ventana *Select Module Type* aparecerá para buscar por el número de catálogo del módulo que se desea agregar. Añadir el módulo de comunicación Ethernet IP cuyo número de catálogo es 1756-EN2TR. Luego y dar clic en Create.

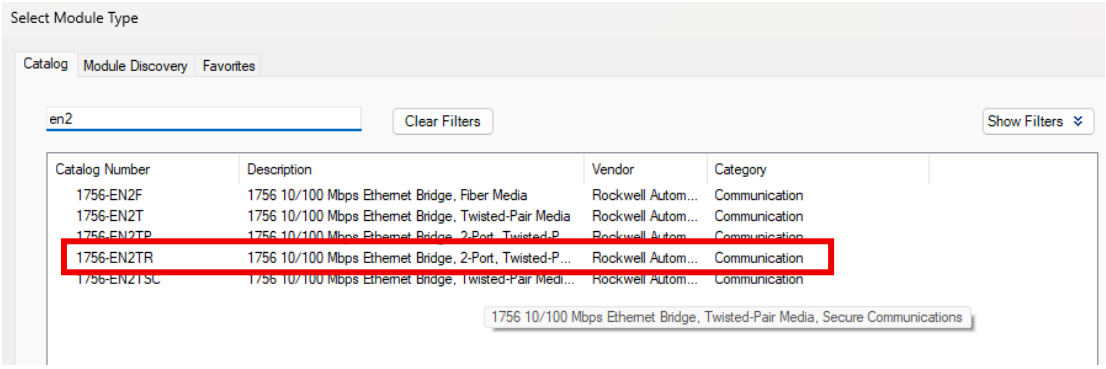


Figura 29. Seleccionar el módulo 1756-EN2TR.

- Abrir RSLinx Classic para revisar los módulos del chasis del PLC ControlLogix. Escoger el controlador de acuerdo con la IP y desplegar lista. Luego, dar clic derecho en el módulo 1756- EN2TR para observar la posición del slot y revisión.

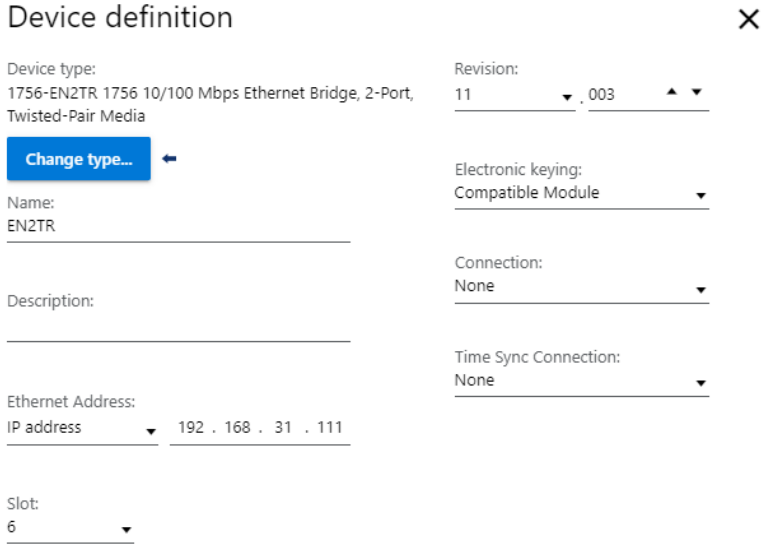


Figura 30. Configurar el módulo 1756-EN2TR.

- Una vez añadido el módulo, importaremos el AOI Modbus Server en la MainRoutine.

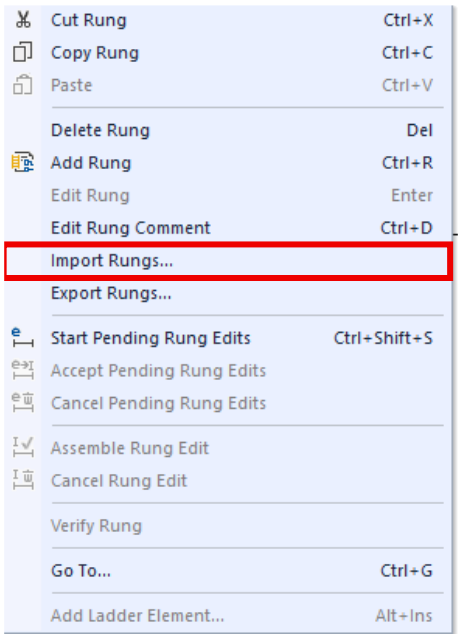


Figura 31. Importar Rungs.

- Seleccionar el raC_Opr_NetModbusTCPServer_Rung.L5x.

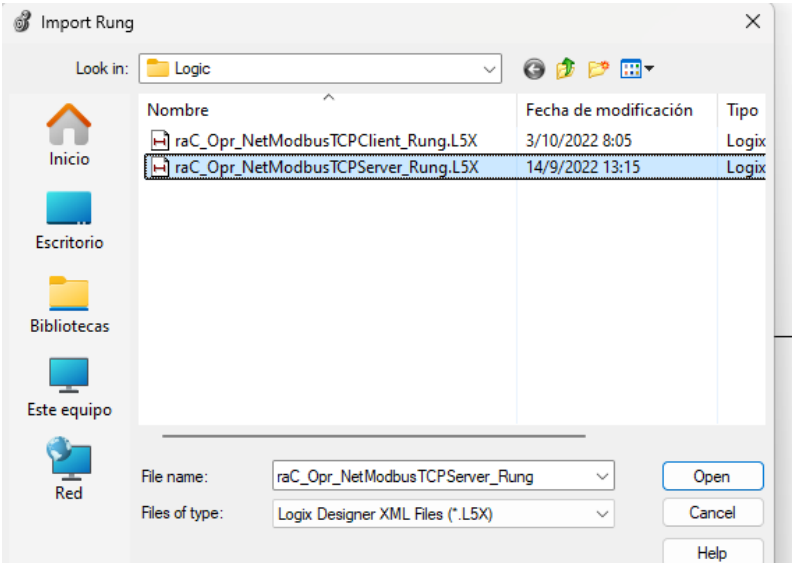


Figura 32. Seleccionar el rung correspondiente.

- Una vez importado, irse a los parámetros del AOI y especificar el slot del módulo EN2TR (En este caso se encuentra en el Slot 6).

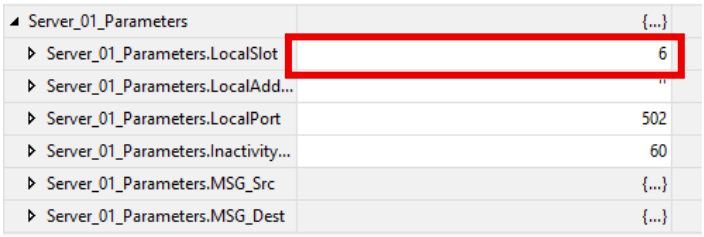


Figura 33. Especificar el LocalSlot del Servidor Modbus.

Nota: Los parámetros siguientes dejarlos por default.

- Irse a los parámetros del Server o Client para poder leer y escribir datos.

Visualización de datos entre Client y Server

- Coils

Name	Value
Client_01_Data.Coils_0xxx	{...}
Client_01_Data.Coils_0xxx[0]	1
Client_01_Data.Coils_0xxx[1]	0
Client_01_Data.Coils_0xxx[2]	1
Client_01_Data.Coils_0xxx[3]	0
Client_01_Data.Coils_0xxx[4]	1
Client_01_Data.Coils_0xxx[5]	1
Client_01_Data.Coils_0xxx[6]	0
Client_01_Data.Coils_0xxx[7]	0
Client_01_Data.Coils_0xxx[8]	0
Client_01_Data.Coils_0xxx[9]	0
Client_01_Data.Coils_0xxx[10]	0

Name	Value
Server_01_Data.Coils_0xxx	{...}
Server_01_Data.Coils_0xxx[0]	1
Server_01_Data.Coils_0xxx[1]	0
Server_01_Data.Coils_0xxx[2]	1
Server_01_Data.Coils_0xxx[3]	0
Server_01_Data.Coils_0xxx[4]	1
Server_01_Data.Coils_0xxx[5]	1
Server_01_Data.Coils_0xxx[6]	0
Server_01_Data.Coils_0xxx[7]	0
Server_01_Data.Coils_0xxx[8]	0
Server_01_Data.Coils_0xxx[9]	0
Server_01_Data.Coils_0xxx[10]	0

Figura 34. Bobinas.

- Inputs Registers

Name	Value
Server_01_Data.InpRegisters_3xxx	{...}
Server_01_Data.InpRegisters_...	50
Server_01_Data.InpRegisters_...	35
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	3460
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0

Name	Value
Server_01_Data.InpRegisters_3xxx	{...}
Server_01_Data.InpRegisters_...	50
Server_01_Data.InpRegisters_...	35
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	3460
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0
Server_01_Data.InpRegisters_...	0

Figura 35. Registros de Entrada.

- Holding Registers

Name	Value
Client_01_Data.HoldRegisters_4xxx	{...}
Client_01_Data.HoldRegisters_4xxx[0]	56
Client_01_Data.HoldRegisters_4xxx[1]	5785
Client_01_Data.HoldRegisters_4xxx[2]	0
Client_01_Data.HoldRegisters_4xxx[3]	4664
Client_01_Data.HoldRegisters_4xxx[4]	544
Client_01_Data.HoldRegisters_4xxx[5]	0
Client_01_Data.HoldRegisters_4xxx[6]	0
Client_01_Data.HoldRegisters_4xxx[7]	0
Client_01_Data.HoldRegisters_4xxx[8]	0
Client_01_Data.HoldRegisters_4xxx[9]	0
Client_01_Data.HoldRegisters_4xxx[10]	650
Client_01_Data.HoldRegisters_4xxx[11]	0

Name	Value
Server_01_Data.HoldRegisters_4xxx	{...}
Server_01_Data.HoldRegisters_4xxx[0]	56
Server_01_Data.HoldRegisters_4xxx[1]	5785
Server_01_Data.HoldRegisters_4xxx[2]	0
Server_01_Data.HoldRegisters_4xxx[3]	4664
Server_01_Data.HoldRegisters_4xxx[4]	544
Server_01_Data.HoldRegisters_4xxx[5]	0
Server_01_Data.HoldRegisters_4xxx[6]	0
Server_01_Data.HoldRegisters_4xxx[7]	567
Server_01_Data.HoldRegisters_4xxx[8]	0
Server_01_Data.HoldRegisters_4xxx[9]	9
Server_01_Data.HoldRegisters_4xxx[10]	650
Server_01_Data.HoldRegisters_4xxx[11]	0

Figura 36. Registros de Retención.

5.2. Comunicación Modbus TCP/IP entre ControlLogix y Node-RED

- Ejecutar el comando *node-red* desde Node.js command prompt.
- Dar Ctrl+Clic en la url del servidor Node-RED.

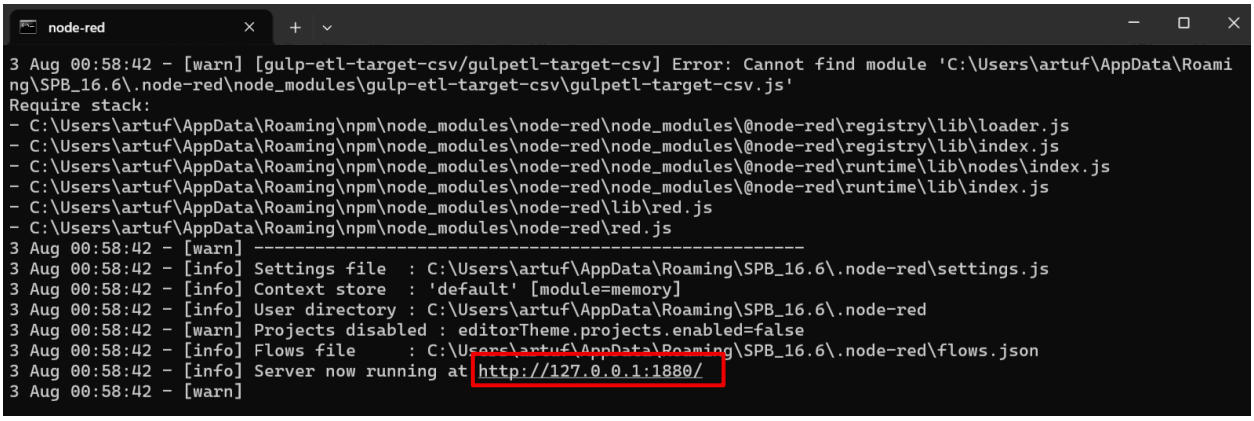


Figura 37. Node.js Command Prompt.

- Debe aparecerle el espacio de trabajo de Node-RED.

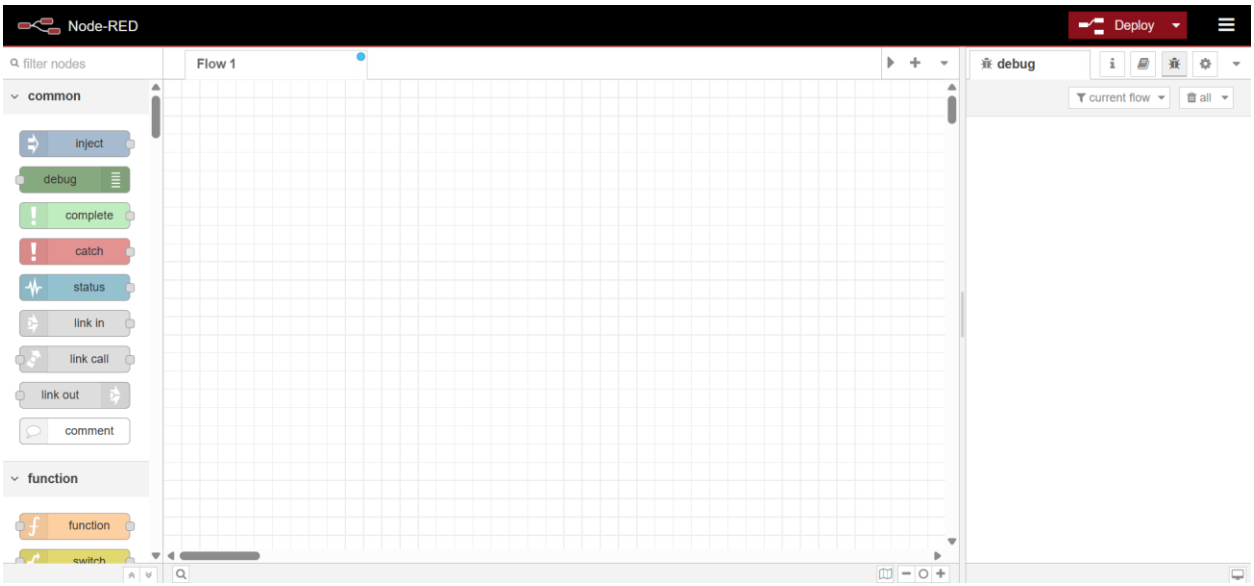


Figura 38. Node-RED.

- Ir a Configuración (a lado de Deploy) y dirigirse a la sección *Manage Palette* para instalar la extensión Modbus. En la pestaña *Palette*, dirigirse a *Install*, y en el buscador escribir Modbus. Instalar la extensión **node-red-contrib-modbus**. (Si ya se encuentra instalada le aparecerá el indicador installed).

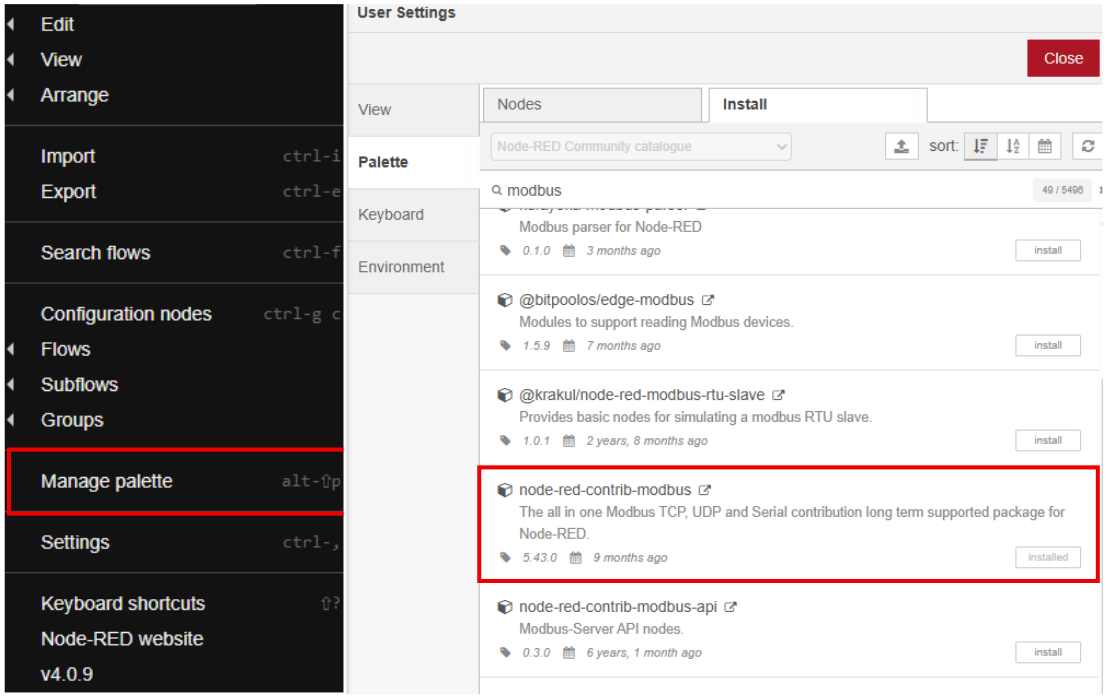


Figura 39. Instalar la extensión Modbus.

- Regresar al espacio de trabajo. Verificar que le aparezca en la sección de Nodos el grupo Modbus.

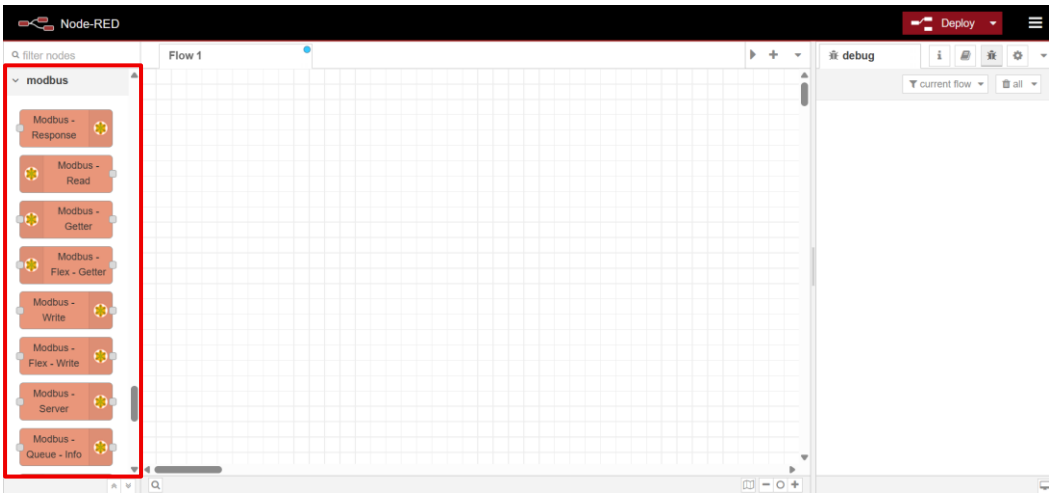


Figura 40. Grupo Modbus.

- Arrastrar un nodo *Modbus-Write* al espacio de trabajo. Dar doble clic al nodo. Buscar en *Settings* la propiedad *Server*. Dar clic en + para una crear un nodo de configuración Modbus Server.

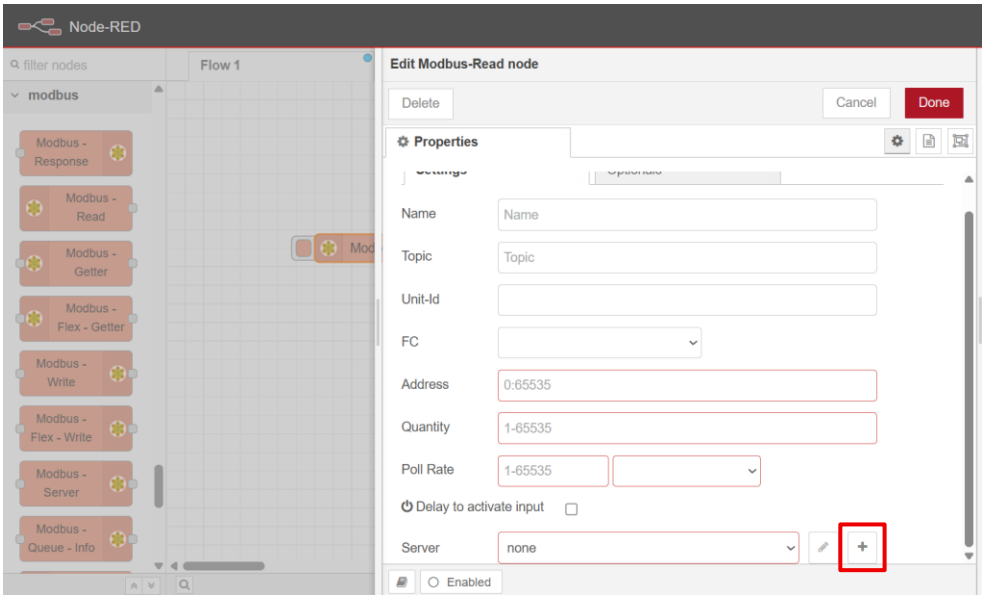


Figura 41. Nodo Modbus-Write.

- Realizar la configuración del Modbus Server. En *Name* colocar el nombre que usted crea conveniente. En *Type* seleccionar TCP. En *Host* colocar la dirección IP del Modbus Server (ControlLogix). En *Port* dejar el valor de 502 por default. En TCP Type dejar por DEFAULT. En *Unit-Id* colocar 0. En Timeout (ms) y Reconnect Timeout (ms) dejar los valores por default (se podrían cambiar según la aplicación). Dar clic en Add.

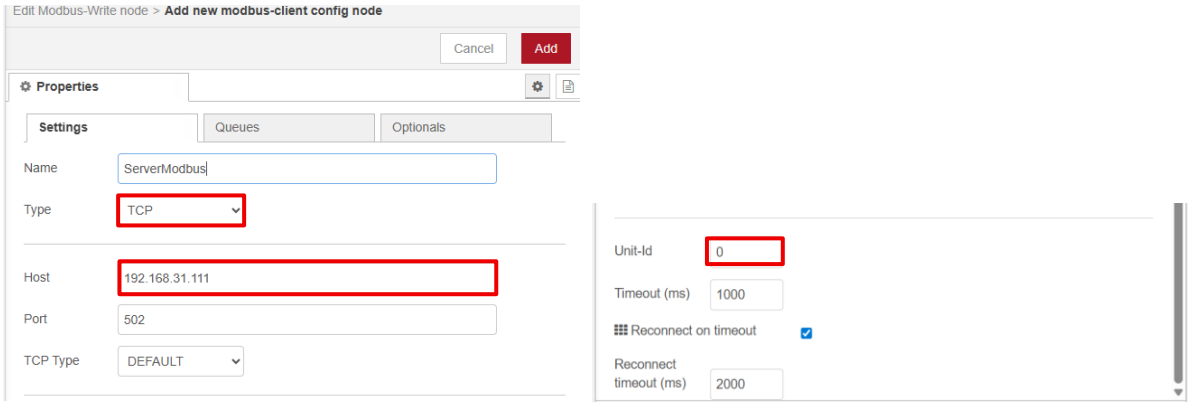


Figura 42. Configuración Modbus Server.

Nota: El Unit-ID es lo mismo que el Station-ID. Hay que recordar que este se usa típicamente para identificar el *dispositivo esclavo (servidor)* al que se quiere acceder. El cliente incluye el Unit-ID

en su solicitud, y el servidor lo recibe para saber a qué dispositivo interno (en caso de gateways o multiplexores) debe redirigir la petición.

Casos:

- Si se tiene un servidor Modbus TCP/IP "puro" (como un PLC ControlLogix actuando como servidor), el Unit ID se ignora o se espera que sea 0 o 1 por convención.
- Si se tiene un gateway Modbus TCP/IP a RTU, el Unit ID *sí importa*, porque indica cuál esclavo RTU (por dirección) debe ser accedido.

- Configurar los nodos de escritura o lectura con sus respectivas propiedades según corresponda la aplicación (Unitd-ID, Código de Función, Dirección, Cuenta).

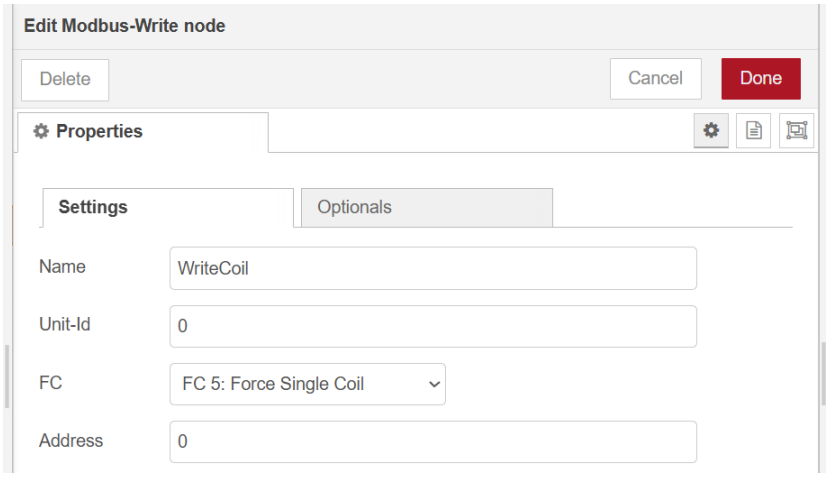


Figura 43. Configuración Modbus-Write Node.

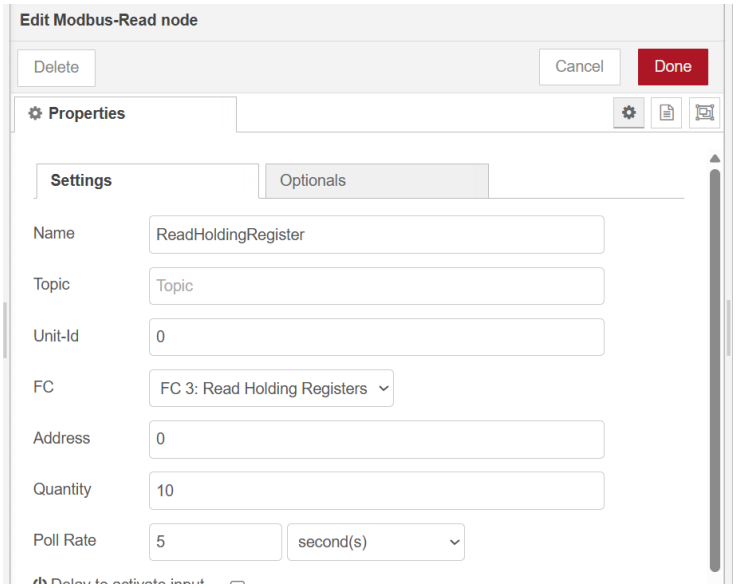


Figura 44. Configuración Modbus-Read Node.

- Realizar el siguiente flujo de prueba para escribir una bobina y leer varios registros de retención. Añadir nodos de inyección para escribir un valor de true o false a la bobina. Añadir nodos de debug para visualizar los mensajes de lectura y escritura.

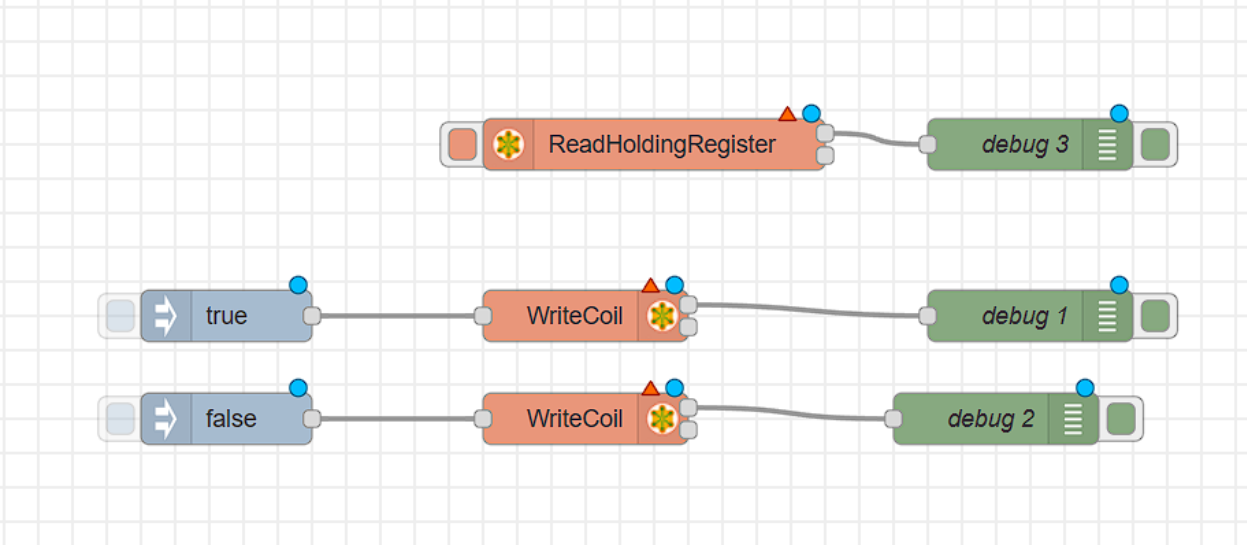


Figura 45. Flujo de Prueba.

- Una vez configurado todo se puede observar en la pestana de debugs los valores que se están escribiendo y leyendo en el ControlLogix.

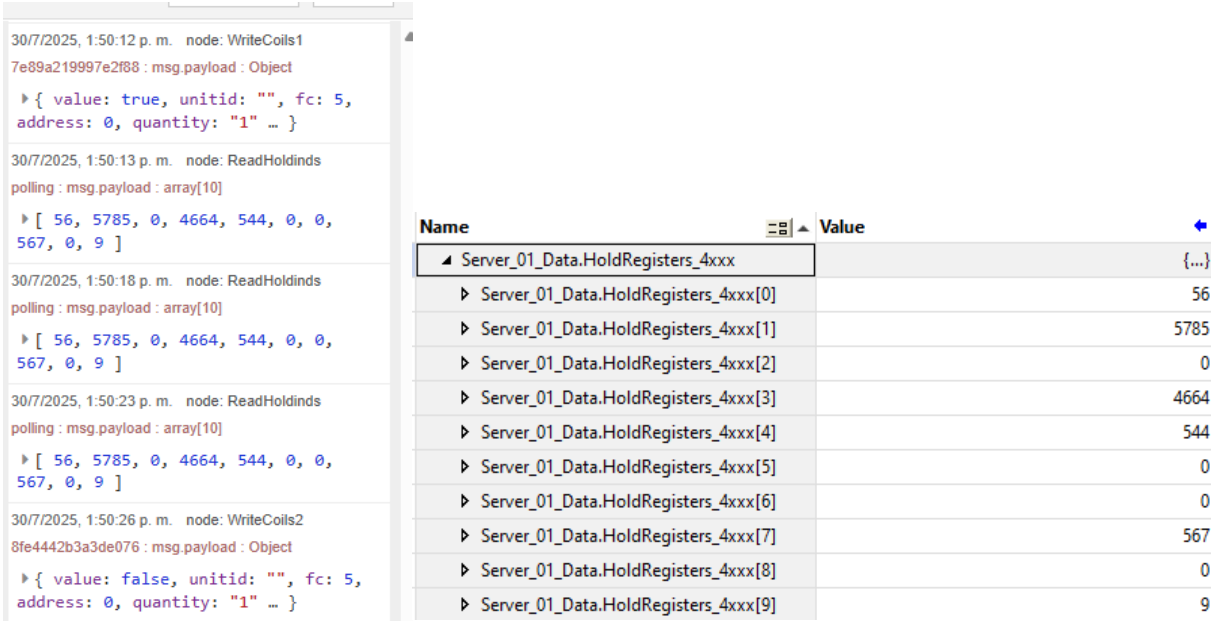


Figura 46. Resultados de la comunicación.

5.3. Comunicación Modbus TCP/IP entre CompactLogix y Sentron PAC3200

Configuración de Conexiones Sentron PAC3200

- Realizar la medición de tensión L-N mediante las entradas voltimétricas. Para ello debe conectar en **(3)** la entrada de alimentación, luego puentear respectivamente a **(4)** que son las entradas de medida de tensión.
- Realizar la medición de corriente de línea mediante las entradas amperimétricas. Para ello debe conectar al TC en **(5)** que son las entradas de medida de corriente.
- Conectar el multímetro a la red mediante **(8)** que es el puerto de conexión Ethernet.

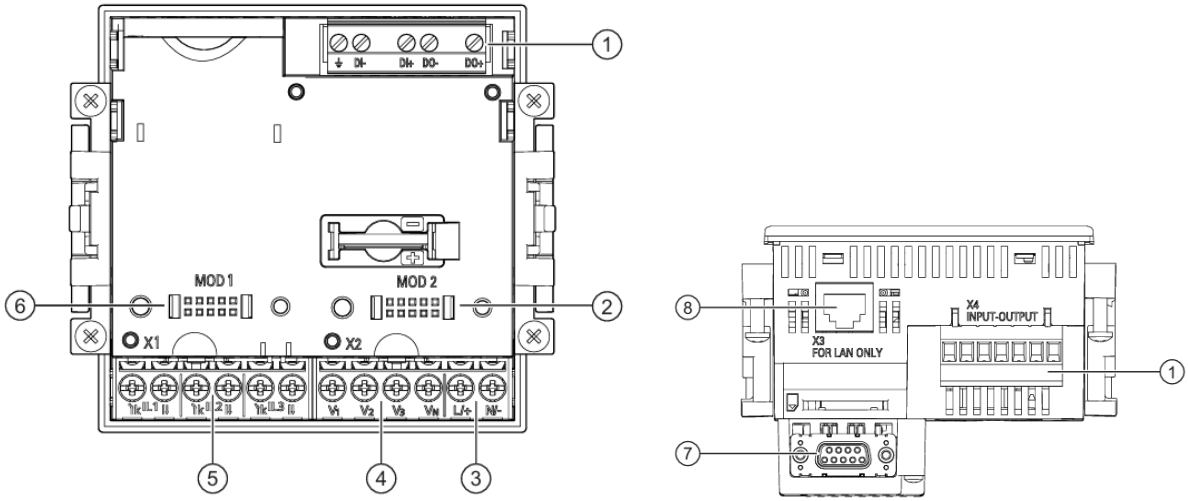


Figura 47. Designación de conexiones.

- Debería aparecerles lo siguiente al conectar la entrada de alimentación.

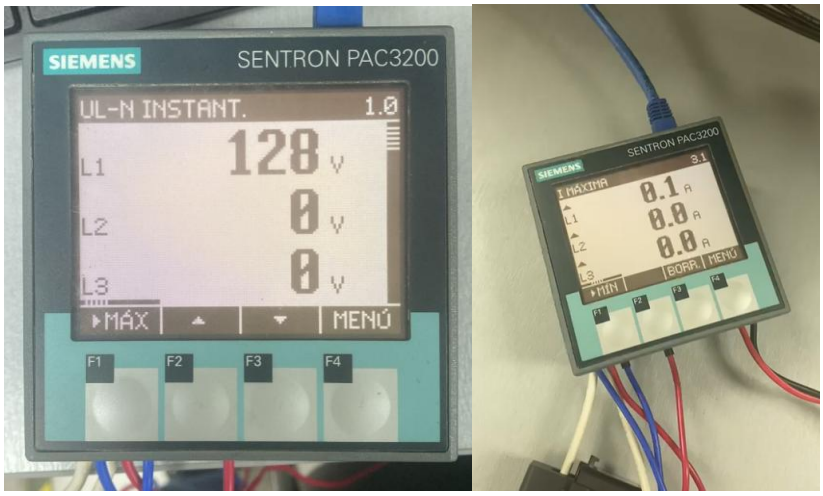


Figura 48. Sentron PAC3200 en funcionamiento.

Nota: Para la medición de corriente se debe realizar una configuración adicional, debido a que se está utilizando un Transformador de Corriente (TC), y por tanto debemos registrar sus datos de placa en el Sentron. Para ello dirigirse a Menú mediante la tecla F4, luego dirigirse a Ajustes, luego dirigirse a Parámetros Básicos y seleccionar la opción Entradas Corriente. Ingresar los siguientes datos de placa del TC: Corriente en Primario, Corriente en Secundario, Límite de Corriente, y si se desea invertir la polaridad del TC.

Configuración de Comunicación Sentron PAC3200

- En Menú, digirse a Ajustes. Luego, dirigirse a Comunicación. Mediante la tecla F4 puede editar la Dirección IP, Submáscara de Red y Gateway según respecte.
- En Protocolo seleccionar Modbus TCP.
- Presionar la tecla F1 para salir, le pedirá reiniciar si se ha realizado un cambio, colocarle que OK mediante F4. Esperar que el dispositivo se reinicie.

Figura 49. Configuración de Comunicación Sentron PAC3200.

Configuración de Modbus Client en Studio 5000

- Inicialmente, se debe crear un proyecto en Studio 5000 Logix Designer para configurar el cliente de la comunicación Modbus, el cual es un CompactLogix con CPU modelo 1769-L33ERM. Considerar la revisión de Firmware.
- Seguir todos los pasos para la importación del rung del Modbus Client.
- Dirigirse a Client_01_Parameters y colocar la siguiente configuración. En *DestAddress* se coloca la dirección IP del Sentron.

Client_01_Parameters	{...}
Client_01_Parameters.LocalSlot	0
Client_01_Parameters.LocalAddress	''
Client_01_Parameters.DestAddress	'192.168.31.202'
Client_01_Parameters.DestinationPort	502

Figura 50. Client Parameters.

- Debido a que se pretende leer dos parámetros (Tensión UL1-N y Corriente máxima L1) se deben configurar dos transacciones. A continuación, la información de los parámetros a leer.

Tabla 3-6 Magnitudes medidas disponibles

Offset	Número de registros	Nombre	Formato	Unidad	Rango admitido	Acceso
1	2	Tensión U_{L1-N}	Float	V	-	R
3	2	Tensión U_{L2-N}	Float	V	-	R
5	2	Tensión U_{L3-N}	Float	V	-	R
7	2	Tensión U_{L1-L2}	Float	V	-	R

Offset	Número de registros	Nombre	Formato	Unidad	Rango admitido	Acceso
77	2	Tensión máxima U_{L2-N}	Float	V	-	R
79	2	Tensión máxima U_{L3-N}	Float	V	-	R
81	2	Tensión máxima U_{L1-L2}	Float	V	-	R
83	2	Tensión máxima U_{L2-L3}	Float	V	-	R
85	2	Tensión máxima U_{L3-L1}	Float	V	-	R
87	2	Corriente máxima L1	Float	A	-	R
89	2	Corriente máxima L2	Float	A	-	R
91	2	Corriente máxima L3	Float	A	-	R

Figura 51. Client Parameters.

- En Client_01_Transactions[0] se solicitará la lectura del parámetro Tensión UL1-N. Colocar el valor de 1 en *Enabled*. Colocar un *Polling Interval* de 1000 ms (puede cambiarse según la aplicación). Colocar en *Transaction Type* el valor de 3 para realizar la función Read Holding Register. Colocar en *StationID* el valor de 1 (esto es propio del Sentron, ya que si se coloca 0 la comunicación será errónea). Colocar en *Begin Address* el valor de 1 (el offset del parámetro a leer). Colocar en *Count* el valor de 2 (dado que es tipo Float, ocupa 2 HR). Colocar en *Local Address* el valor de 0 para que empiece a almacenar desde el registro [0].

◢ Client_01_Transactions	{...}
◢ Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
▷ Client_01_Transactions[0].PollInterval	1000
▷ Client_01_Transactions[0].TransType	3
▷ Client_01_Transactions[0].StationID	1
▷ Client_01_Transactions[0].BeginAddress	1
▷ Client_01_Transactions[0].Count	2
▷ Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	0
▷ Client_01_Transactions[0].TransStat	1
▷ Client_01_Transactions[0].Diagnostic	{...}

Figura 52. Client Transaction 1.

- En Client_01_Transactions[1] se solicitará la lectura del parámetro Corriente máxima L1. Colocar el valor de 1 en *Enabled*. Colocar un *Polling Interval* de 1000 ms (puede cambiarse según la aplicación). Colocar en *Transaction Type* el valor de 3 para realizar la función Read Holding Register. Colocar en *StationID* el valor de 1 (esto es propio del Sentron, ya que si se coloca 0 la comunicación será errónea). Colocar en *Begin Address* el valor de 87 (el offset del parámetro a leer). Colocar en *Count* el valor de 2 (dado que es tipo Float, ocupa 2 HR). Colocar en *Local Address* el valor de 2 para que empiece a almacenar desde el registro [2].

◢ Client_01_Transactions[1]	{...}
Client_01_Transactions[1].Enabled	1
▷ Client_01_Transactions[1].PollInterval	1000
▷ Client_01_Transactions[1].TransType	3
▷ Client_01_Transactions[1].StationID	1
▷ Client_01_Transactions[1].BeginAddress	87
▷ Client_01_Transactions[1].Count	2
▷ Client_01_Transactions[1].LocalAddress	2
Client_01_Transactions[1].TransComplete	0
▷ Client_01_Transactions[1].TransStat	1

Figura 53. Client Transaction 2.

- Habilitar la instrucción Modbus Client y verificar que se haya ejecutado sin errores. Luego, dirigirse a Client_01_Data y desplegar los Holding Registers. Verificar que le aparezca algo similar a la siguiente figura.

◢ Client_01_Data	{...}	{...}
▷ Client_01_Data.Coils_0xxx	{...}	{...}
▷ Client_01_Data.DiscInputs_1xxx	{...}	{...}
▷ Client_01_Data.InpRegisters_3xxx	{...}	{...}
◢ Client_01_Data.HoldRegisters_4xxx	{...}	{...}
▷ Client_01_Data.HoldRegisters_4xxx[0]	17144	
▷ Client_01_Data.HoldRegisters_4xxx[1]	-29319	
▷ Client_01_Data.HoldRegisters_4xxx[2]	15713	
▷ Client_01_Data.HoldRegisters_4xxx[3]	14508	

Figura 54. Holding Registers obtenidos.

Nota: Para poder proseguir con la comunicación debemos intentar interpretar qué es lo que estamos recibiendo en nuestros registros de retención. Según como se configuraron las transacciones, los dos primeros registros [0] y [1] pertenecen a la Tensión UL1-N, mientras que los dos siguientes a la Corriente máxima L1. Se dará cuenta que el primer registro de ambos parámetros por momentos es fijo, mientras que el segundo cambia con cada polling (hasta a valores negativos). Lo que hay que entender de aquí es que estamos leyendo parámetros de tipo Float, por lo que en un registro *se debería almacenar la parte entera*, mientras que en el otro *la parte flotante*

o decimal. Por lógica, la parte decimal o flotante tiende a ser mucho más fluctuante que la parte entera, por lo que, por inspección, ya podemos interpretar estos valores enteros que estamos recibiendo. Por ejemplo, según la captura, para el primer parámetro, la parte entera en formato entero seria el valor 17144, y la parte decimal o flotante -29319. Lo mismo se aplicaría para el segundo parámetro. Además, podemos considerar a la parte entera como AB, y a la parte flotante como CD, por lo que el parámetro completo sería ABCD. Nótese que, el ordenamiento es Big-Endian, tal como habíamos explicado del protocolo. De todas formas, no todos los fabricantes trabajan de la misma forma, y por tanto habría que inspeccionar más a nivel de bits estos valores enteros (en el caso que haya que realizar operaciones de SWAP para cambiar el ordenamiento).

En realidad lo correcto sería explorar todas las posibles combinaciones factibles de ordenamiento de las palabras (registros de retención) debido al endianismo previamente explicado. Por ejemplo, para el segundo parámetro, un primer paso sería transformar los valores enteros recibidos a un formato hexadecimal (debido a que trabajamos con palabras) como sigue (se utilizó https://binaryconvert.com/):

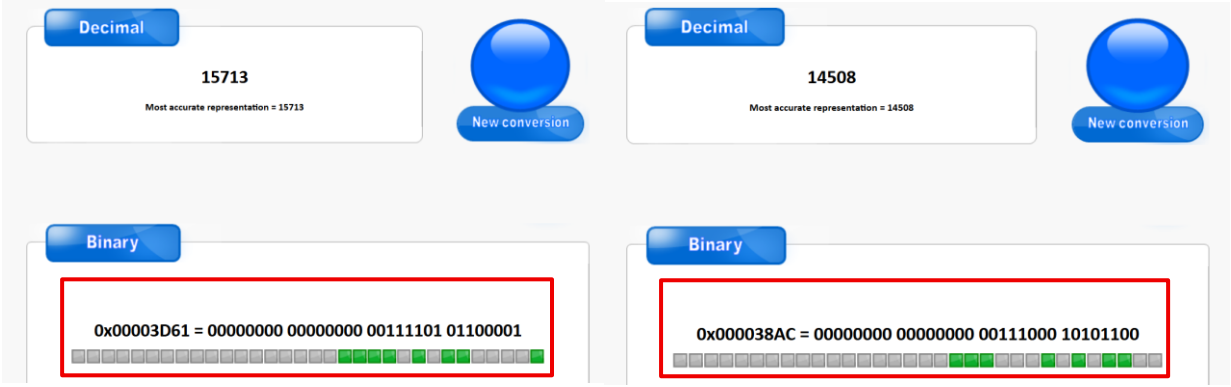


Figura 55. Conversión Decimal a Binario/Hexadecimal.

De esto observamos que el registro [2] que tiene el valor decimal 15713, en hexadecimal sería 0x00003D61; mientras que el registro registro [3] que tiene el valor decimal 14508, en hexadecimal sería 0x000038AC. Dado que esperamos un dato REAL o Float de 32 bits, lo que deberíamos hacer es concatenar ambos valores hexadecimales. Debido a que también desconocemos a priori el ordenamiento, podríamos probar concatenarlo de la siguiente manera (el dato vendría como 0x38AC3D61, lo cual sería Little-Endian) y transformarlo a Float:

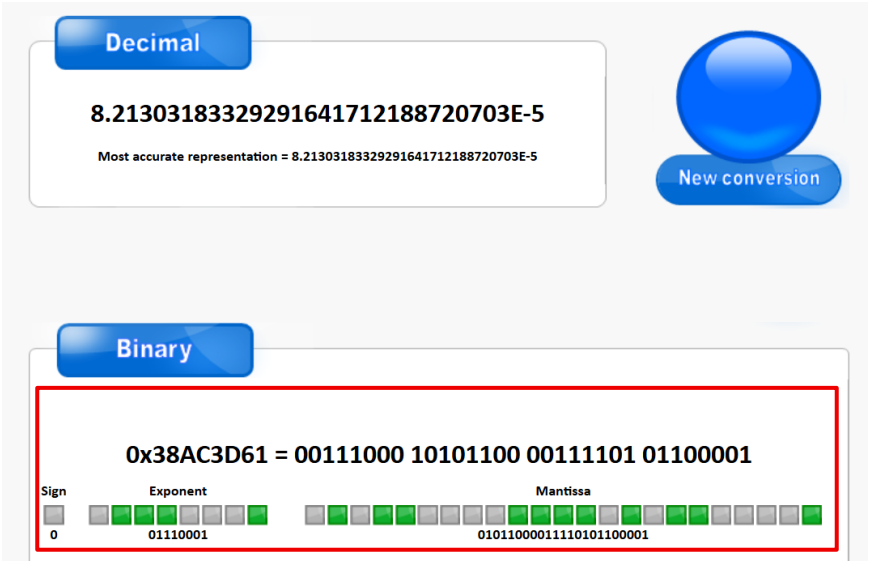


Figura 56. Conversión Binario/Hexadecimal a Real o Float.

De esto observamos que en teoría el valor leído de corriente sería aproximadamente 8.213e-5 A. Pero, este valor no concuerda con lo observado en el Sentron. Por lo que, probamos ahora con el ordenamiento Big-Endian (0x3D6138AC):

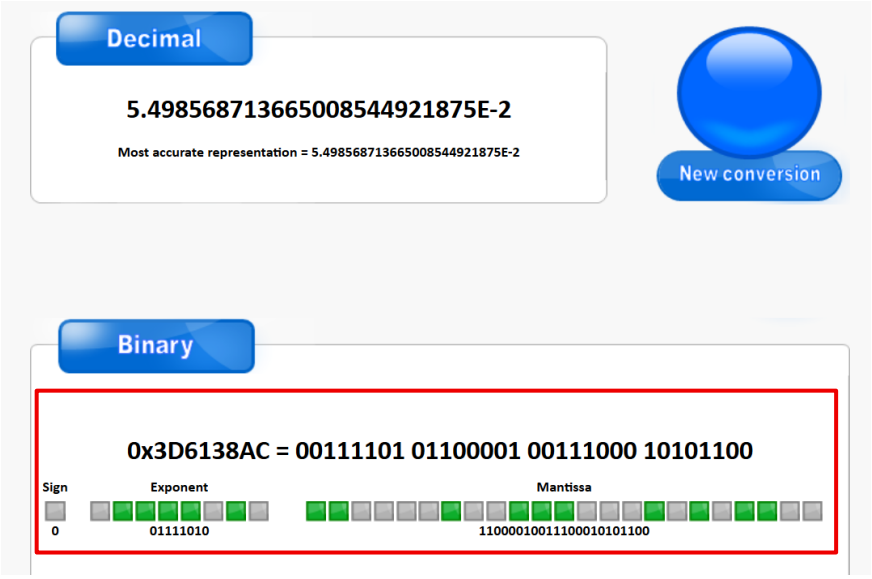


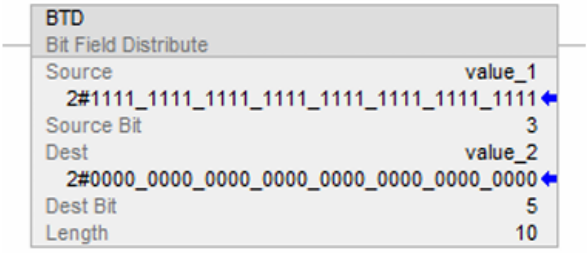
Figura 57. Conversión Binario/Hexadecimal a Real o Float.

De esto observamos que el valor leído de corriente sería aproximadamente 5.498e-2 A. Lo cual sí concuerda con lo observado en el Sentron. Con esto podemos generalizar el ordenamiento para todos los parámetros, y por tanto la parte entera se almacenerá en el registro [i], mientras que la

parte flotante en el [i+1]. El ordenamiento es Big-Endian y no se necesita realizar ninguna operación de SWAP. Ahora, debemos realizar dicha concatenación binaria en Studio 5000.

- Para concatenar ambos registros de retención en un solo dato DINT (entero de 32 bits) en ordenamiento Big-Endian debemos implementar la siguiente operación a nivel de bits: $\text{ValueAsDint} = (\text{HR}[0] \ll 16)||(\text{HR}[1])$. Con esta operación, desplazamos el contenido binario del primer registro 16 bits a la izquierda (debido que es el MSB), y a eso le sumamos el contenido binario del segundo registro.
- Para implementar aquello, podemos utilizar la instrucción BTD (Distribución de campo de bits), la cual copia los bits especificados del origen, desplaza los bits a la posición adecuada y escribe los bits en el destino. A continuación un ejemplo de la ayuda de Studio 5000 respecto a esta instrucción:

Ejemplo 2



Cuando se habilita, la instrucción BTD mueve 10 bits de value_1 a value_2.

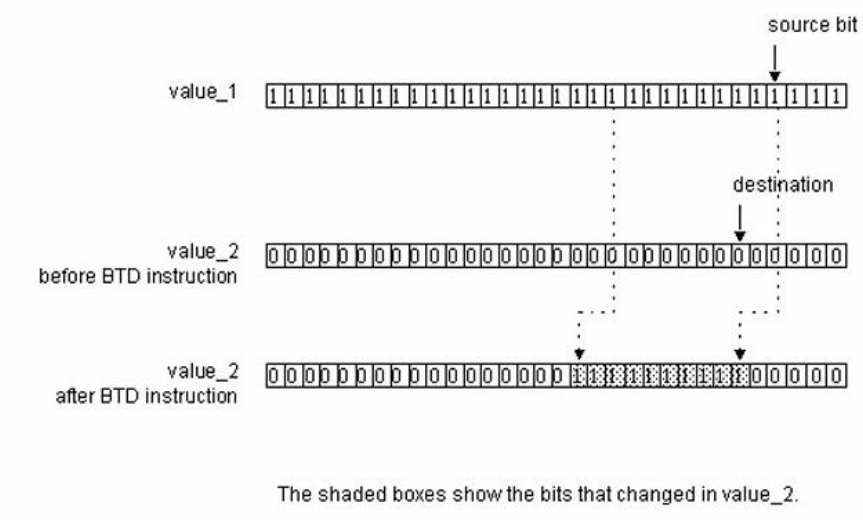


Figura 58. Ejemplo de uso instrucción BTD.

- Una vez entendido el funcionamiento de esta instrucción, se implementa la siguiente lógica para ambos parámetros. Utilizamos dos instrucciones BTD en paralelo para copiar el contenido binario de ambos registros respectivamente en la variable de tipo DINT "VoltL1NAsDint". Nótese que para el registro [0] se configura el Dest Bit como 16, debido al desplazamiento de 16 bits previamente mencionado (ocuparía desde el bit 16 hasta el 31); mientras que el registro [1] se almacena desde el bit 0 hasta el 15. Luego de ello, se coloca una instrucción COP para copiar el contenido binario a la variable de tipo REAL "VoltL1N" (Colocar 32 bits en *Length*). Este procedimiento se repite para el segundo parámetro.

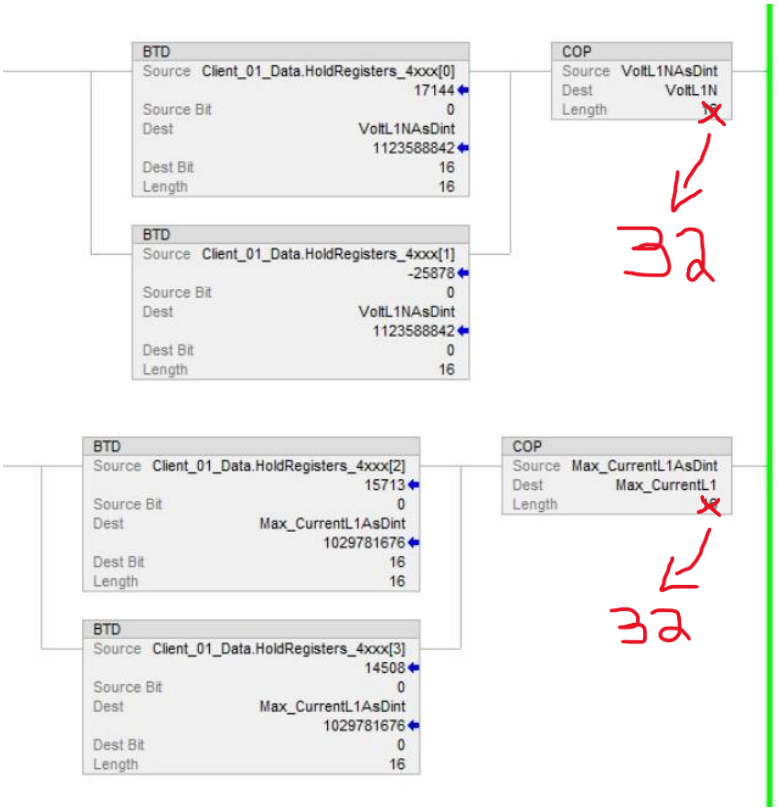


Figura 59. Lógica de Concatenación y Conversión.

- Dirigirse a Controller Tags, luego a Monitor Tags para observar el estado de las variables creadas. Debería aparecerle algo similar a esto:

VoltL1NAsDint	1123578620
VoltL1N	124.22458
Max_CurrentL1AsDint	1029781676
Max_CurrentL1	0.054985687

Figura 60. Parámetros correctamento leídos.

5.4. Implementación de la Arquitectura Propuesta

- Iniciaremos adaptando el código que se descargará al CompactLogix para que este trabaje como un Modbus Client, tanto con el Sentron como para el ControlLogix. Para ello se recomienda estructurar el MainProgram en la MainRoutine y dos subrutinas para las comunicaciones (MB_Client_SentronPAC3200 y MB_Client_ControlLogix).
- Por ejemplo, si estamos adaptando el código donde teníamos únicamente la comunicación con el Sentron (donde las variables importadas empiezan con Client_01), en la nueva subrutina para el Control **debemos volver a importar el archivo para el Modbus Client** (no copiar y pegar el bloque), y **cambiar los nombres de las Tags** para que empiecen con Client_02.

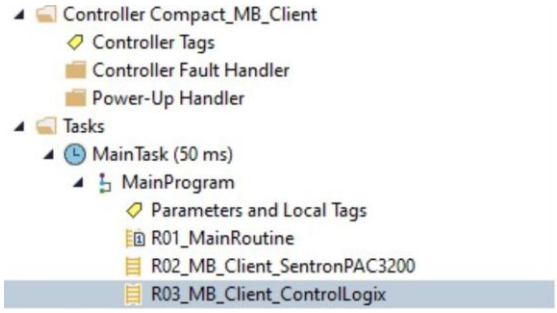


Figura 61. Organización del Programa.

- Revisar que no haya problemas en la importación (verificar que se hayan creado todas las variables necesarias para el Client_02 en Controller Tags).
- A continuación, la lógica de la subrutina MB_Client_SentronPAC3200.

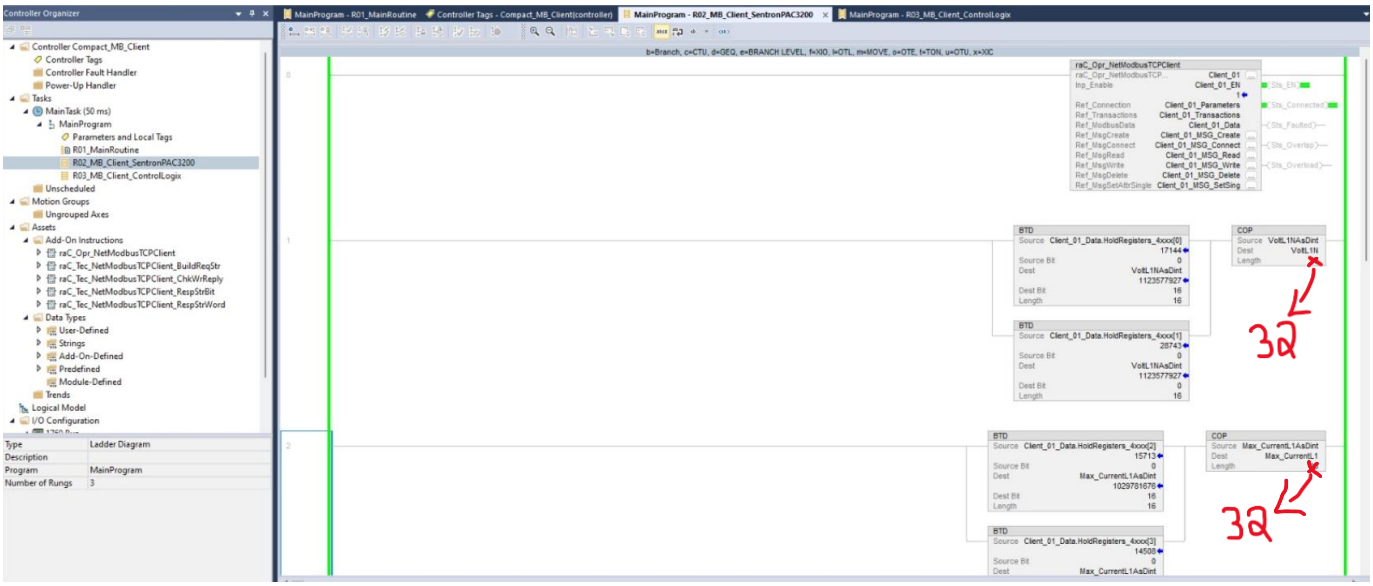


Figura 62. Subrutina MB_Client_SentronPAC3200.

- Realizar la configuración de Client_01_Parameters y Client_01_Transactions siguiendo la comunicación con el Sentron.

◢ Client_01_Parameters	{...}
▶ Client_01_Parameters.LocalSlot	0
▶ Client_01_Parameters.LocalAddress	''
▶ Client_01_Parameters.DestAddress	'192.168.31.202'
▶ Client_01_Parameters.DestinationPort	502

◢ Client_01_Transactions	{...}
◢ Client_01_Transactions[0]	{...}
Client_01_Transactions[0].Enabled	1
▶ Client_01_Transactions[0].PollInterval	1000
▶ Client_01_Transactions[0].TransType	3
▶ Client_01_Transactions[0].StationID	1
▶ Client_01_Transactions[0].BeginAddress	1
▶ Client_01_Transactions[0].Count	2
▶ Client_01_Transactions[0].LocalAddress	0
Client_01_Transactions[0].TransComplete	0
▶ Client_01_Transactions[0].TransStat	1
▶ Client_01_Transactions[0].Diagnostic	{...}
◢ Client_01_Transactions[1]	{...}
Client_01_Transactions[1].Enabled	1
▶ Client_01_Transactions[1].PollInterval	1000
▶ Client_01_Transactions[1].TransType	3
▶ Client_01_Transactions[1].StationID	1
▶ Client_01_Transactions[1].BeginAddress	87
▶ Client_01_Transactions[1].Count	2
▶ Client_01_Transactions[1].LocalAddress	2
Client_01_Transactions[1].TransComplete	0
▶ Client_01_Transactions[1].TransStat	1

Figura 63. Client 01 Parameters y Transactions.

- A continuación, la lógica de la subrutina MB_Client_ControlLogix.

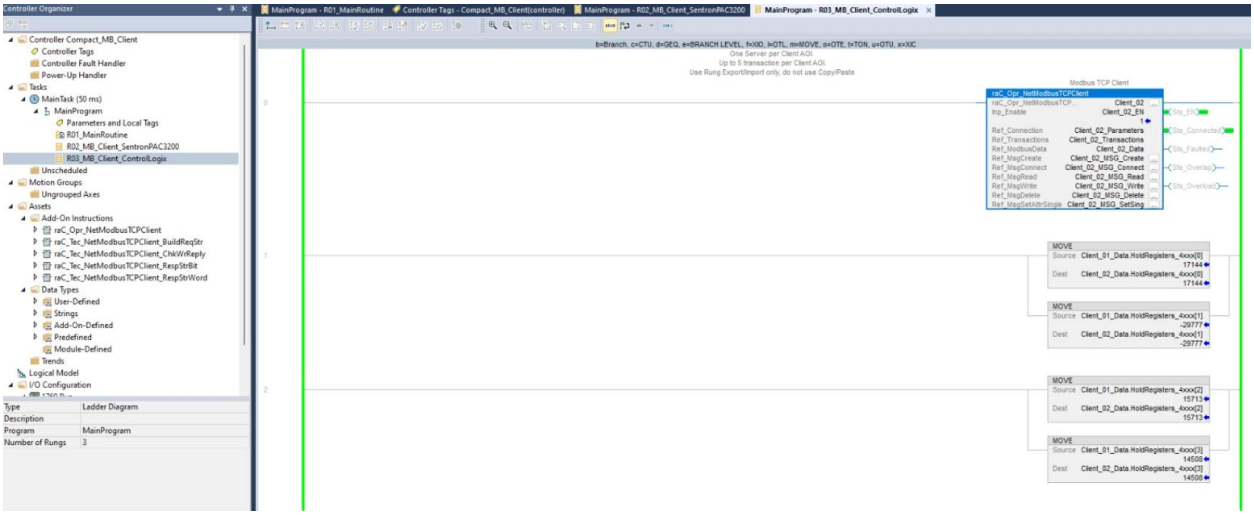


Figura 64. Subrutina MB_Client_ControlLogix.

- En la subrutina de Client_02, lo que se implementa es una lógica con bloques MOVE, de tal manera que el contenido binario de los Holding Registers de Client_01 (Lo que se lee del Sentron), se almacena en los Holding Registers de Client_02 (Lo que se escribirá en el ControlLogix).

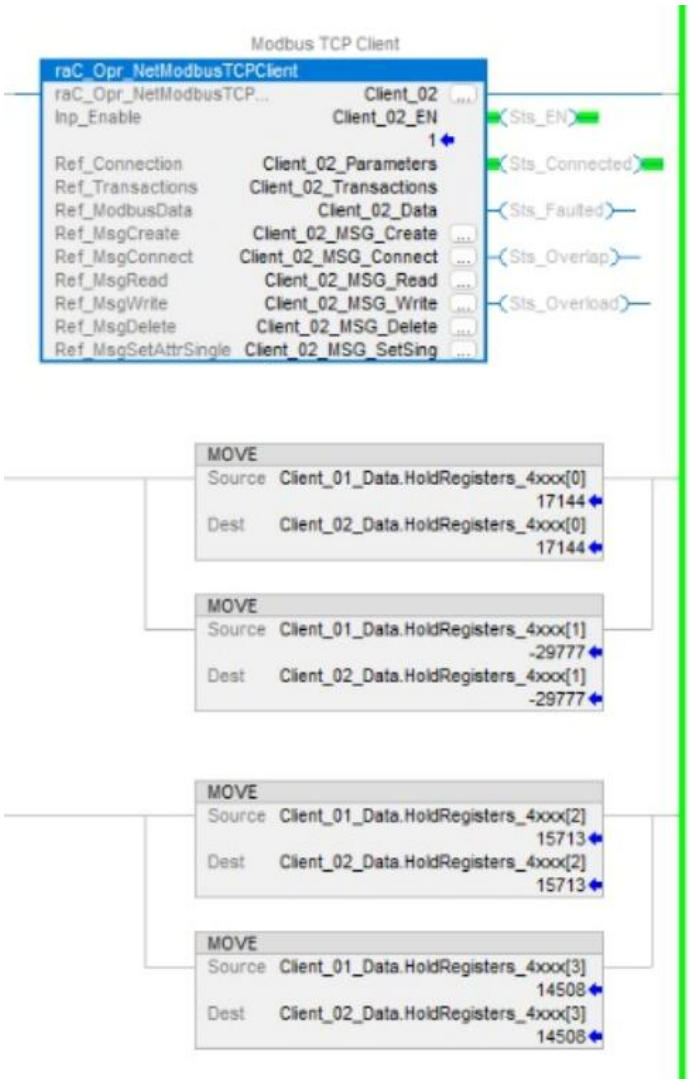


Figura 65. Lógica con bloques MOVE.

- Realizar la configuración de Client_02_Parameters y Client_02_Transactions siguiendo la comunicación con el ControlLogix. Se debe considerar el uso de la función 16 (Write Multiple Registers) en *Transaction Type* debido a que se pretende escribir el contenido binario de parámetros de tipo REAL.

- Considerar también que el *StationID* para el ControlLogix es el 0 (valor por default en controladores logix5000), de tal manera que no exista un conflicto con el Sentron.

◢ Client_02_Parameters	{...}
▷ Client_02_Parameters.LocalSlot	0
▷ Client_02_Parameters.LocalAddress	''
▷ Client_02_Parameters.DestAddress	'192.168.31.116'
▷ Client_02_Parameters.DestinationPort	502

◢ Client_02_Transactions	{...}
◢ Client_02_Transactions[0]	{...}
Client_02_Transactions[0].Enabled	1
▷ Client_02_Transactions[0].PollInterval	1000
▷ Client_02_Transactions[0].TransType	16
▷ Client_02_Transactions[0].StationID	0
▷ Client_02_Transactions[0].BeginAddress	0
▷ Client_02_Transactions[0].Count	2
▷ Client_02_Transactions[0].LocalAddress	0
Client_02_Transactions[0].TransComplete	1
▷ Client_02_Transactions[0].TransStat	0
▷ Client_02_Transactions[0].Diagnostic	{...}
◢ Client_02_Transactions[1]	{...}
Client_02_Transactions[1].Enabled	1
▷ Client_02_Transactions[1].PollInterval	1000
▷ Client_02_Transactions[1].TransType	16
▷ Client_02_Transactions[1].StationID	0
▷ Client_02_Transactions[1].BeginAddress	2
▷ Client_02_Transactions[1].Count	2
▷ Client_02_Transactions[1].LocalAddress	2
Client_02_Transactions[1].TransComplete	0
▷ Client_02_Transactions[1].TransStat	1

Figura 66. Client 02 Parameters y Transactions.

Nota: Se puede evitar el uso de dos transacciones. Con una sola transacción, se logra escribir ambos parámetros colocando en *Count* el valor de 4.

- A continuación, la lógica de la MainRoutine.

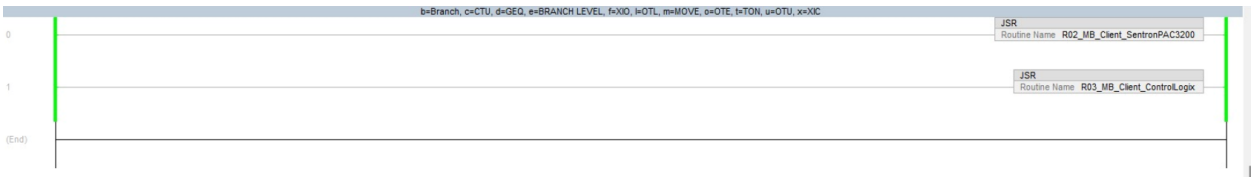


Figura 67. MainRoutine.

- Crear un nuevo proyecto en Studio 5000 para configurar el Modbus Server, seleccionar 1756-L73 ControlLogix.
- Seguir los pasos de la Configuración de un ControlLogix como Modbus Server.
- Verificar la configuración de Server_01_Parameters.

Server_01_Parameters	{...}
Server_01_Parameters.LocalSlot	6
Server_01_Parameters.LocalAddress	''
Server_01_Parameters.LocalPort	502
Server_01_Parameters.InactivityTimeout	60

Figura 68. Server 01 Parameters.

- A continuación, la lógica de la MainRoutine.

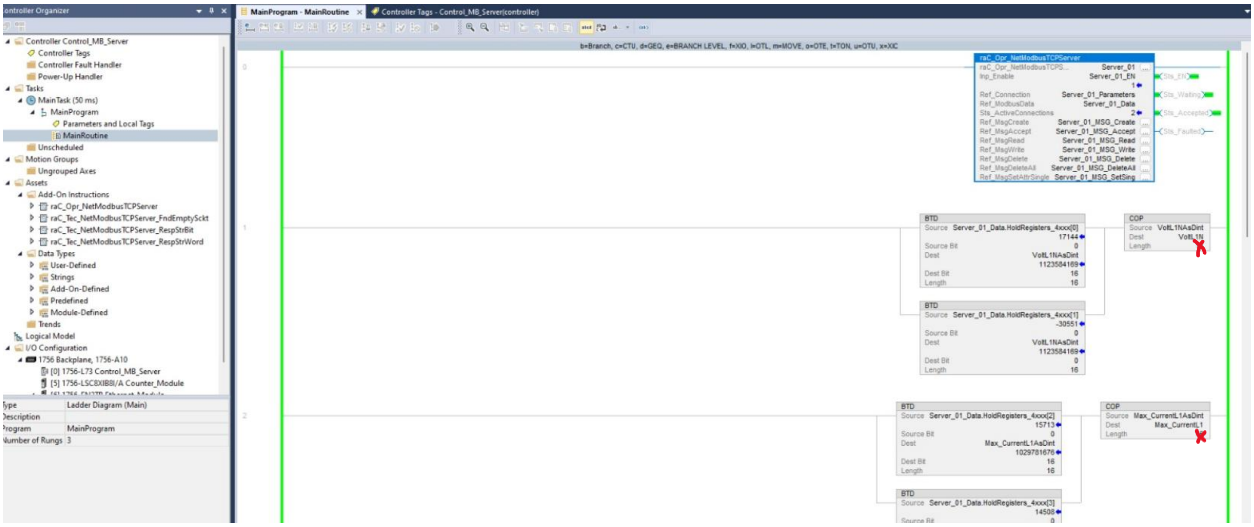


Figura 69. MainRoutine MB_Server_ControlLogix.

- Se implementa la misma lógica con la instrucción BTD y la instrucción COP.

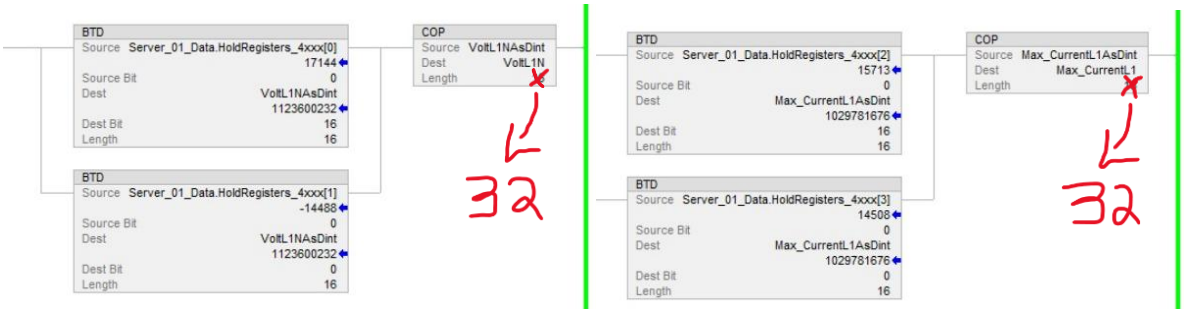


Figura 70. Lógica de Concatenación y Conversión.

- Dirigirse a Controller Tags, luego a Monitor Tags para observar el estado de los Holding Registers del Server. Debería aparecerle algo similar a esto:

Server_01_Data	{...}
Server_01_Data.Coils_0xxx	{...}
Server_01_Data.DiscInputs_1xxx	{...}
Server_01_Data.InpRegisters_3xxx	{...}
Server_01_Data.HoldRegisters_4xxx	{...}
Server_01_Data.HoldRegisters_4xxx[0]	17144
Server_01_Data.HoldRegisters_4xxx[1]	31276
Server_01_Data.HoldRegisters_4xxx[2]	15713
Server_01_Data.HoldRegisters_4xxx[3]	14508
Server_01_Data.HoldRegisters_4xxx[4]	0
Server_01_Data.HoldRegisters_4xxx[5]	0

Figura 71. Escritura correcta en el Server.

- Iniciar el Servidor de Node-RED desde la terminal de Node.js.
- Crear el siguiente flujo para la lectura de los Holding Registers del ControlLogix.

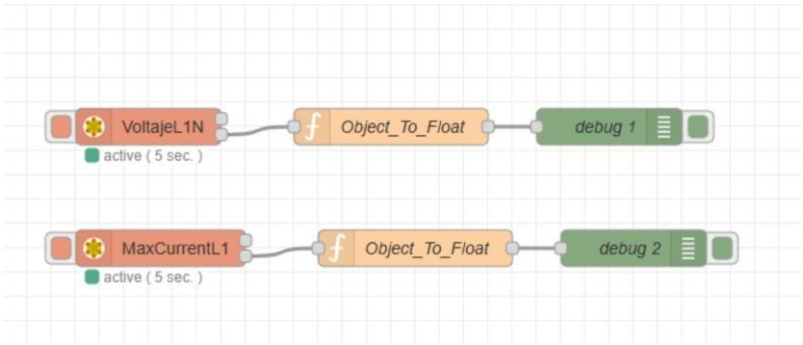


Figura 72. Flujo de Lectura.

- Para los nodos Modbus-Read crear el nodo de configuración de Modbus-Client según como se explicó previamente.

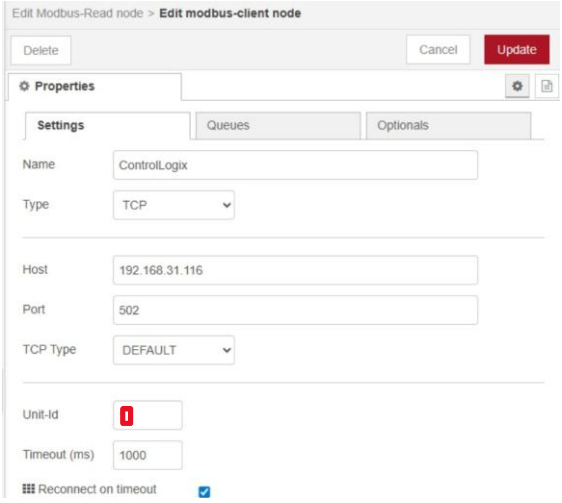


Figura 73. Nodo de Configuración Modbus-Client.

- Realizar la configuración de los nodos Modbus-Read respectivamente.

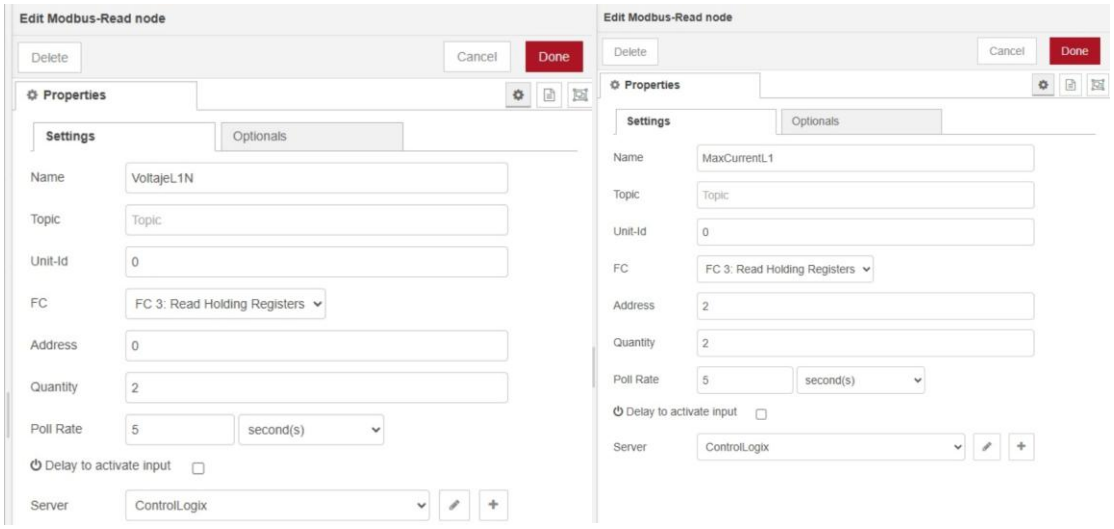


Figura 74. Nodos Modbus-Read.

- Realizar la configuración de los nodos Function. La razón del por qué se utiliza una función es por el formato recibido de los datos, recordando que obtendremos un arreglo de dos valores enteros. También se puede obtener del nodo Modbus-Read un dato tipo objeto o estructura formado por un arreglo de dos valores enteros y un arreglo llamado *buffer* de 4 posiciones con los octetos correspondientes en hexadecimal.

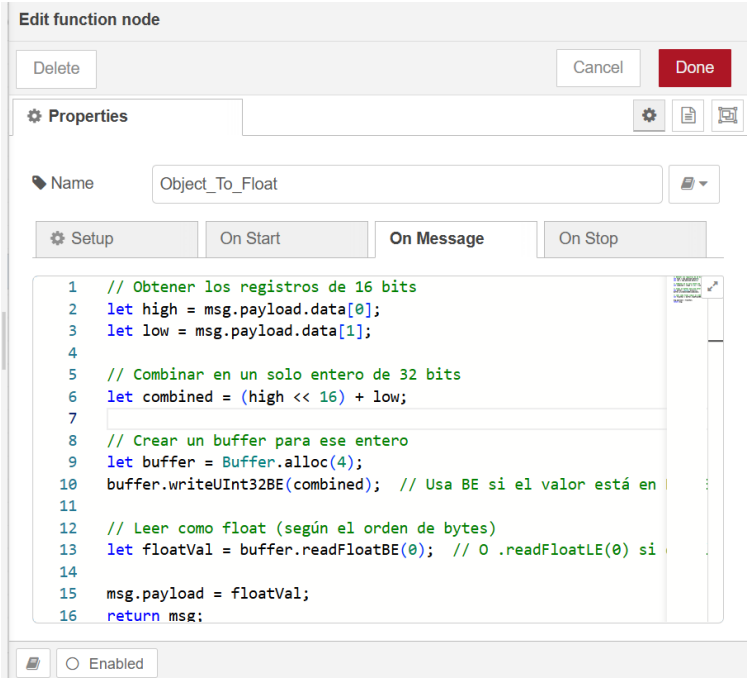


Figura 75. Función Object_To_Float.

- A continuación, el código en Javascript:

```
// Obtener los registros de 16 bits
   let high = msg.payload.data[0];
   let low = msg.payload.data[1];

   // Combinar en un solo entero de 32 bits
   let combined = (high << 16) + low;

   // Crear un buffer para ese entero
   let buffer = Buffer.alloc(4);
   buffer.writeUInt32BE(combined);  // Usar BE si el valor está en Big Endian

   // Leer como float (según el orden de bytes)
   let floatVal = buffer.readFloatBE(0);  // O .readFloatLE(0) si es Little
   Endian

   msg.payload = floatVal;
   return msg;
```

- Verificar en la sección de Debug que se esté leyendo correctamente los parámetros en formato REAL.

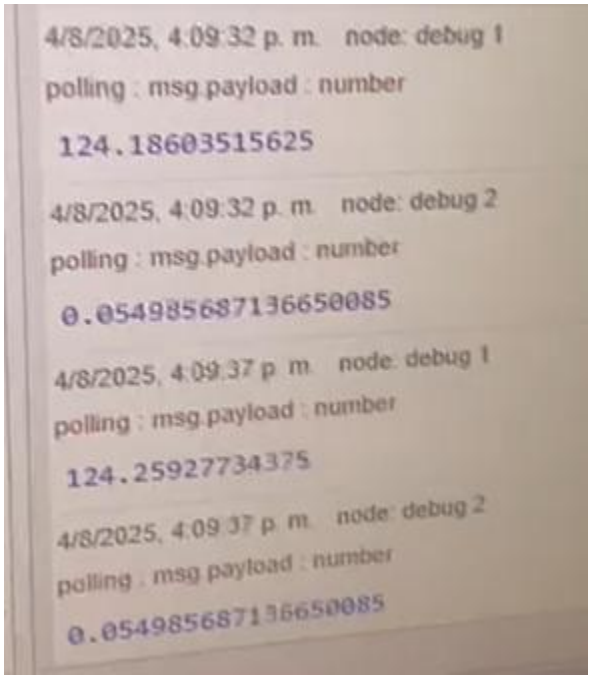


Figura 76. Lectura correcta de los parámetros.

- Finalmente… usted ha logrado implementar la arquitectura propuesta.

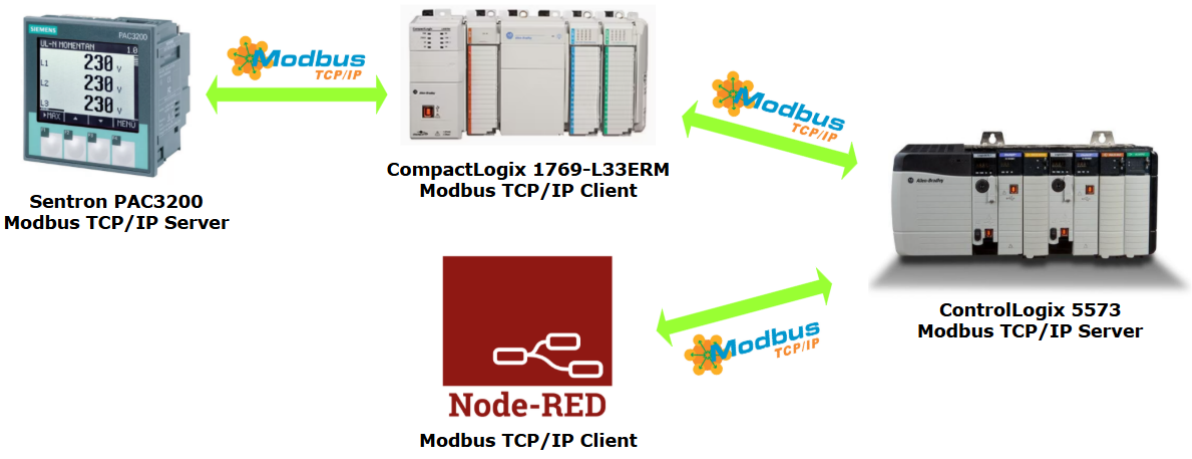


Figura 77. Arquitectura de Comunicación Modbus TCP/IP.