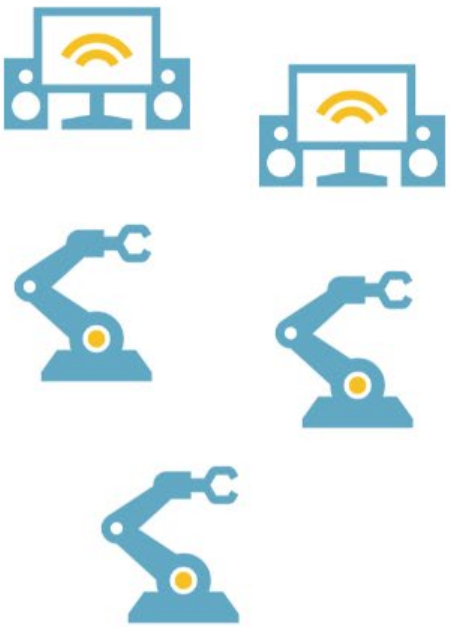
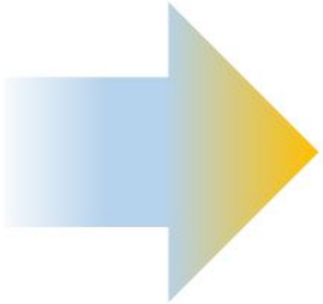
Introducción a MODBUS

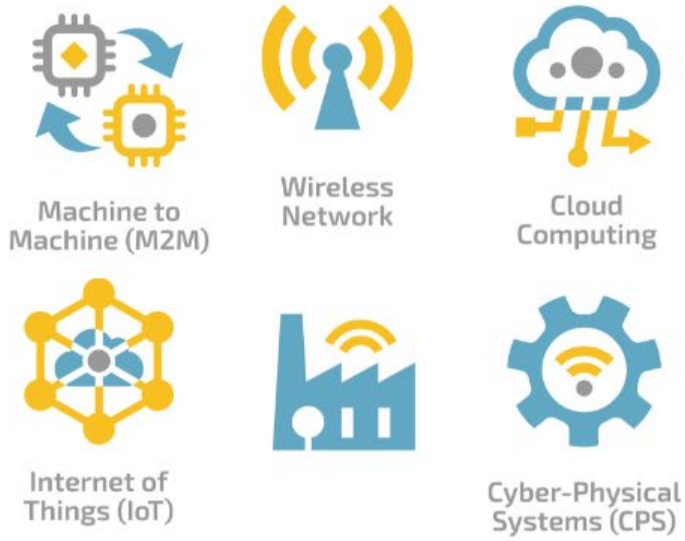
Almudena Díaz Zayas

Contexto



Industria 3.0



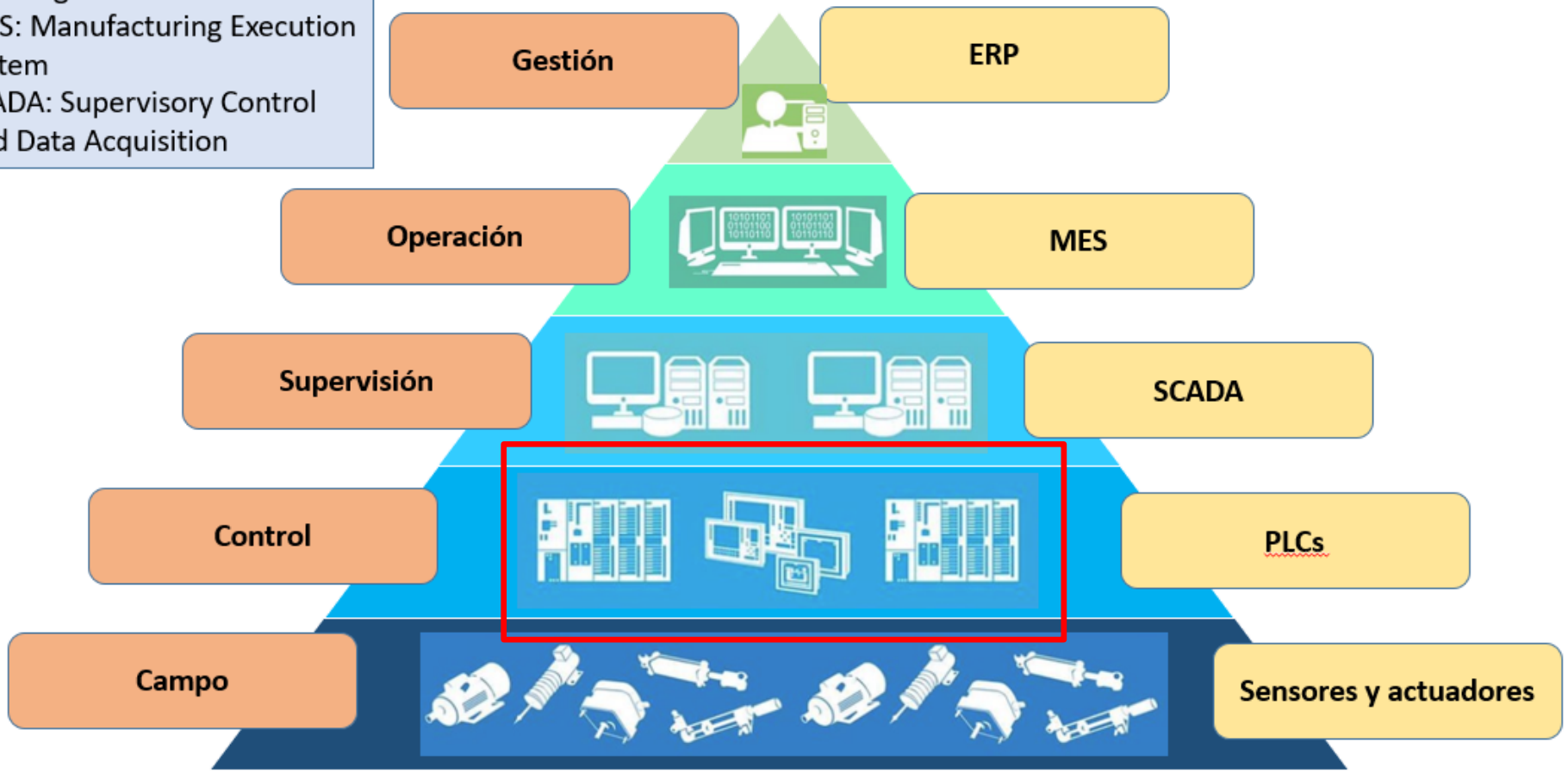


Industria 4.0

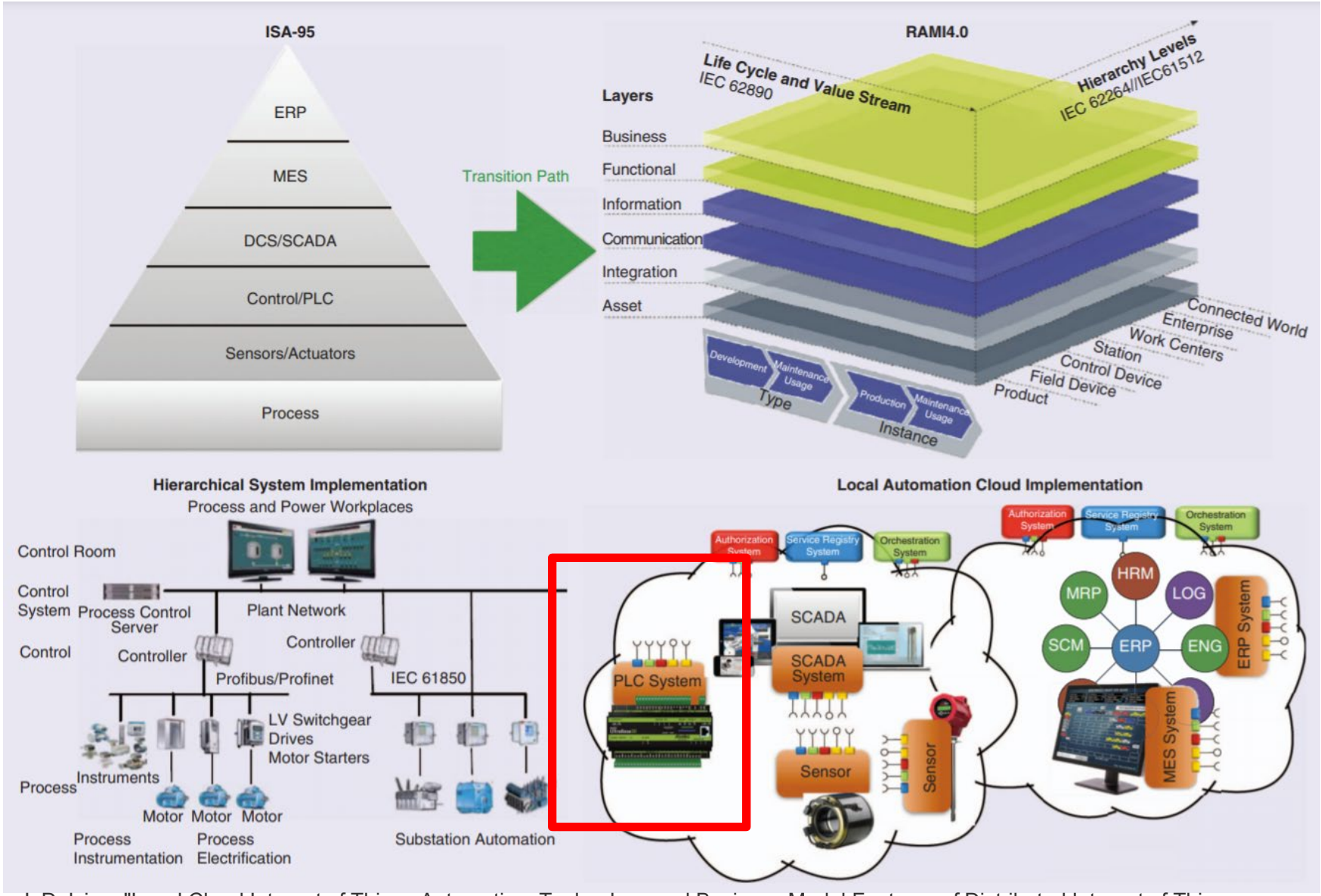
Pirámide de automatización (CIM – Computer integrated Manufacturing)

INDUSTRIA 3.0

ERP: Enterprise Resource Planning
MES: Manufacturing Execution System
SCADA: Supervisory Control And Data Acquisition

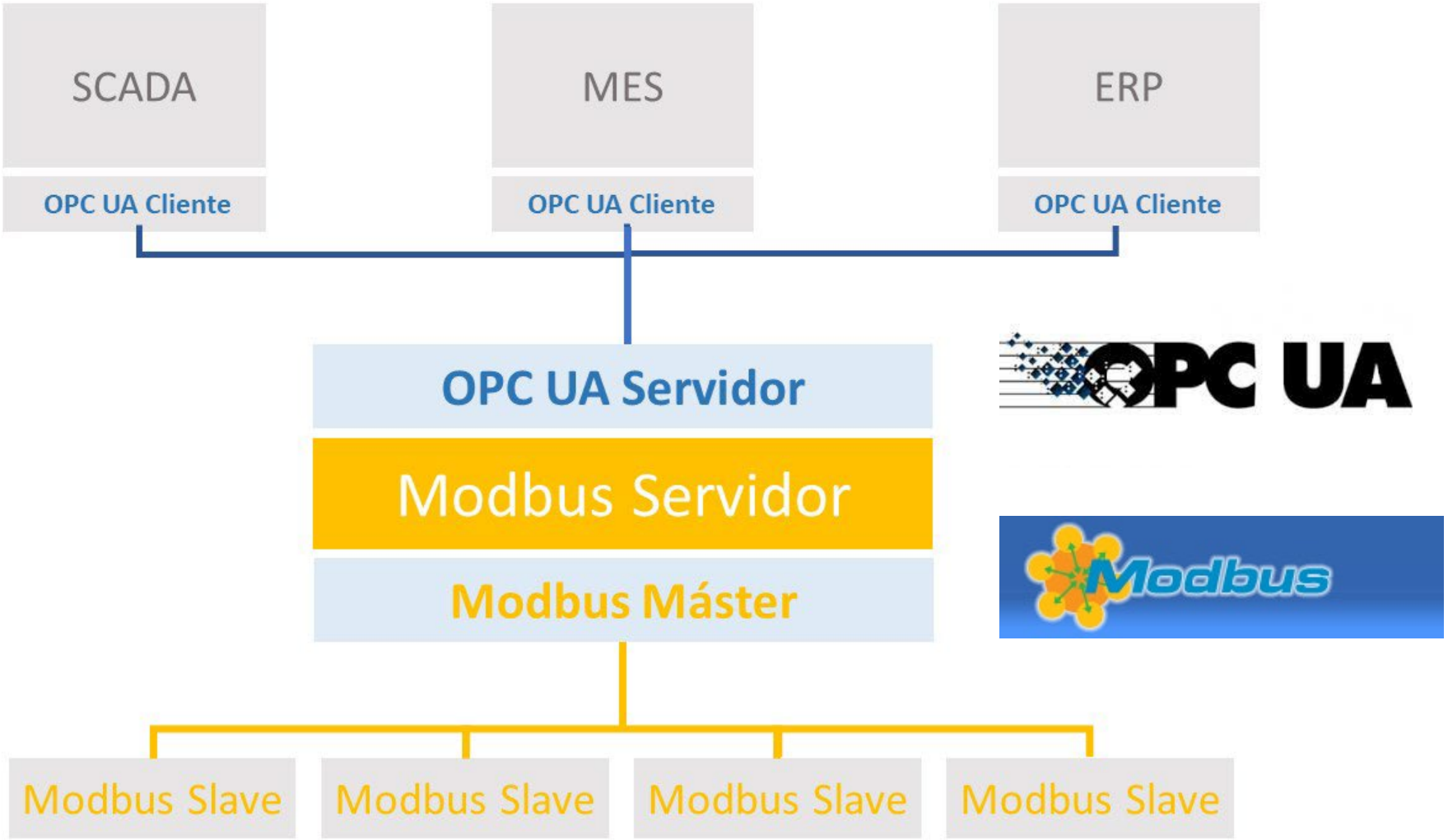


Transición Industria 4.0



J. Delsing, "Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions," in *IEEE Industrial Electronics Magazine*, vol. 11, no. 4, pp. 8-21, Dec. 2017, doi: 10.1109/MIE.2017.2759342.

Modbus y OPC UA



Introducción a Modbus

Lógica cableada vs Lógica programada

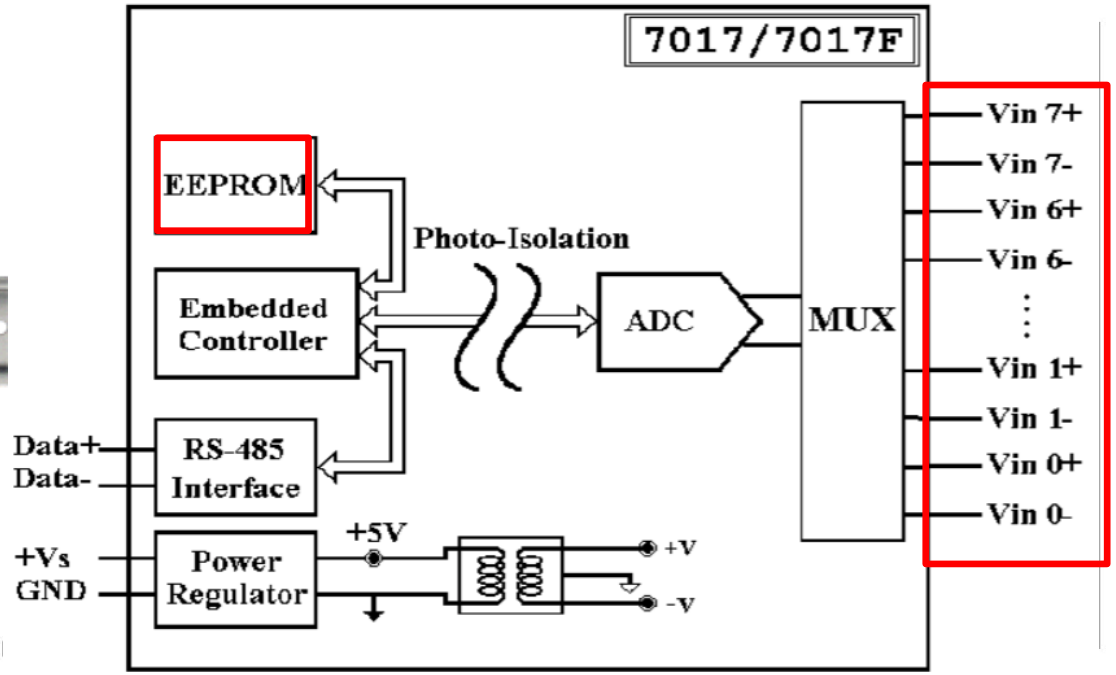
Sistemas de control distribuidos para el control de plantas industriales.

- **Lógica cableada**: Los sensores y los actuadores se conectan desde los puntos de instalación distribuidos a lo largo de toda la planta a la sala de control central y los automatismos se realizan a través de relés, pulsadores, etc. → Alto coste de despliegue y mantenimiento. Escalabilidad limitada.

- **Lógica programada:** En 1968 se desarrolló el primer CONTROLADOR LÓGICO PROGRAMABLE (PLC), cuyo nombre era **Modular Digital CONtroller** (MODICON), con la intención de sustituir a la lógica cableada.

Definición PL según norma IEC 61131

Un **controlador lógico programable** es una máquina electrónica programable diseñada para ser utilizada en un **entorno industrial (hostil), que utiliza una memoria programable** para el almacenamiento interno de instrucciones orientadas al usuario, para implantar soluciones especificas tales como funciones lógicas, secuenciales, temporizaciones, recuentos y funciones aritméticas, con el fin de **controlar mediante entradas y salidas, digitales y analógicas diversos tipos de máquinas o procesos.**



Protocolo Modbus

A continuación surge la necesidad de comunicar los equipos Modicon. En 1979 surge el protocolo **Modbus (Modicon Field Bus)**. Usando el protocolo Modbus se pueden interconectar todos los PLCs con una **misma red** de comunicaciones.

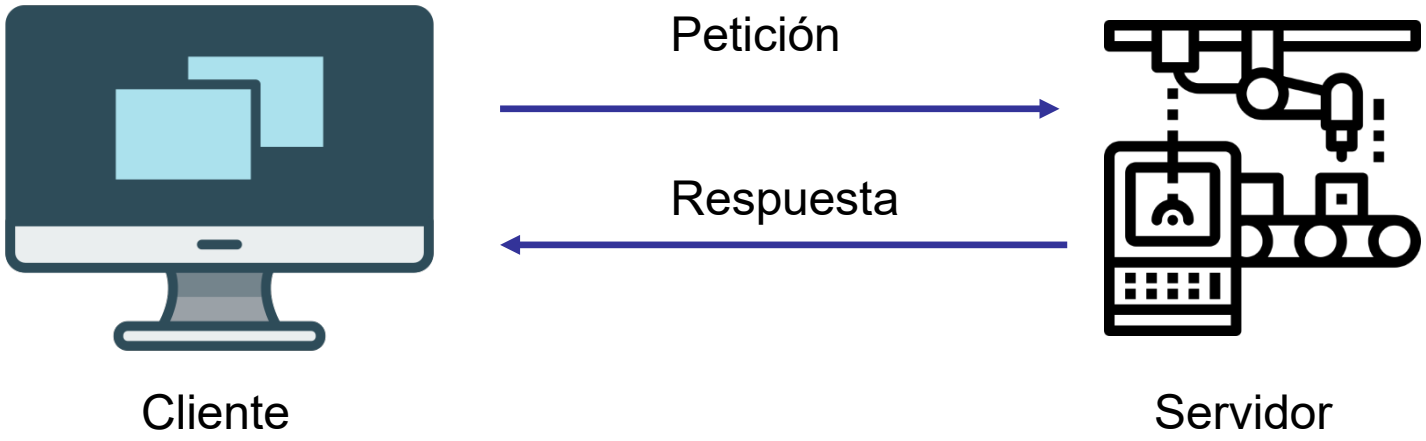
En 2004 Modbus pasó a ser un **estándar abierto** y **gratuito**.

Muchas empresas adoptaron este protocolo alcanzando una alta inserción y convirtiéndose en el **protocolo de comunicaciones de facto** en la **industria de control de procesos**.

Protocolo Modbus

Es un **protocol** de **nivel de aplicación** que proporciona una comunicación **cliente/servidor** entre dispositivos conectados a distintos tipos de buses o redes.

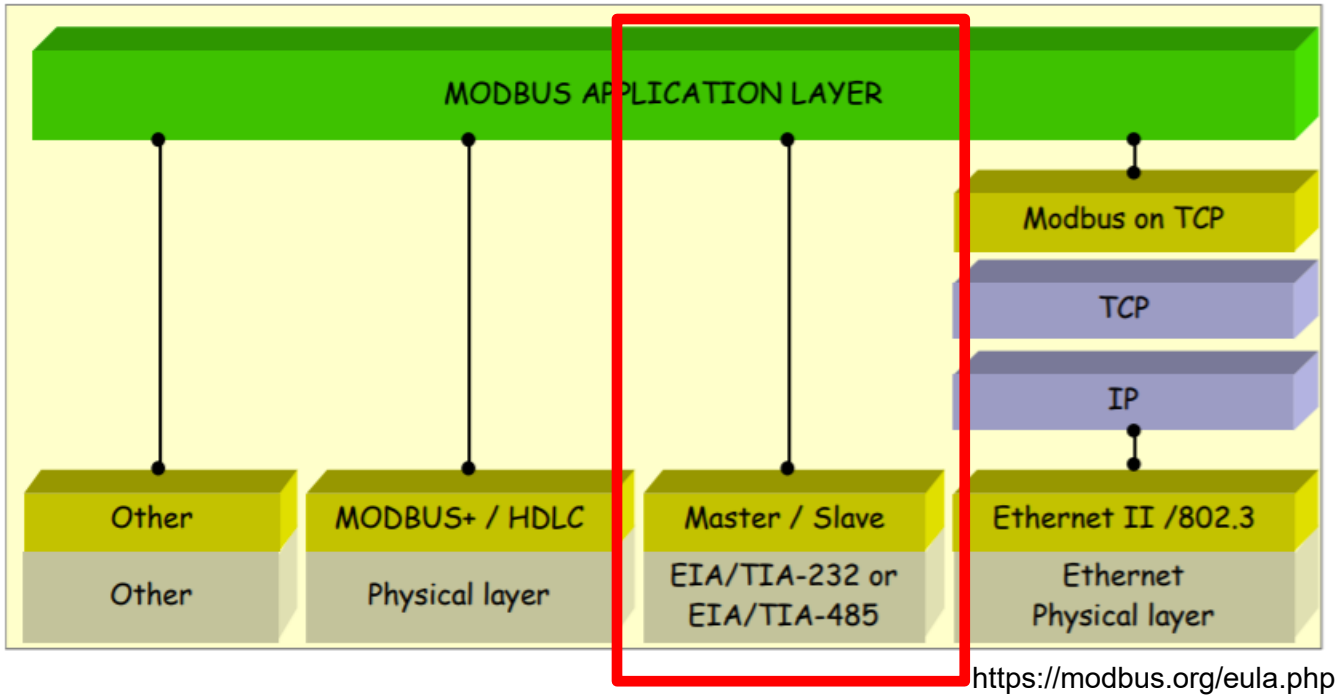
Se basa en un mecanismo de **solicitud/respuesta** para ofrecer servicios especificados por los denominados códigos de función. Los códigos de función Modbus son elementos que se insertan en la unidad de datos del protocolo.



Modbus: versiones más extendidas

Es un protocol de nivel de aplicación que proporciona una comunicación cliente/servidor **entre dispositivos conectados a distintos tipos de buses o redes:**

- Transmisión asíncrona serie + Maestro/esclavo (acceso al medio)
  - **Modbus RTU (Remote Terminal Unit)**
  - Modbus ASCII
- Ethernet → **Modbus TCP**
- HDLC → Modbus Plus **(Modbus+)**



https://modbus.org/eula.php

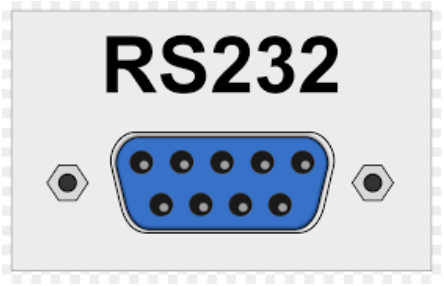
Interfaces eléctricos: RS232 Y RS485

RS232 fue diseñado principalmente para conectar módems a computadoras pero se extendió rápidamente a una gran cantidad de dispositivos y en la industria de automatización. Sin embargo este protocolo presenta varios inconvenientes:

- Estándar **punto a punto**
- Distancia de **15 metros**
- No es resistente frente a las fuentes de ruido eléctrico

Se creó el estándar **RS485** para abordar muchos de estos problemas:

- **RS485** es un protocolo **punto a multipunto.**
- Se pueden conectar hasta 32 dispositivos sin repetidor.
- El alcance es de **1200 metros.**
- Es resistente al ruido eléctrico



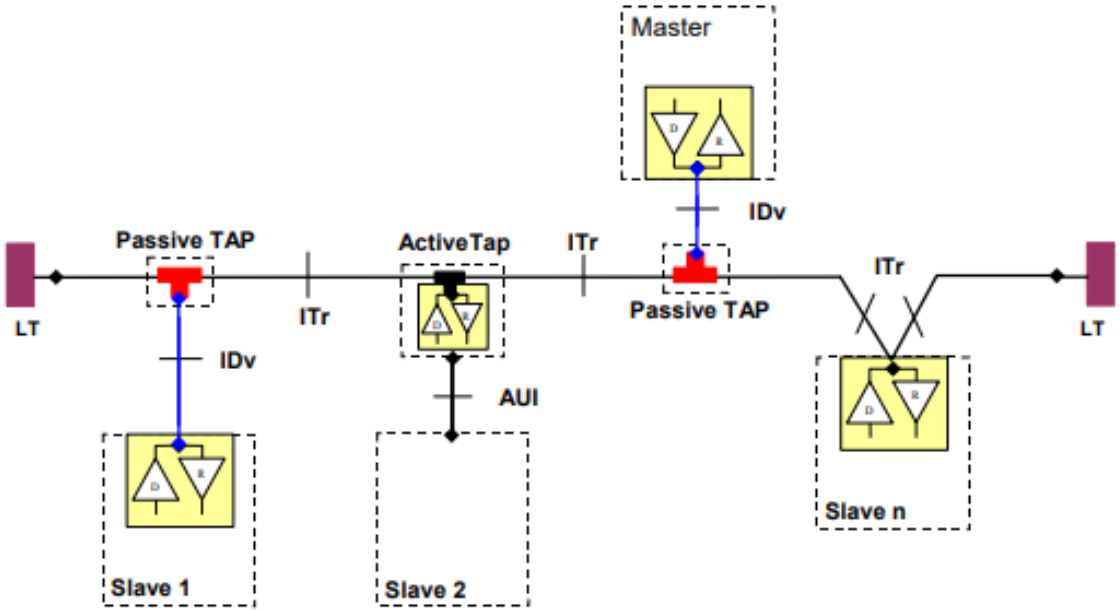
Capa física

Una solución Modbus debe implementar una interfaz eléctrica de acuerdo al estándar **RS485** (se debe soportar la versión de **2 cables** y opcionalmente la de 4 cables).

Adicionalmente puede implementar el estándar **RS232**.

Tasas de señalización de datos

- **Baude rate (bps):** 1200, 2400, 4800, **9600, 19200**, 38400



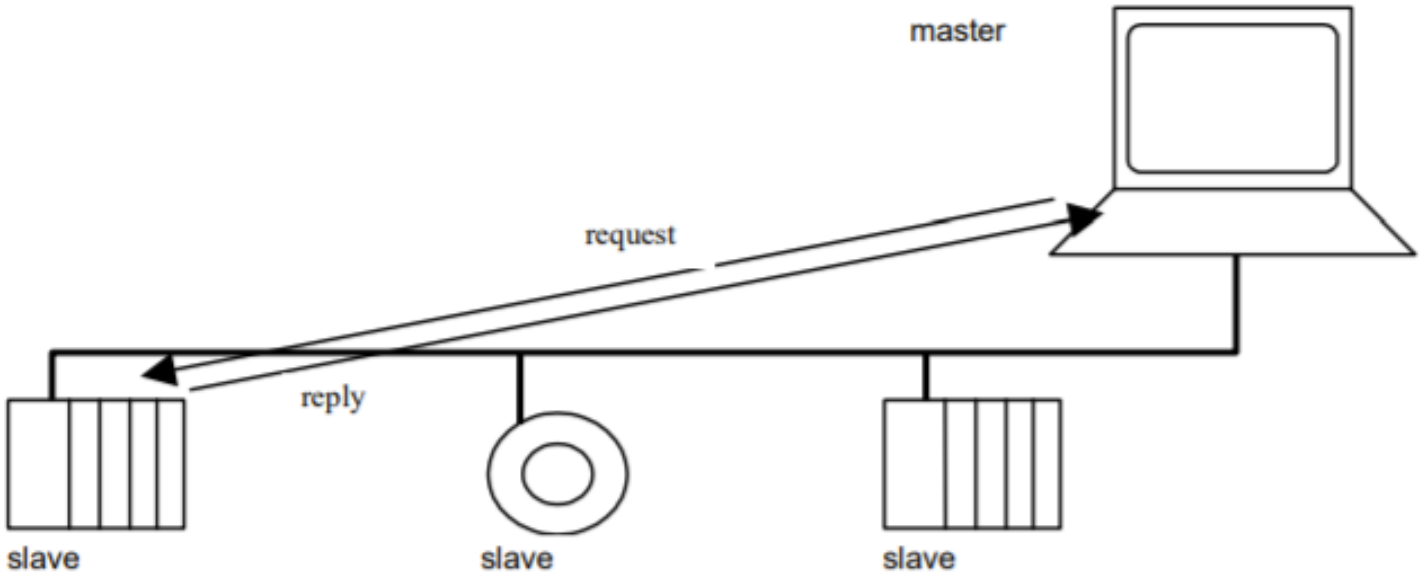
https://modbus.org/eula.php

Acceso al medio Modbus serie

El mecanismo de acceso al medio del protocolo Modbus para buses serie es el **maestro-esclavo**.

Sólo un maestro (al mismo tiempo) **puedes estar conectado al bus**, y uno o varios nodos esclavos (247 como máximo) pueden estar conectados al mismo bus serie.

La comunicación es siempre **iniciada por el maestro**. Los nodos esclavos nunca transmitirán datos sin recibir una solicitud del nodo maestro. Los nodos esclavos nunca se comunicarán entre sí. El maestro inicia sólo una transacción Modbus al mismo tiempo.



https://modbus.org/eula.php

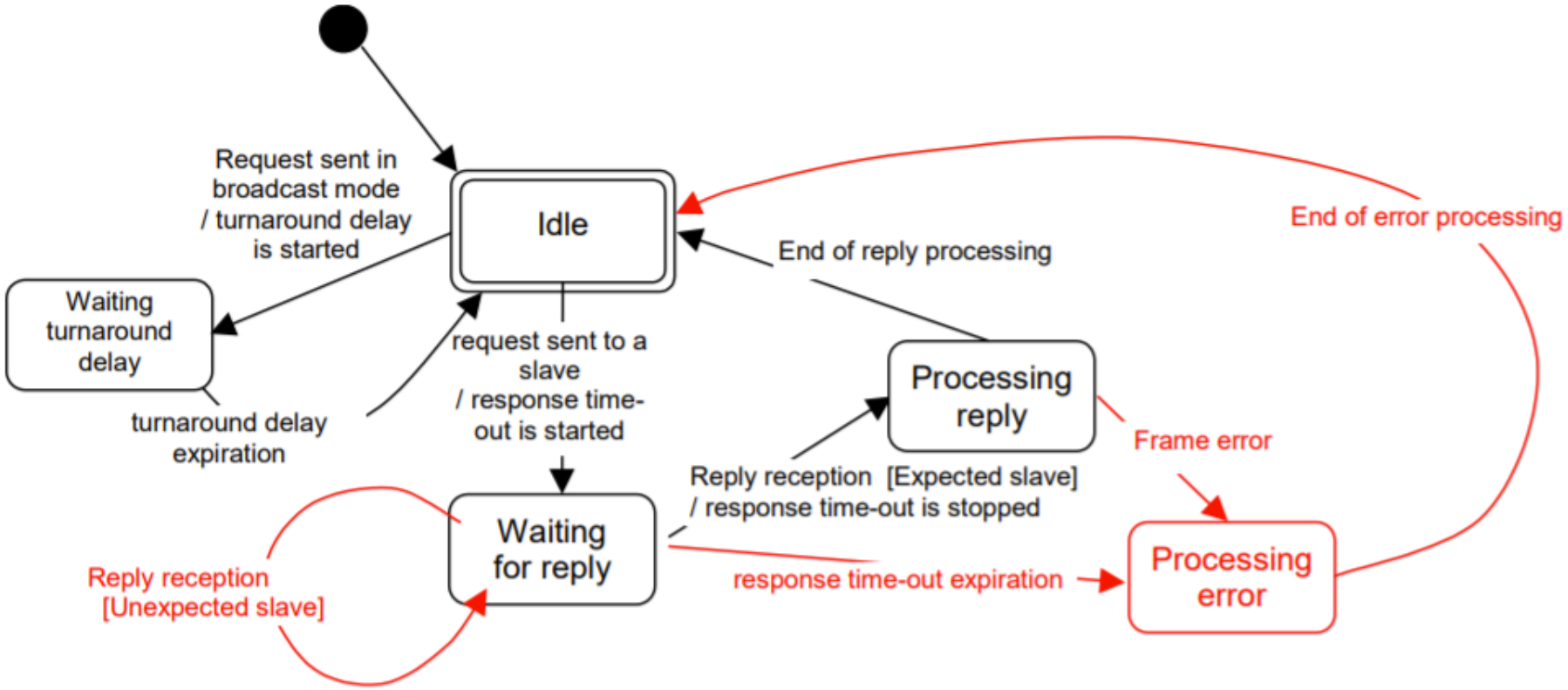
Acceso al medio Modbus serie: Direccionamiento

- El espacio de direcciones está limitado **a 256 direcciones distintas**
- La dirección **0** está reservada como la dirección de **broadcast**.
- El maestro no tiene una dirección específica, **sólo los nodos esclavos deben tener una dirección.**
- La **dirección única debe ser única.**

0	1-247	248-255
Broadcast	Esclavos	Reservadas

Acceso al medio Modbus serie

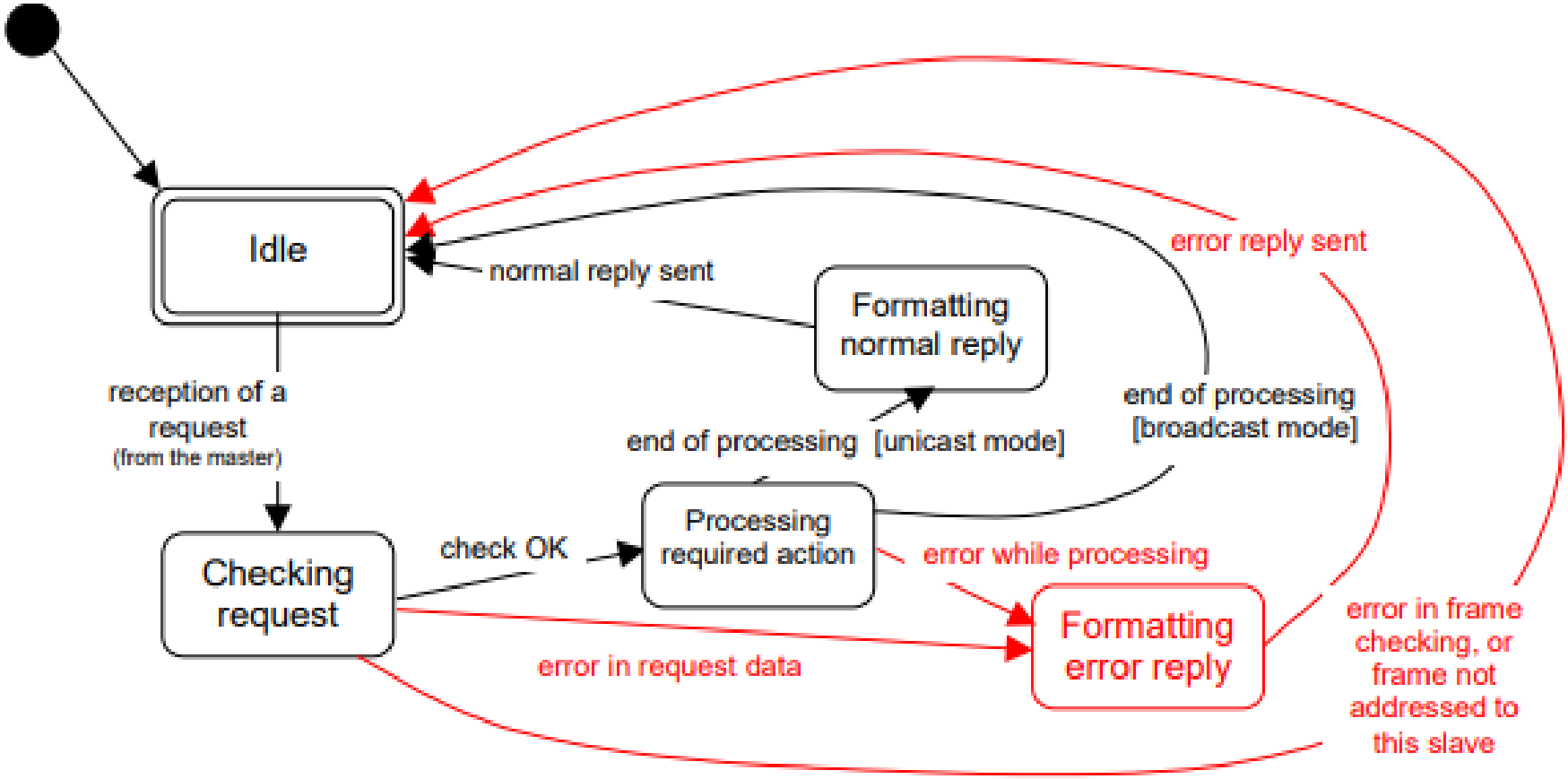
Diagrama de estado del **maestro**



https://modbus.org/eula.php

Acceso al medio Modbus serie

Diagrama de estado del **esclavo**



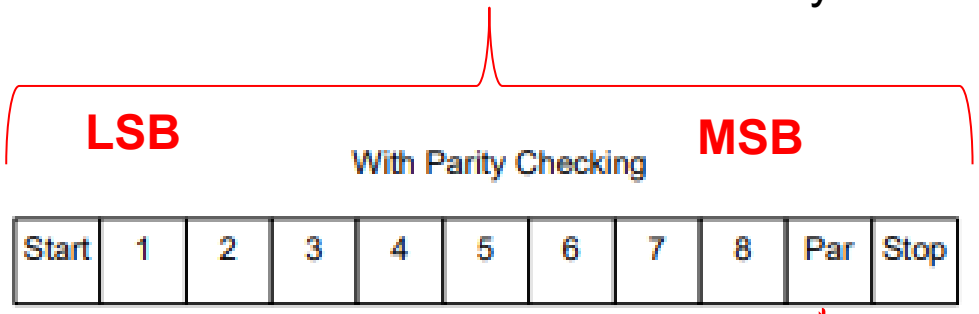
https://modbus.org/eula.php

Modo de transmisión Modbus RTU serie

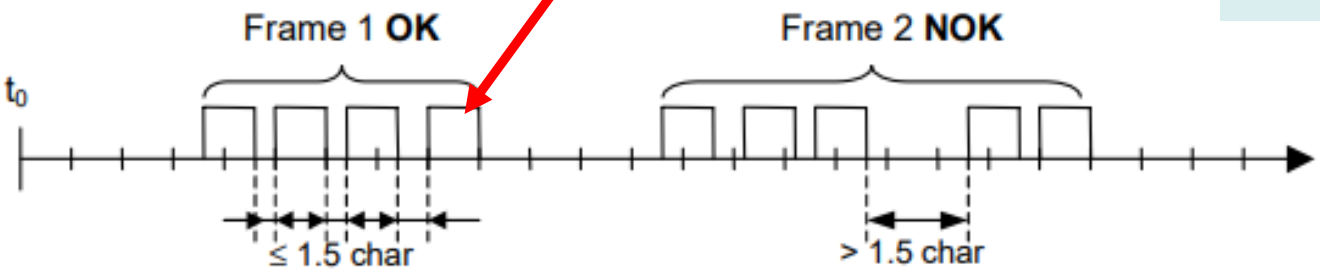
Secuencia continua de caracteres: Por cada **8 bits** de datos se le añade un bit de parada, un bit de inicio y un bit de paridad.

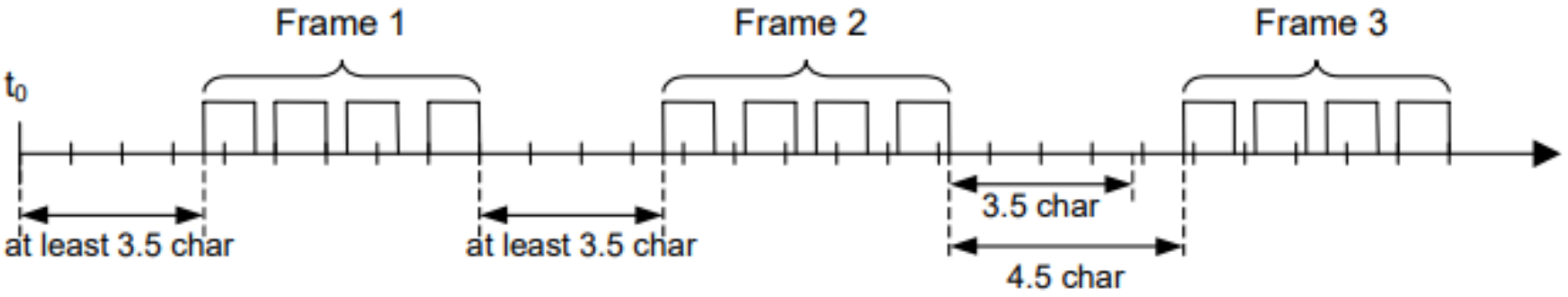
Transmisión de los caracteres/bytes

Sistema de codificación binario

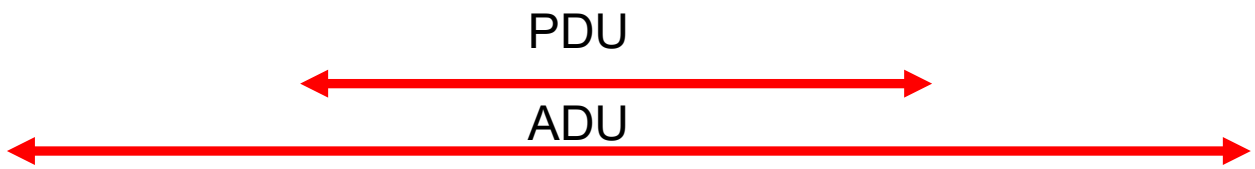


Paridad: Even
Odd,
No Parity checking





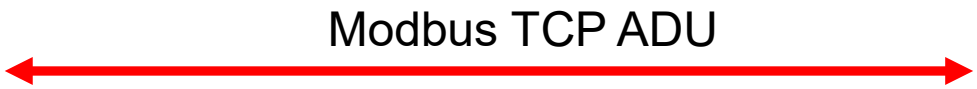
Nivel de aplicación Modbus



Address field	Function field	Data field	Error check field
1 byte	1 byte	Variable	2 bytes

El protocolo Modbus define una unidad de datos de protocolo **(PDU)** simple e **independiente de las capas de comunicación subyacentes**.

El mapeo del protocolo MODBUS en buses específicos o protocolos de red puede introducir algunos campos adicionales en la **unidad de datos de aplicación** (ADU).



MBAP Header	Function field	Data field
7 bytes	1 byte	Variable

Tamaño de PDU

El tamaño de la PDU está limitado por la restricción de tamaño heredada de la primera implementación Modbus para comunicación serie: **ADU = 256 bytes**.

PDU Modbus comunicación serie = 256 - Dirección del servidor (1 byte) - CRC (2 bytes) = ***253 bytes.***

TCP MODBUS ADU = 253 bytes + MBAP (7 bytes) = 260 bytes.

Modbus utiliza una representación **Big Endian** para las direcciones y los datos.

Max ADU = 256 bytes

Address field	Function field	Data field	Error check field
1 byte	1 byte	Variable	2 bytes

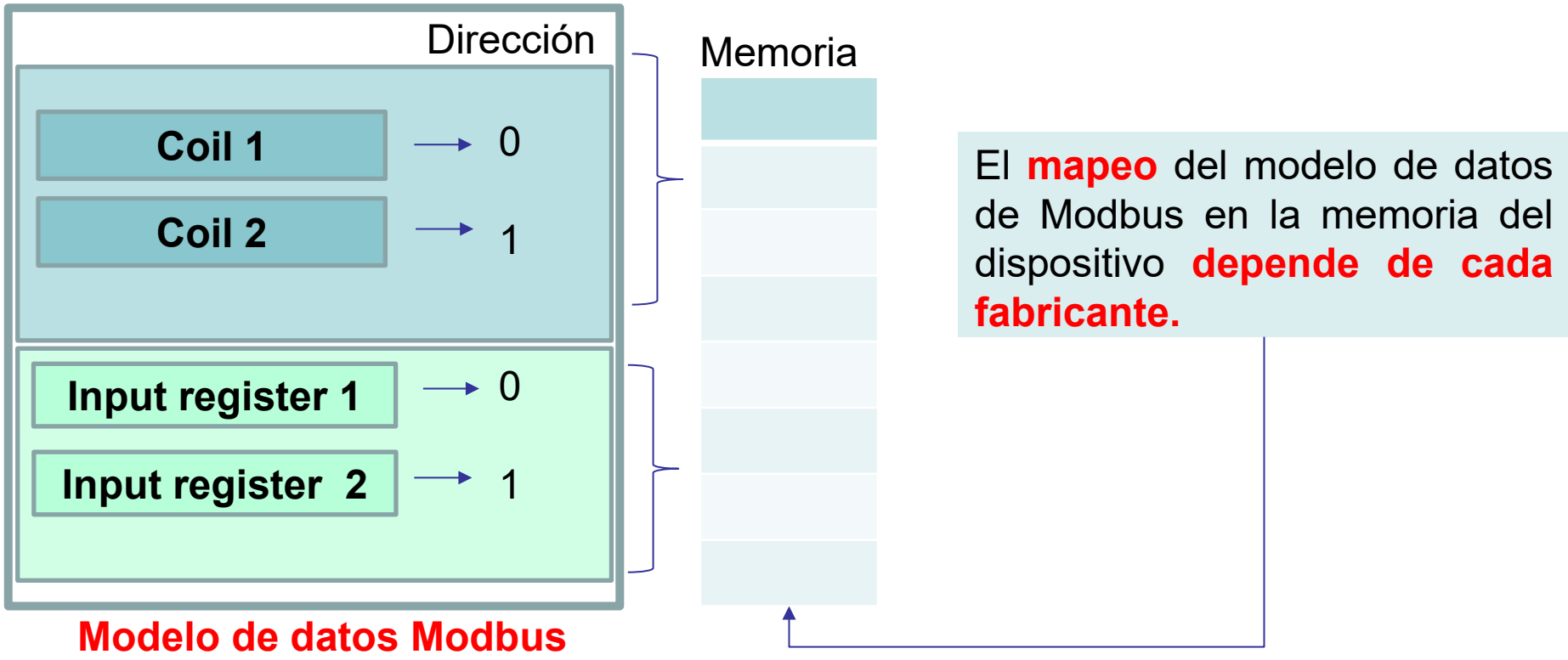
Modelo de datos Modbus

- Se definen **cuatro bloques de datos conceptuales**.
- No es necesario implementar todos los tipos de bloques.

Bloques de datos	Tipo de datos	Tipo de acceso
Coils	1 bit	Lectura/Escritura
Discrete Input	1 bit	Sólo lectura
Input registers	16 bits	Sólo lectura
Holding registers	16 bits	Lectura/Escritura

Modelo de datos Modbus

- Cada bloque tiene un tamaño máximo de **65.536 elementos**.
- Dentro de cada bloque los datos tienen **direcciones** que van desde la 0 al **65535**.
- Cada elemento del banco está numerado de 1 hasta n.



Modelo de datos Modbus

Modelo de datosModbus

Existe una convención (no forma parte del estándar) que añade un **prefijo** al número del elemento que indica el tipo de bloque al que pertenece. Se debe expresar las referencias a los bloques de datos con 6 dígitos, siendo el primero el prefijo.

Bloque de datos	Prefijo
Coils	0
Discrete Inputs	1
Input registers	3
Holding registers	4

NÚMEROS DE REFERENCIA

Coils → 0:00001 – 0:65535

Discrete inputs → 1:00001 – 1:65535

Input Registers → 3:00001 – 3:65535

Holding registers → 4:00001-4:65535

Referencia 400001 → Holding register 1 → Dirección 0

Referencia 400000 → Holding register 1 → Dirección 0

Tipos de PDU

Se definen tres tipos de PDUs:

Request	Function code	Function Data
Response	Function code	Response Data
Exception Response	Error code	Exception Data

Request Modbus RTU

Address field	Function field	Data field	Error check field
1 byte	1 byte	Variable	2 bytes

REQUEST

Device address	←	ID del destinatario (1 Byte)
Function code	←	Indica la función a realizar (1 Byte)
Data	←	Indica el rango de direcciones (4 Bytes)
Error check	←	CRC (Cyclical Redundancy Check) para detectar errores en la transmisión (2 Bytes)

Formato de la trama

Address field	Function field	Data field	Error check field
1 byte	1 byte	Variable	2 bytes

RESPONSE

Device address	←	ID del dispositivo que envía la respuesta (1 Byte)
Function code	←	Devuelve el mismo código de función (1 Byte)
Data	←	Datos (4 Bytes) (8 Bytes) (16 Bytes) …
Error check	←	CRC (Cyclical Redundancy Check) para detectar errores en la transmisión (2 Bytes)

Formato de la trama

Ejemplo: Trama para pedir los datos almacenados en el registro 300003

REQUEST

Device address	←	[ ]
Function code	←	[ ]
Data	←	[ ] [ ] [ ] [ ]
Error check		

ID=3

Formato de la trama

Ejemplo: Trama para pedir los datos almacenados en el registro 300003

REQUEST

Device address	←	[03]
Function code	←	[ ]
Data	←	[ ] [ ] [ ] [ ]
Error check		

ID=3

Códigos de función

- Read coil status **(Function code 1)**
- Read input status **(Function code 2)**
- Read holding registers **(Function code 3)**
- Read Input Registers **(Function code 4)**
- Force Single Coil **(Function code 5)**
- Preset Single Register **(Funtion code 6)**

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2
		Internal Bits Or Physical coils	Read Coils	01		01	6.1
			Write Single Coil	05		05	6.5
			Write Multiple Coils	15		0F	6.11
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3
			Write Single Register	06		06	6.6
			Write Multiple Registers	16		10	6.12
			Read/Write Multiple Registers	23		17	6.17
			Mask Write Register	22		16	6.16
			Read FIFO queue	24		18	6.18
	File record access		Read File record	20		14	6.14
			Write File record	21		15	6.15
	Diagnostics		Read Exception status	07		07	6.7
			Diagnostic	08	00-18,20	08	6.8
			Get Com event counter	11		OB	6.9
			Get Com Event Log	12		0C	6.10
			Report Server ID	17		11	6.13
			Read device Identification	43	14	2B	6.21
	Other		Encapsulated Interface Transport	43	13,14	2B	6.19
			CANopen General Reference	43	13	2B	6.20

Formato de la trama

Ejemplo: Trama para consultar los datos almacenados en el registro 300003.

ID=3

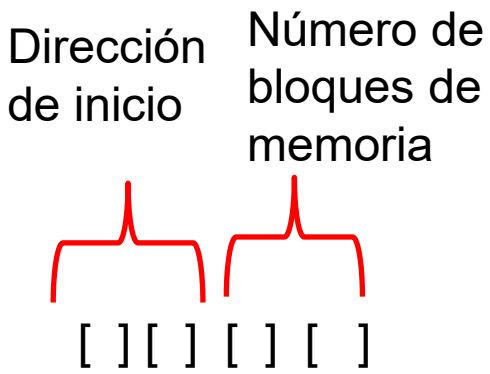
REQUEST

Device address	←	[03]
Function code	←	[04]
Data	←	[ ] [ ] [ ] [ ]
Error check		

Formato de la trama

Ejemplo: Trama para consultar los datos almacenados en el registro 300003

REQUEST		
Device address	←	[03]
Function code	←	[04]
Data	←	
Error check		

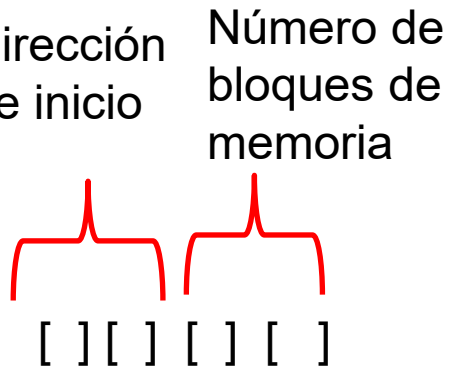


Formato de la trama

Ejemplo: Trama para consultar los datos almacenados en el registro 300003

QUERY

Device address	←	[03]
Function code	←	[04]
Data	←	
Error check		



Dirección de inicio: 300003 – 300001 (límite inferior del rango de registros de entrada) → [00] [02]

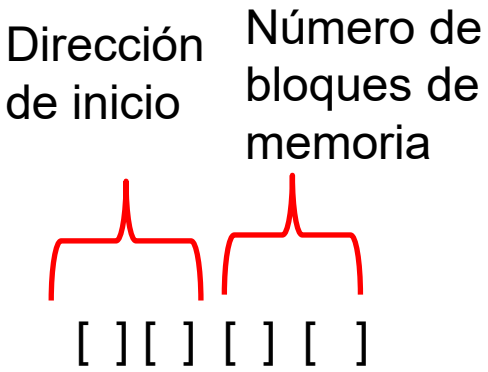
Número de bloques de memoria: [00] [01]

Formato de la trama

Ejemplo: Trama para consultar los datos almacenados en el registro 300003.

REQUEST

Device address	←	[03]
Function code	←	[04]
Data	←	
Error check		



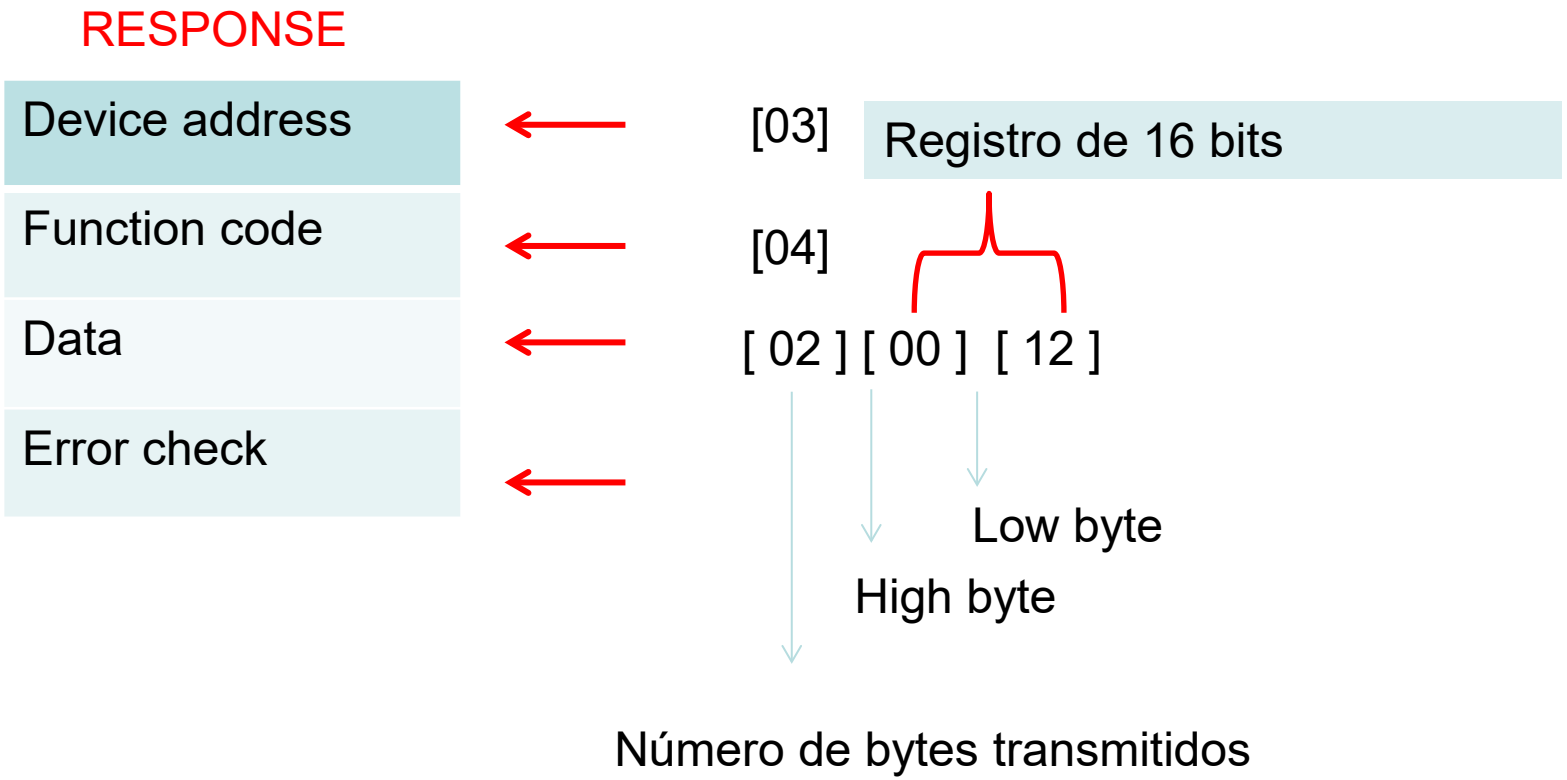
Dirección de inicio: 300003 – 300001 (límite inferior del rango de registros de entrada) → [00] [02]

Número de bloques de memoria: [00] [01]

[03] [04] [00] [02] [00] [01]

Formato de la trama

Ejemplo: Respuesta con los datos almacenado en el registro 300003.



Formato de la trama

Ejemplo: Trama para leer los datos almacenados en los registros 100001 y 100002

REQUEST

Device address	←	[03]
Function code	←	[ ]
Data	←	
Error check		

ID=3

Request commands

- Read coil status **(Function code 1)**
- Read input status **(Function code 2)**
- Read holding registers **(Function code 3)**
- Read Input Registers **(Function code 4)**
- Force Single Coil **(Function code 5)**
- Preset Single Register **(Funtion code 6)**

Formato de la trama

Ejemplo: Trama para leer los datos almacenados en los registros 100002 y 100003

REQUEST

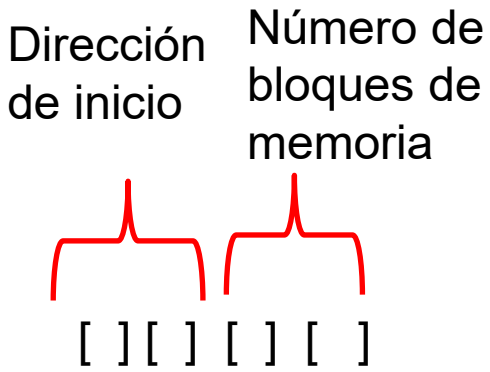
Device address	←	[03]
Function code	←	[02]
Data	←	
Error check		

Formato de la trama

Ejemplo: Trama para leer los datos almacenados en los registros 100002 y 100003

REQUEST

Device address	←	[03]
Function code	←	[02]
Data	←	
Error check		

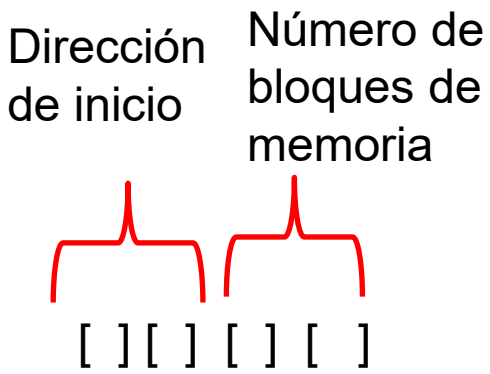


Formato de la trama

Ejemplo: Trama para leer los datos almacenados en los registros 100002 y 100003

REQUEST

Device address	←	[03]
Function code	←	[02]
Data	←	
Error check		



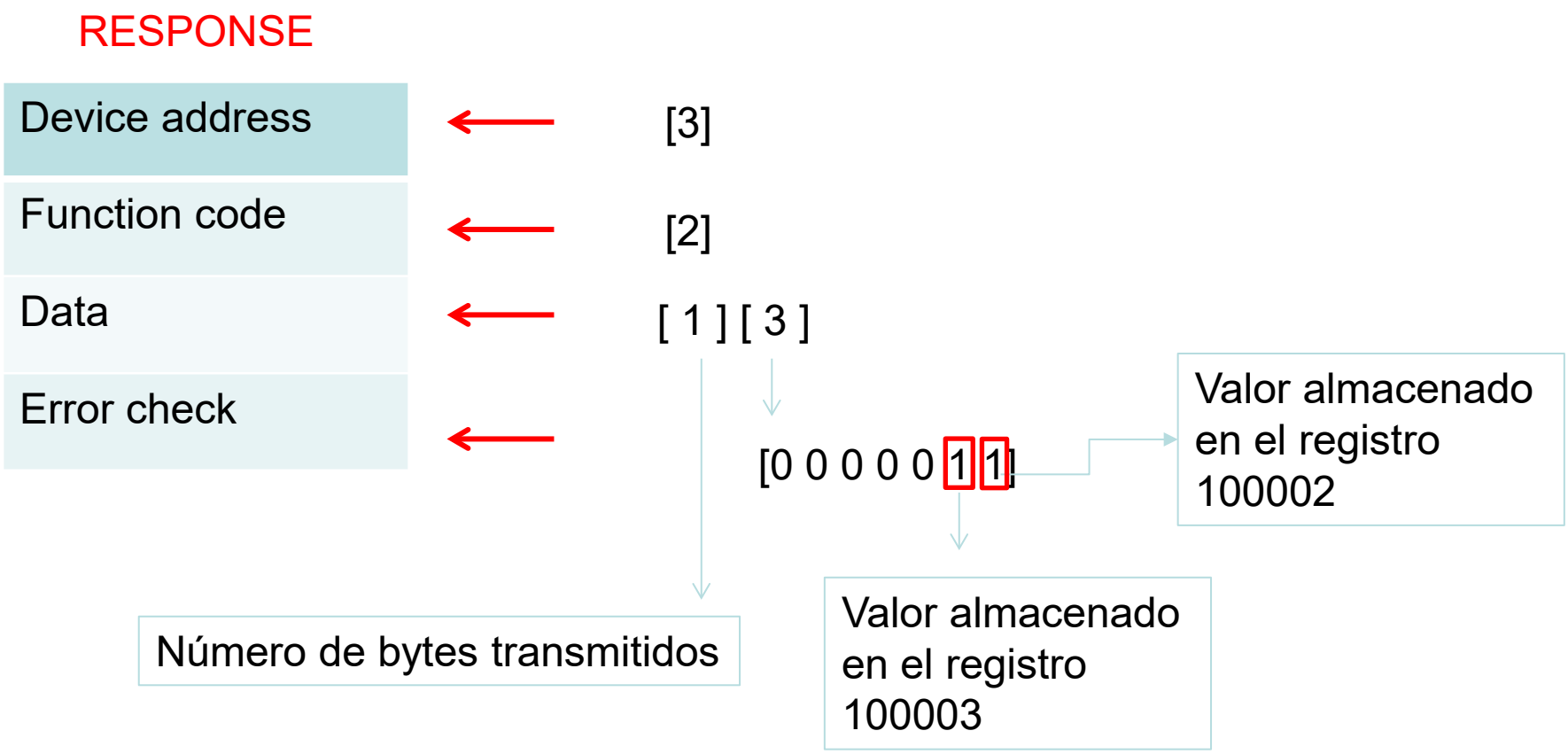
Dirección de inicio: 100002-100001 (límite inferior del rango de registros de entrada) → [00] [01]

Número de bloques de memoria: [00] [02]

[03] [02] [00] [01] [00] [02]

Formato de la trama

Respuesta con los datos almacenados en los registros 100002 y 100003.



Formato de la trama

Trama de escritura en registro discreto de salida (COIL) mapeado en el bloque de memoria 000003

ID=7

REQUEST

Device address	←	[7]
Function code	←	[5]
Data	←	[00] [2] [FF] [00]
Error check		

Activar 0xFF 00
Desactivar 0x00 00.

Formato de la trama

Respuesta de escritura en registro discreto de salida (COIL)

ID=7

RESPONSE

Device address	←	[7]
Function code	←	[5]
Data	←	[0] [2] [FF] [00]
Error check		

La respuesta es exactamente igual a la petición

Excepciones

Address field	**Function field** (1 byte)	**Data field** (1 byte)	**Error check field**
1 byte	Function code+ 0x80	EXCEPTION CODES	2 bytes

- 01 ILLEGAL FUNCTION
- 02 ILLEGAL DATA ADDRESS
- 03 ILLEGAL DATA VALUE
- 06 SLAVE DEVICE BUSY

Cuando tiene lugar una excepción la respuesta sólo contiene **5 bytes**.

Por ejemplo, intentamos escribir en un registro de salida que no existe en el PLC
Request -> 04 05 00 09 00 X X
Response-> 04 85 02 X X

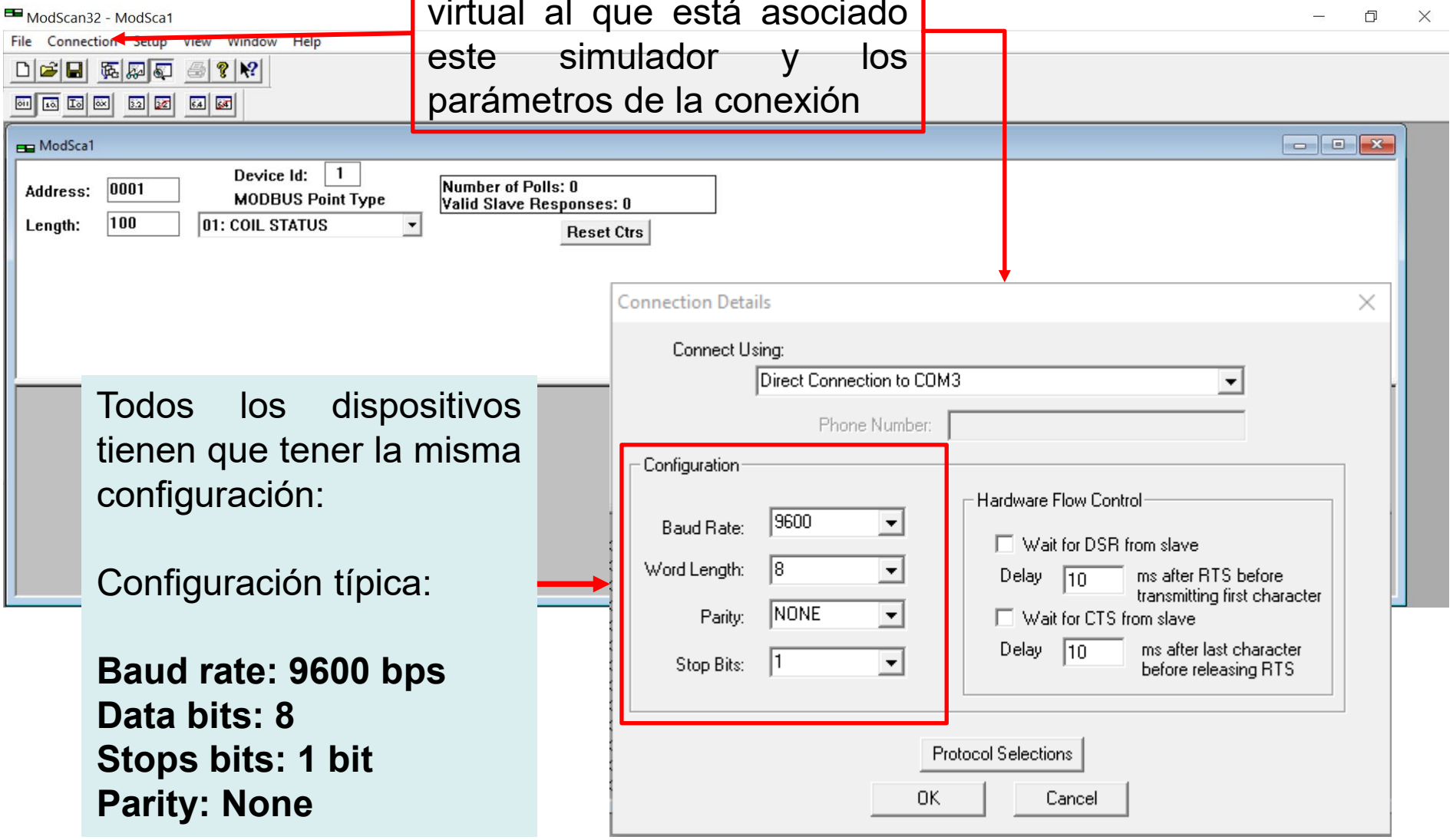
Exception funtion code: 0x80+ 0x05

Exception code: illegal data address

Herramientas de simulación Modbus

ModScan32 (Simulador maestro Modbus)

Configuración el puerto COM virtual al que está asociado este simulador y los parámetros de la conexión

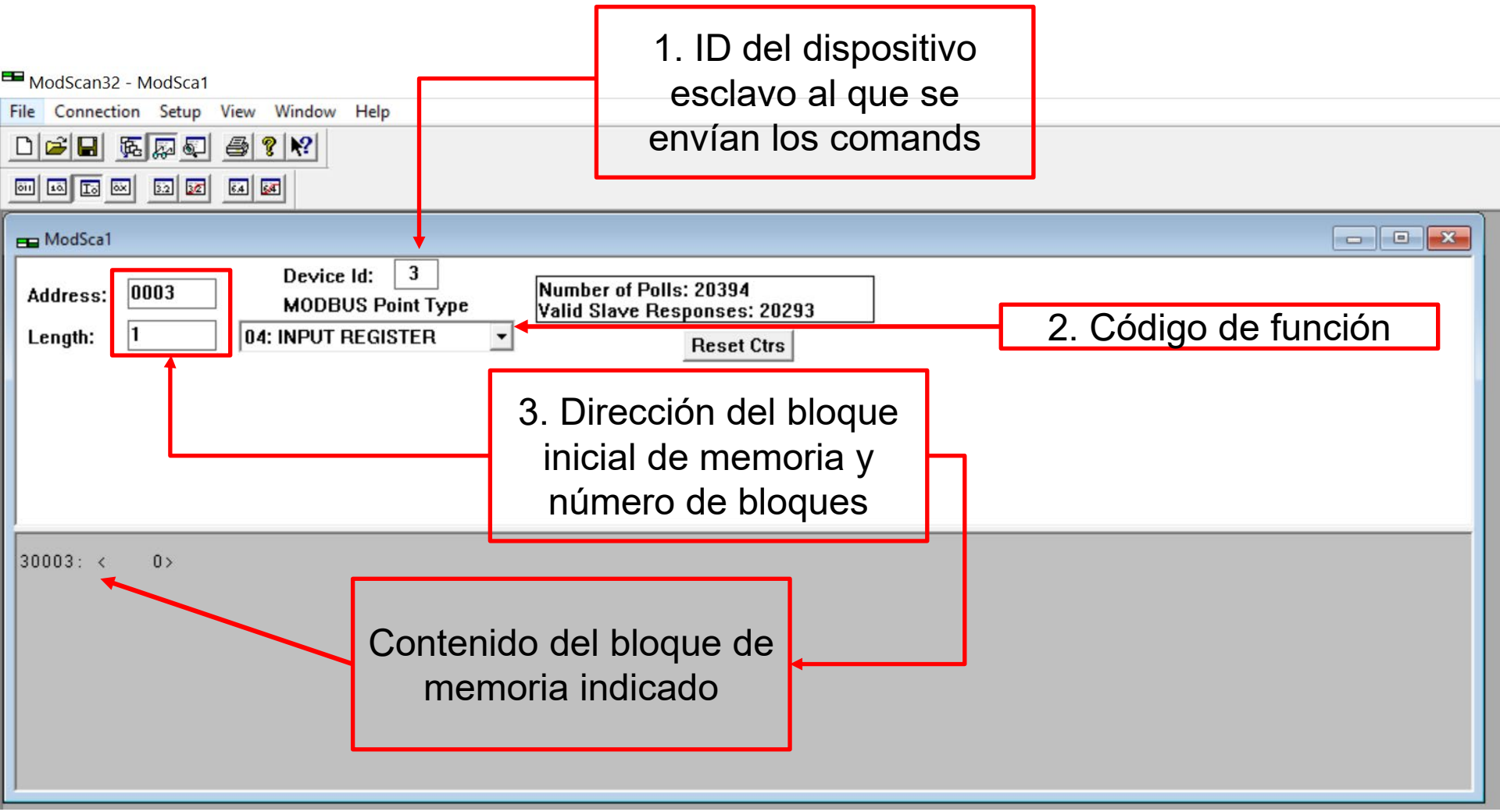


Todos los dispositivos tienen que tener la misma configuración:

Configuración típica:

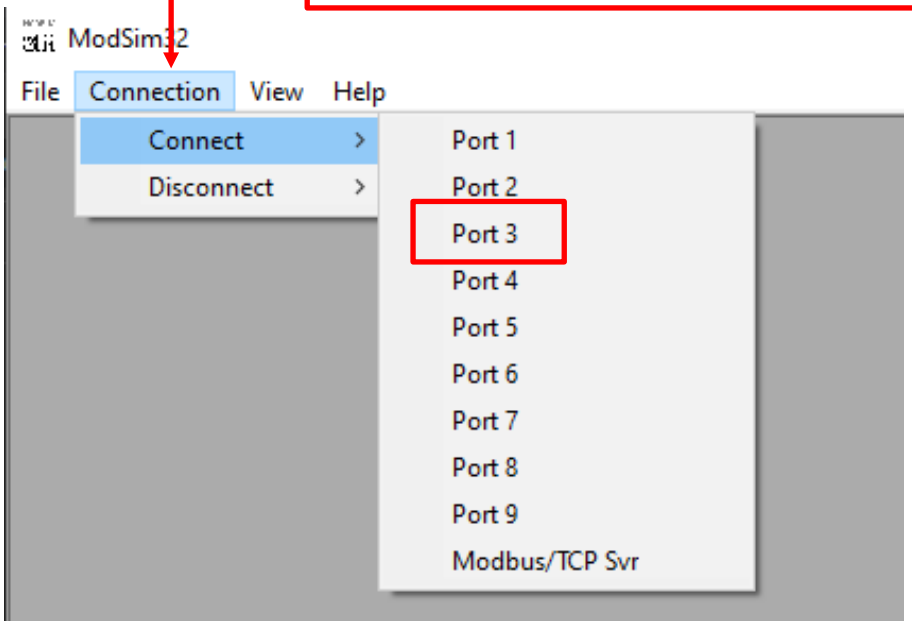
Baud rate: 9600 bps
Data bits: 8
Stops bits: 1 bit
Parity: None

ModScan32 (Simulador maestro Modbus)

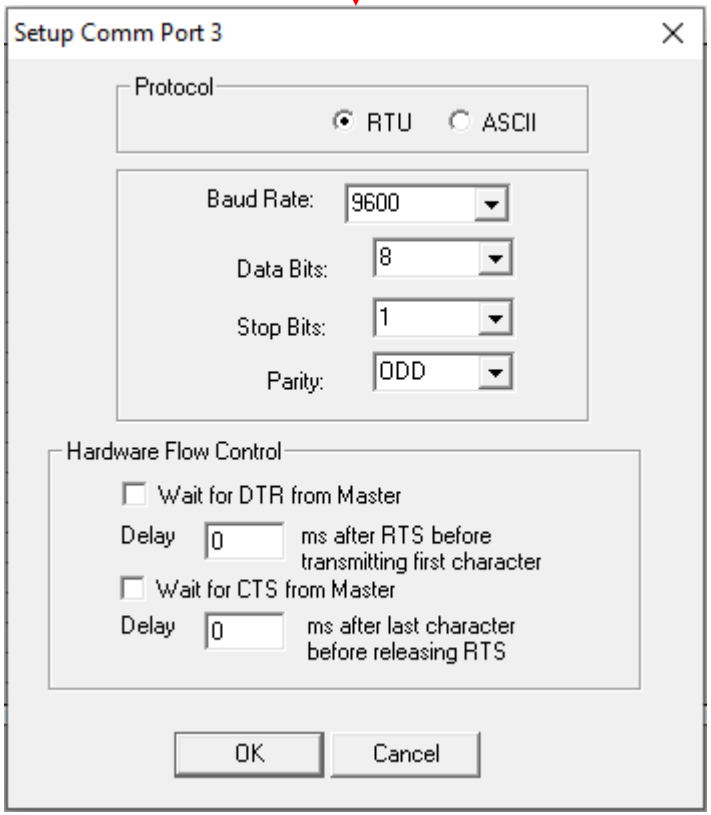


ModSim32 (Simulador esclavo Modbus)

Configurar el puerto COM virtual al que está asociado este simulador

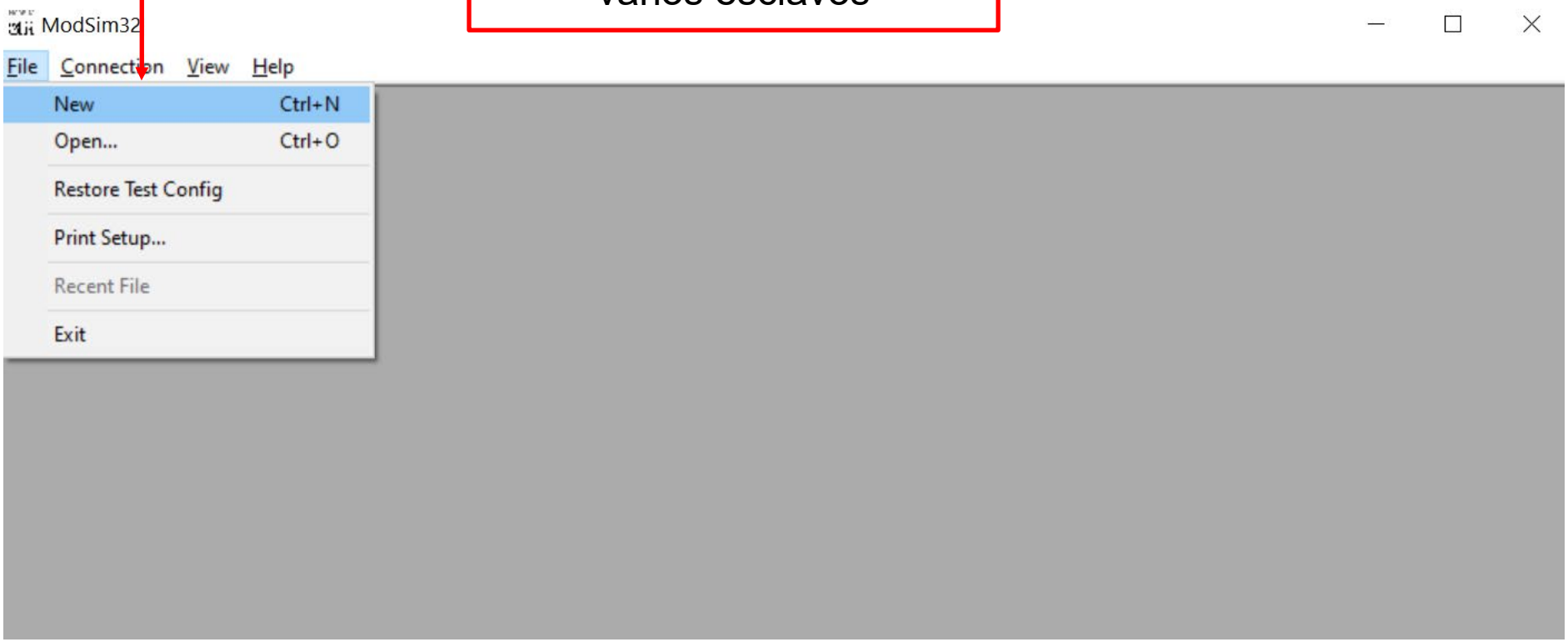


Configuración de la conexión

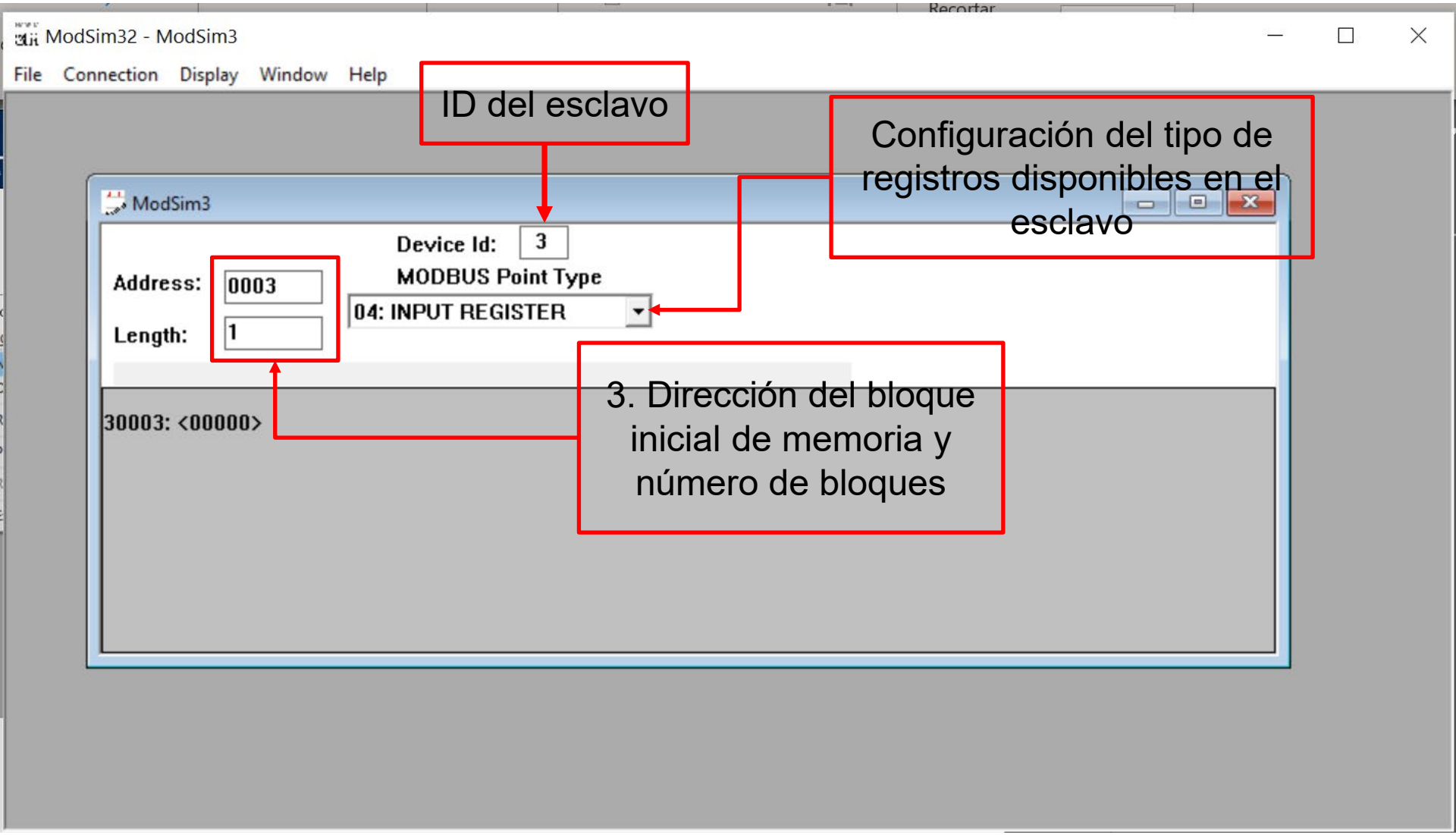


ModSim32 (Simulador del esclavo)

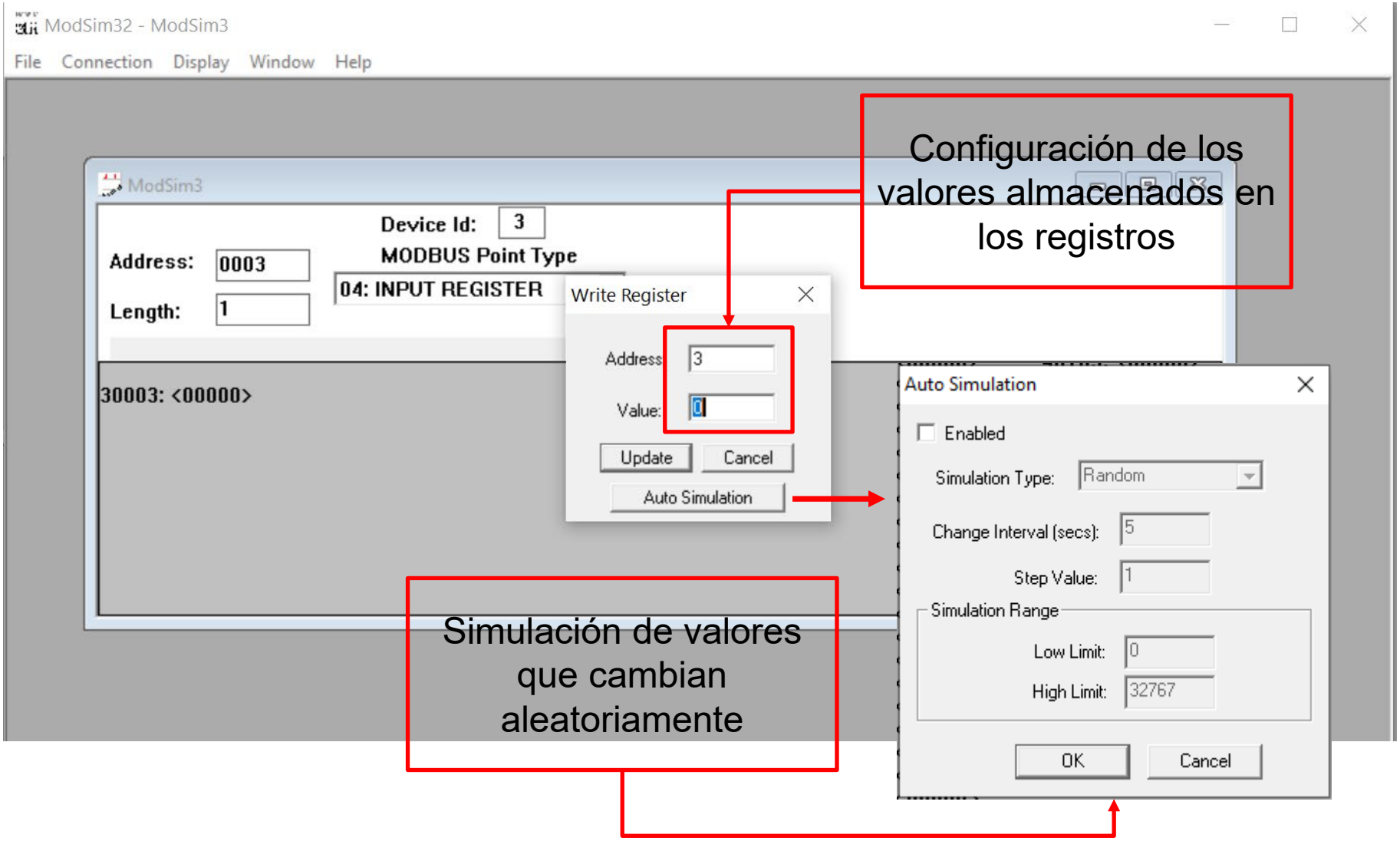
Crear una nueva instancia de esclavo, se puede tener varios esclavos



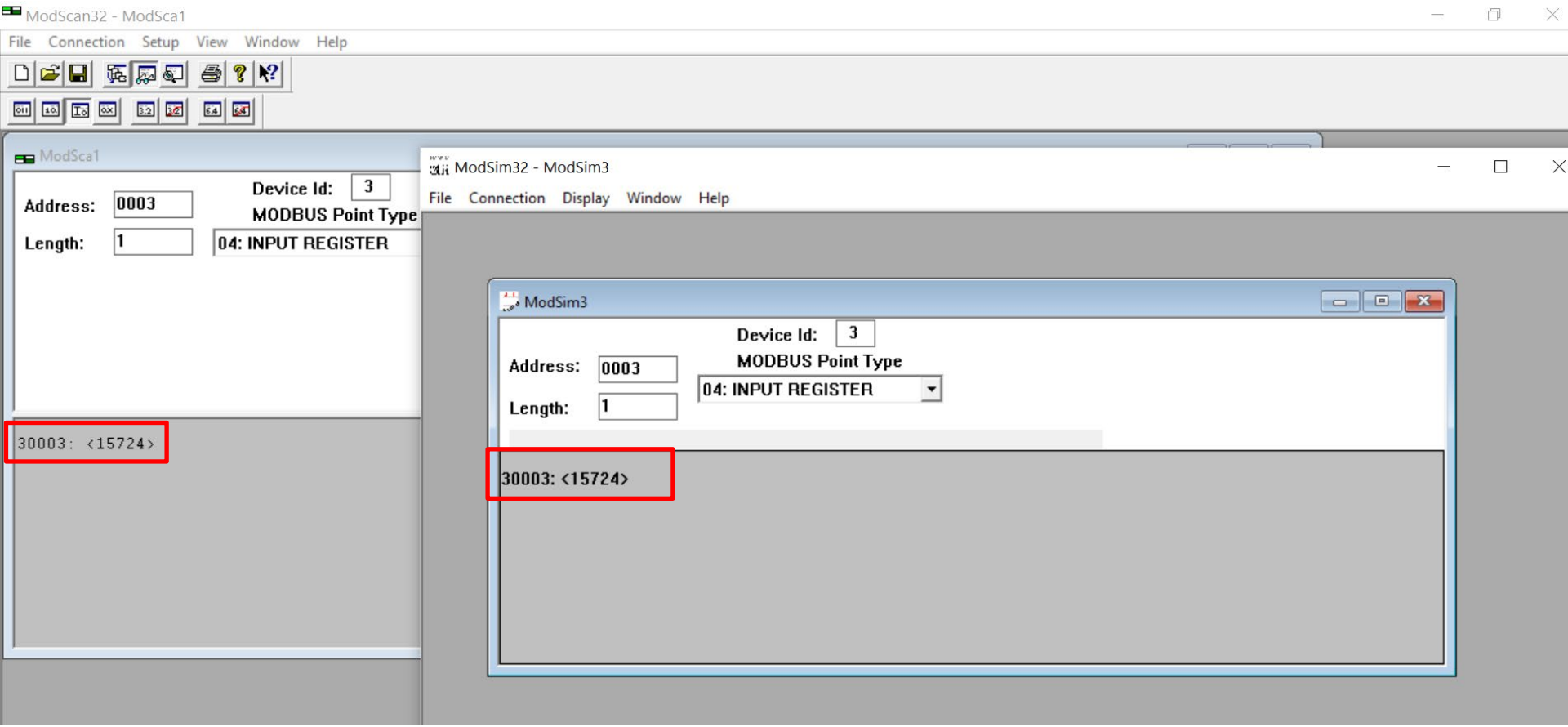
ModSim32 (Simulador del esclavo)



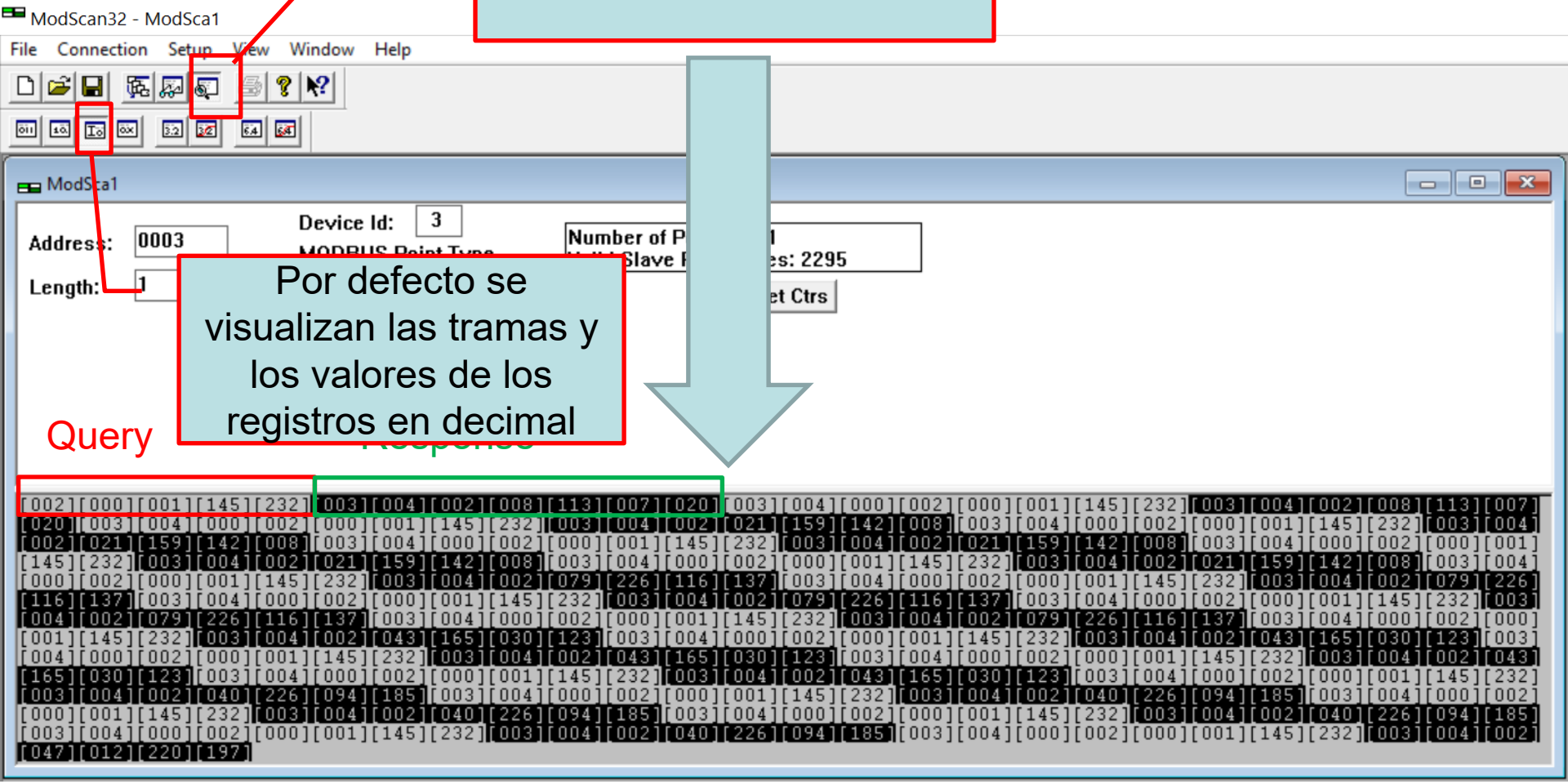
ModSim32 (Simulador del esclavo)



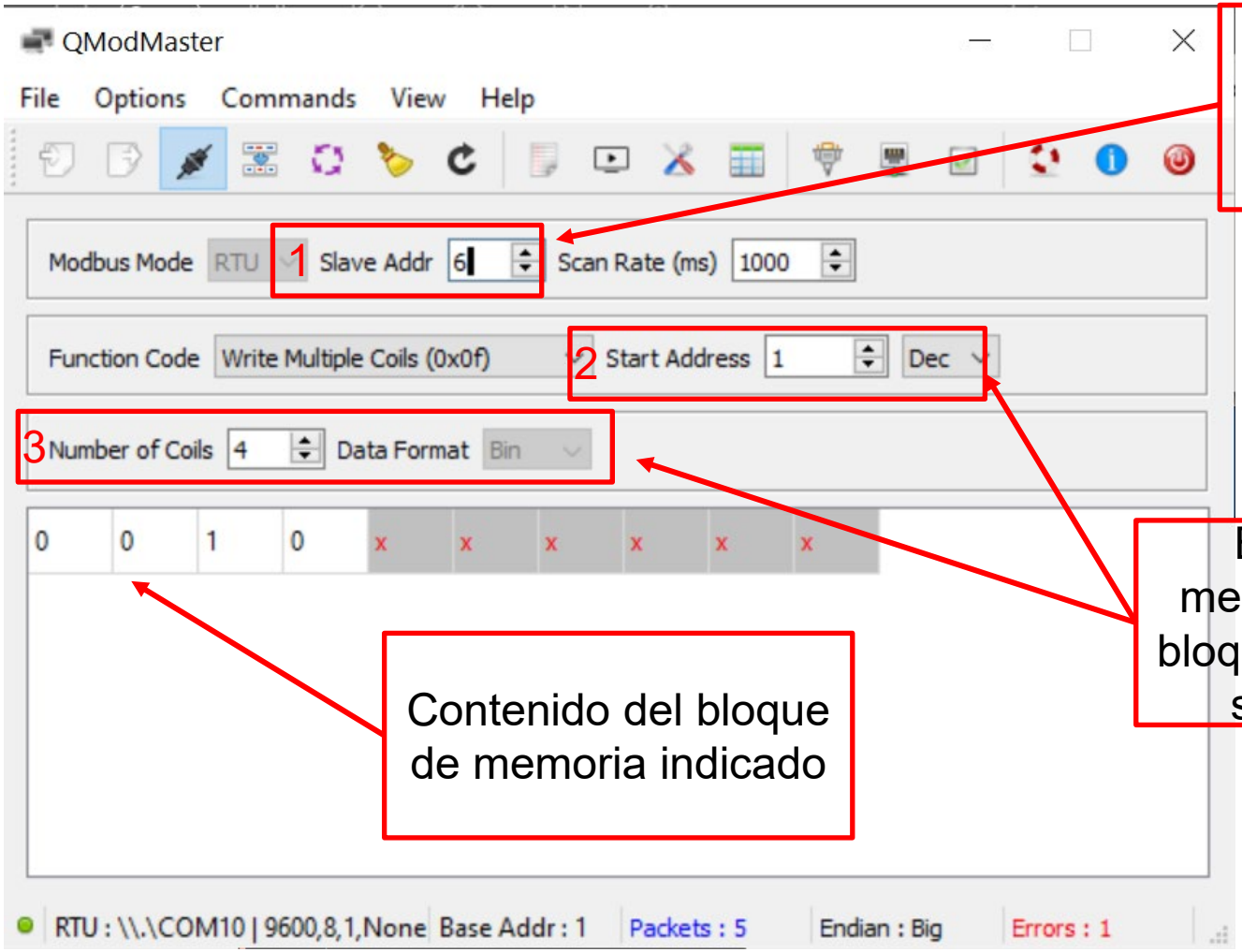
ModScan32 and ModSim32



Show traffic!



QModMaster

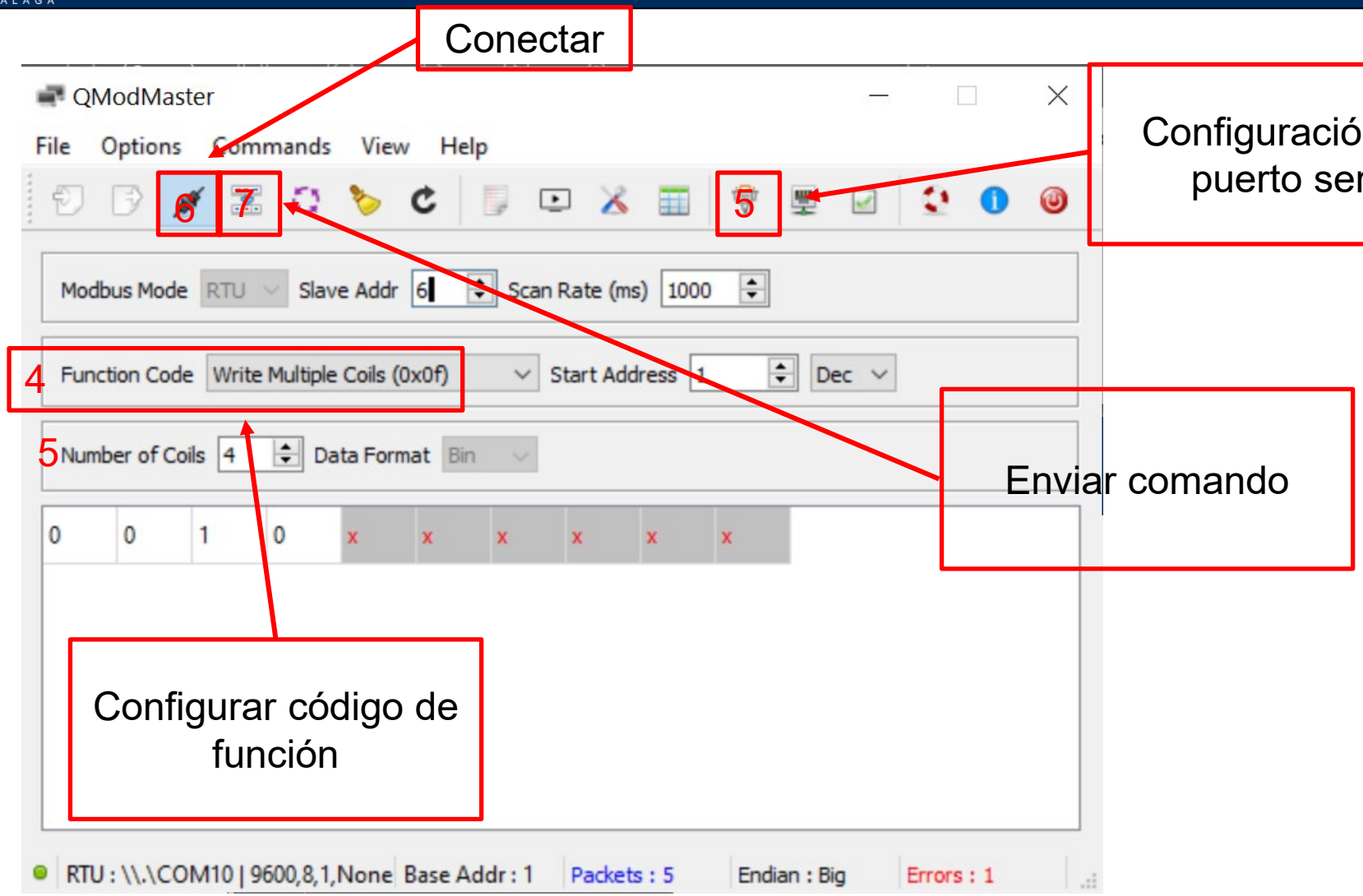


ID del dispositivo esclavo al que se envían los comands

Bloque inicial de memoria y número de bloques del esclavo que se van a solicitar

Contenido del bloque de memoria indicado

QModMaster


Conectar
QModMaster
File Options Commands View Help
6
7
5
Modbus Mode RTU Slave Addr 6 Scan Rate (ms) 1000
4 Function Code Write Multiple Coils (0x0f) Start Address 1 Dec
5 Number of Coils 4 Data Format Bin
0 0 1 0 x x x x x x
RTU : \\.\COM10 | 9600,8,1,None Base Addr : 1 Packets : 5 Endian : Big Errors : 1
Configuración del puerto serie
Enviar comando
Configurar código de función

QModMaster

View -> Bus Monitor

Se muestra el tráfico

Raw Data

```
[RTU]>Rx > 19:47:33:824 - 03 01 02 03 00 C0 CC
[RTU]>Tx > 19:47:33:875 - 03 01 00 01 00 10 6D E4
[RTU]>Rx > 19:47:33:986 - 03 01 02 03 00 C0 CC
[RTU]>Tx > 19:53:57:939 - 03 0F 00 00 00 10 02 18 C0 F1 10
[RTU]>Rx > 19:53:58:044 - 03 0F 00 00 00 10 55 E5
Sys > 19:53:58:044 - values written correctly.
[RTU]>Tx > 20:07:22:806 - 03 0F 00 00 00 10 02 1C C0 F3 D0
[RTU]>Rx > 20:07:22:913 - 03 0F 00 00 00 10 55 E5
Sys > 20:07:22:913 - values written correctly.
```

ADU

```
Type : Tx Message
Timestamp : 19:53:57:939
Slave Addr : 03
Function Code : 0F
Starting Address : 0000
Quantity of Registers : 0010
Byte Count : 02
Output Values : 18 C0
CRC : F110
```

Hands on!

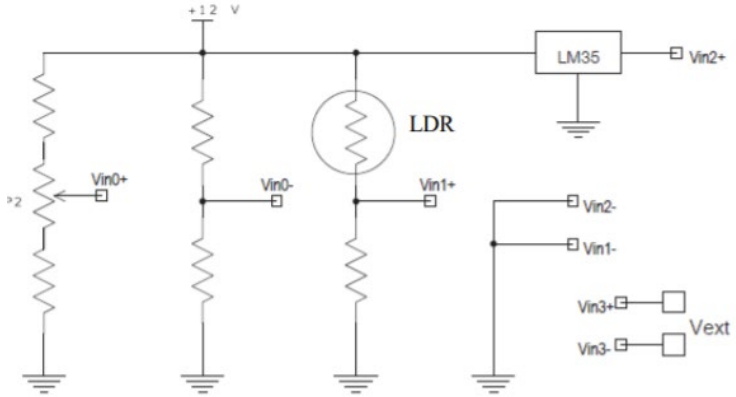
1.Crear dos instancias de esclavos usando la herramienta ModSim32 para simular los dispositivos M-7017D y M-7060D .

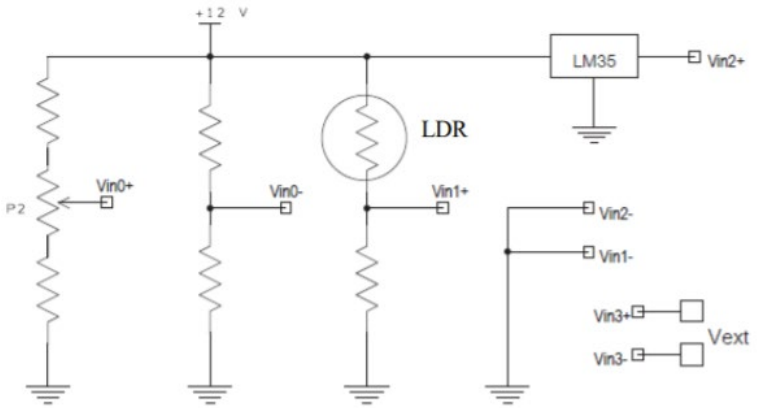
2. Dado el esquemático de la placa prototipo con al que se va a trabajar y usando los simuladores, enviar un comando para que, de forma simultánea, se apague el ventilador y se enciendan los leds.

3. Usando los simuladores enviar un comando para leer la temperatura del sensor LM35.

Consultar los manuales de los módulos M-7017D y M-7060D disponibles en el campus virtual de la asignatura.

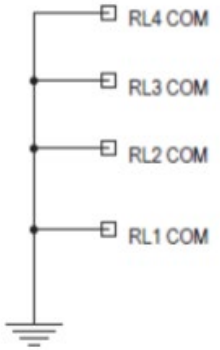
Esquemático tarjeta prototipo





Conector trasero 7060

RL1 NO	(11)	RL3 NO	(15)
RL1 COM	(12)	RL3 COM	(17)
RL2 NO	(13)	RL4 NO	(18)
RL2 COM	(14)	RL4 COM	(20)



Conector trasero 7017

Vin0+ (11)	Vin2+ (11)
Vin0- (12)	Vin2- (12)
Vin1+ (13)	Vin3+ (13)
Vin1- (14)	Vin3- (14)

Hands on!

M-7017D

Especificaciones:

Model: M-7017

Specifications

Analog Input	
Input Channels	8 differential
Input Type	+/-150mV, +/-500mV , +/-1V, +/-5V, +/-10V +/-20mA (requires optional external 125 ohm resistor)
Resolution	16-bit
Sampling Rate	10 Samples/ Second
Accuracy	+/-0.1%
BandWidth	15.7 Hz
Zero drift	+/-20uV/°C
Span drift	+/-25ppm/°C
Input Impedance	20M Ohms
Common Mode Rejection	86dB min
Normal Mode Rejection	100dB
Over voltage Protection	+/-35V(P-P)
Intra-module Isolation, Field to Logic	3000 VDC
Interface	
Interface	RS-485
Format	N, 8, 1
Baud Rate	1200 ~ 115200bps
LED Display	
1 LED as Power/ Communication indicator	
Power	
Input	
Power Consumption	
Protocol	
Support Modbus and DC	
Environment	
Operating temperature	
Storage temperature	
Humidity