Sécurité des Systèmes Industriels

Les réseaux MODBUS

Maxime Puys

22 mars 2025

CLERMONT AUVERGNE INP Isima

Historique et utilisations

Contexte 1/2

- Avant les **années 1970**, les industriels utilisaient des **systèmes propriétaires** incompatibles entre eux.
- Chaque fabricant d'automates (PLC) avait son propre protocole de communication.
- Implique une forte dépendance à une **supply chain** :
  - Quasiment impossible de changer de marque de matériel sans devoir tout changer.
- Besoin d'un protocole **standardisé et simple** pour faciliter l'interopérabilité.

Contexte 2/2

- En **1979**, Modicon (Schneider Electric) développe **MODBUS** pour ses automates.
- **Objectif :** unifier la communication entre automates et équipements industriels.
- Modbus devient rapidement populaire pour son **ouverture** et sa **simplicité**.
- Encore aujourd'hui, il reste un standard **très utilisé** dans l'industrie.

Intérêt historique de MODBUS

- **Premier protocole ouvert** pour l'automatisation industrielle.
- Historiquement développé pour fonctionner sur **RS-232** et **RS-485**.
- Permet un dialogue **Maître/Esclave** simple et efficace.
- Facile à **implémenter** et **adapter** sur différentes architectures.

Chronologie de l'invention de MODBUS

- **1979** : Création de Modbus par **Modicon** pour ses **PLCs**.
- **1981** : Standardisation et adoption dans d'autres équipements.
- **1990s** : Extension avec **Modbus TCP/IP** pour réseaux Ethernet.
- **1997** : Acquisition de MODICON par Schneider Electric.
- **2004** : **Modbus Organization** devient l'organisme de standardisation officiel.

Place de MODBUS dans les systèmes industriels

- **Protocole universel** utilisé dans les systèmes d'**énergies**, l'**industrie 4.0**, etc.
- Intégré dans des **millions** d'équipements industriels.
- **Interopérabilité** entre équipements de différents fabricants.
- Quantité enorme de réseaux MODBUS en production, très difficiles à faire évoluer.

Fonctionnement physique d'un bus MODBUS

Architectures Maître/Esclave (Unicast)

- Un **Maître** envoie une requête à un **Esclave** spécifique (adresse unique).
- L'Esclave exécute la commande et répond uniquement au Maître.

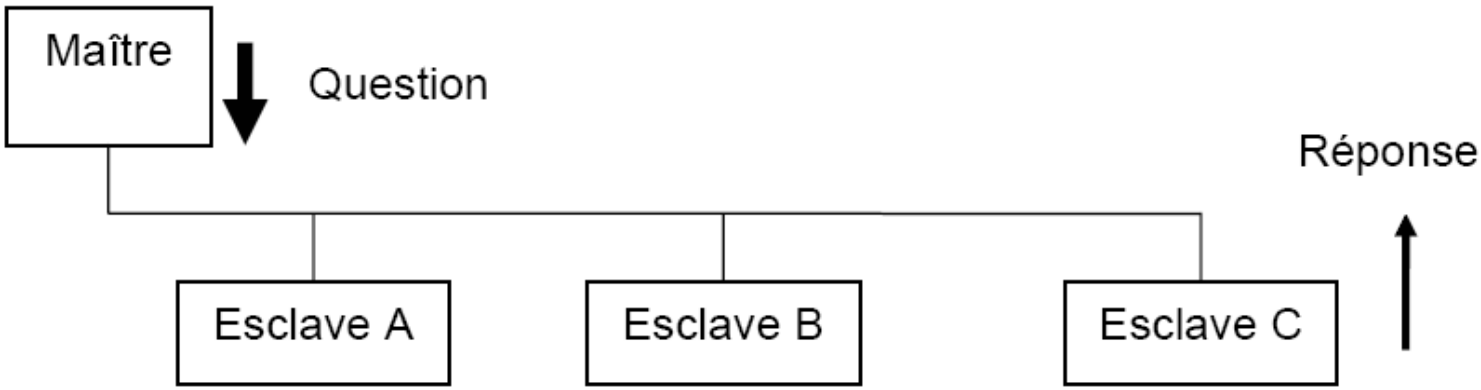

Figure – MODBUS Unicast

Architectures Maître/Esclave (Broadcast)

- Le Maître envoie une commande à **tous les Esclaves** en même temps.
- **Adresse 0** est utilisée pour une diffusion (*broadcast*).
- Les esclaves exécutent la commande mais **ne répondent pas**.

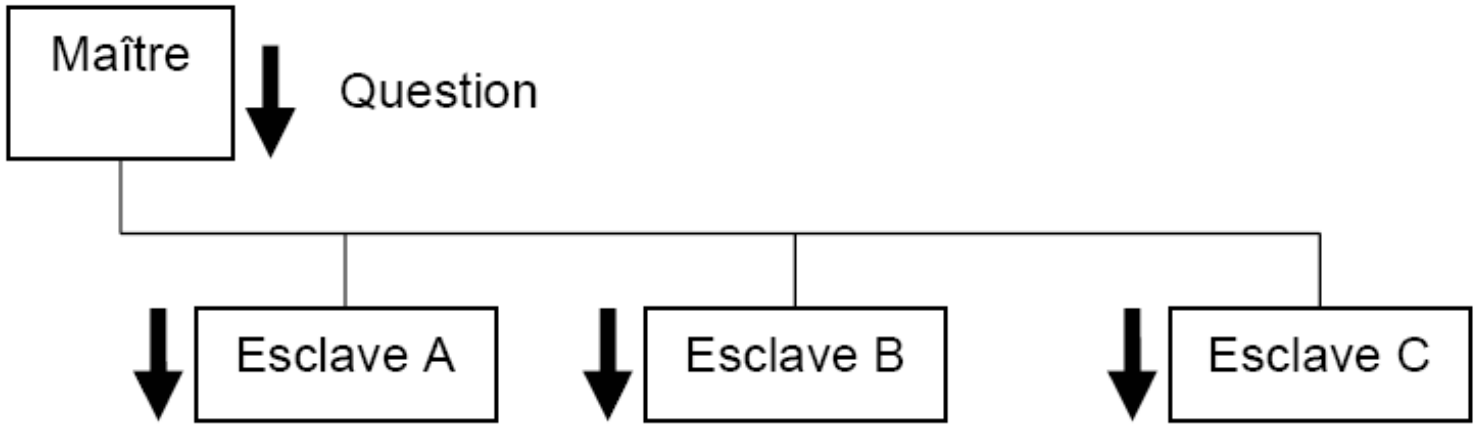

Figure – MODBUS Broadcast

Plusieurs types de MODBUS

- **Modbus RTU (Remote Terminal Unit)**
  - Fonctionne sur **bus série** (RS-232, RS-485).
  - Utilise un format **binaire compact**.
  - Vérification d'erreur avec **CRC**.
- **Modbus ASCII**
  - Identique à RTU mais encodé en **ASCII** (lisible).
  - Vérification d'erreur avec **LRC**.
  - **Moins efficace** que RTU (messages plus longs).
- **Modbus TCP**
  - Fonctionne sur **réseau Ethernet (TCP/IP)**.
  - Remplace l'adresse esclave par un **identifiant de session**.
  - Plus rapide et permet plusieurs Maîtres.

ADU vs. PDU

PDU (Protocol Data Unit) :

- Partie **commune** entre Modbus RTU et TCP.
- **Structure :**
  - **Code de fonction** (1 octet).
  - **Données** (n octets).

ADU (Application Data Unit) :

- Diffère selon le type de Modbus :
  - **RTU** : Adresse esclave + PDU + CRC.
  - **TCP** : En-tête MBAP + PDU.

Format	Préfixe	PDU	Suffixe
Modbus RTU	Adresse esclave	Code fonction + Données	CRC
Modbus TCP	MBAP Header	Code fonction + Données	Aucun

MODBUS sur Bus Série

- Plusieurs supports physiques possibles :

Caractéristique	RS-232	RS-422	RS-485
Nombre de fils	3	5	3
Distance max	15m	1200m	1200m
Vitesse max	115 kbps	10 Mbps	10 Mbps
Communication	Point à point	Point à point	Multi-point
Esclaves max	1	10	32 (extensible)
Utilisation	Liaison PC-PLC	Liason longue	Réseaux indus.

Structure d'un message Modbus RTU

Format	Préfixe	PDU	Suffixe
Modbus RTU	Adresse esclave	Code fonction + Données	CRC

Champ	Taille	Description
Adresse esclave	1 octet	Identifie l'esclave ciblé
Code fonction	1 octet	Décrit l'action demandée
Données	n octets	Paramètres de la commande
CRC	2 octets	Vérification d'erreur

Exemples de Requêtes et Réponses en Modbus RTU

Échange avec l'Esclave 1

Requête 1 : Lecture de données

```
[ESCLAVE 1] [CODE FONCTION] [DONNEES] [CRC]
```

Réponse 1 : Données demandées

```
[ESCLAVE 1] [CODE FONCTION] [DONNEES] [CRC]
```

Requête 2 : Écriture d'une valeur

```
[ESCLAVE 1] [CODE FONCTION] [DONNEES] [CRC]
```

Réponse 2 : Confirmation d'écriture

```
[ESCLAVE 1] [CODE FONCTION] [DONNEES] [CRC]
```

Exemples de Requêtes et Réponses en Modbus RTU

Échange avec l'Esclave 2

Requête 3 : Lecture d'un registre

```
[ESCLAVE 2] [CODE FONCTION] [DONNEES] [CRC]
```

Réponse 3 : Données du registre

```
[ESCLAVE 2] [CODE FONCTION] [DONNEES] [CRC]
```

Requête 4 : Écriture multiple

```
[ESCLAVE 2] [CODE FONCTION] [DONNEES] [CRC]
```

Réponse 4 : Confirmation d'écriture

```
[ESCLAVE 2] [CODE FONCTION] [DONNEES] [CRC]
```

Structure d'un message Modbus TCP

Format	Préfixe	PDU	Suffixe
Modbus TCP	MBAP Header	Code fonction + Données	Aucun

- **Pas de CRC**, car TCP gère la vérification d'intégrité.

Champ	Taille	Description
Transaction ID	2 octets	Identifiant unique pour le suivi
Protocole ID	2 octets	Toujours `0x0000` (Modbus)
Longueur	2 octets	Taille du message suivant
Unité ID	1 octet	Identifie un esclave spécifique
Code fonction	1 octet	Action demandée
Données	n octets	Paramètres de la commande

CLERMONT AUVERGNE INP Isima

Exemples de Requêtes et Réponses en Modbus TCP

Échange avec l'Esclave 1

Requête 1 : Lecture de données

```
1 [TRANSACTION 1] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 1] [CODE FONCTION] [DONNEES]
```

Réponse 1 : Données demandées

```
1 [TRANSACTION 1] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 1] [CODE FONCTION] [DONNEES]
```

Requête 2 : Écriture d'une valeur

```
1 [TRANSACTION 2] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 1] [CODE FONCTION] [DONNEES]
```

Réponse 2 : Confirmation d'écriture

```
1 [TRANSACTION 2] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 1] [CODE FONCTION] [DONNEES]
```

Exemples de Requêtes et Réponses en Modbus TCP

Échange avec l'Esclave 2

Requête 3 : Lecture d'un registre

```
[TRANSACTION 3] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 2] [CODE FONCTION] [DONNEES]
```

Réponse 3 : Données du registre

```
[TRANSACTION 3] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 2] [CODE FONCTION] [DONNEES]
```

Requête 4 : Écriture multiple

```
[TRANSACTION 4] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 2] [CODE FONCTION] [DONNEES]
```

Réponse 4 : Confirmation d'écriture

```
[TRANSACTION 4] [PROTOCOLE ID] [LONGUEUR] [ESCLAVE 2] [CODE FONCTION] [DONNEES]
```

Couche applicative d'un bus MODBUS

Types de données en MODBUS

- 4 types de données manipulables :

	RO	RW
Bool	Discrete Input	Coil
uint16	Input Register	Holding Register

- **Coils (Sorties binaires) :** Bits modifiables (activés/désactivés) par le Maître.
- **Discrete Inputs (Entrées binaires) :** Bits lecture seule, souvent liés à des capteurs.
- **Holding Registers :** Valeurs 16 bits modifiables (consignes, variables).
- **Input Registers :** Valeurs 16 bits lecture seule, souvent utilisées pour les mesures.

Exemples

Exemple d'Utilisation	**Type de Donnée**
Activation d'un moteur, d'un relais	
Lecture d'un bouton poussoir	
Réglage d'une vitesse, d'un seuil	
Température mesurée, tension, courant	

Exemples

Exemple d'Utilisation	**Type de Donnée**
Activation d'un moteur, d'un relais	**Coil** (RW)
Lecture d'un bouton poussoir	
Réglage d'une vitesse, d'un seuil	
Température mesurée, tension, courant	

Exemples

Exemple d'Utilisation	**Type de Donnée**
Activation d'un moteur, d'un relais	**Coil** (RW)
Lecture d'un bouton poussoir	**Discrete Input** (RO)
Réglage d'une vitesse, d'un seuil	
Température mesurée, tension, courant	

Exemples

Exemple d'Utilisation	**Type de Donnée**
Activation d'un moteur, d'un relais	**Coil** (RW)
Lecture d'un bouton poussoir	**Discrete Input** (RO)
Réglage d'une vitesse, d'un seuil	**Holding Register** (RW)
Température mesurée, tension, courant	

Exemples

Exemple d'Utilisation	**Type de Donnée**
Activation d'un moteur, d'un relais	**Coil** (RW)
Lecture d'un bouton poussoir	**Discrete Input** (RO)
Réglage d'une vitesse, d'un seuil	**Holding Register** (RW)
Température mesurée, tension, courant	**Input Register** (RO)

Types de requêtes possibles

	RO	RW
READ	DI / IR	CO / HR
WRITE_SINGLE		CO / HR
WRITE_MULTIPLE		CO / HR
READ_WRITE_MULT		HR

Types de requêtes possibles

Type		Nom	Code
Bool	Read	Read DI	2
		Read CO	1
	Write	Write_S CO	5
		Write_M CO	15
uint16	Read	Read IR	4
		ReadM HR	3
	Write	Write_S HR	6
		Write_M HR	16
		Read/Write_M HR	23

Fonctions 1 & 2 - Read Boolean

Request

Function code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x01**
Byte count	1 Byte	**N***
Coil Status	**n** Byte	n = N or N+1

Figure – Spécs des codes 1 et 2

- *N = Nombre de booléens retounés / 8, partie suppérieure
- Les bits sont retournés packés par octers (première adresse en LSB), padding par zéro si besoin.

Fonctions 1 & 2 - Read Boolean

- **Exemple :** Lecture des coils 19 à 37 (inclu).

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**01**	Function	**01**
Starting Address Hi	**00**	Byte Count	**03**
Starting Address Lo	**13**	Outputs status 27-20	**CD**
Quantity of Outputs Hi	**00**	Outputs status 35-28	**6B**
Quantity of Outputs Lo	**13**	Outputs status 38-36	**05**

Figure – Spécs des codes 1 et 2

Fonctions 1 & 2 - Read Boolean

Adresse	Octet	Positions	Bits
0x13	**0xCD** (0b11001101)	Bit 0	**1**
0x14		Bit 1	**0**
0x15		Bit 2	**1**
...		...	...
0x1A		Bit 7	**1**
0x1B	**0x6B** (0b01101011)	Bit 0	**1**
...		...	...
0x22		Bit 7	**0**
0x23	**0x05** (0b00000101)	Bit 0	**1**
0x24		Bit 1	**0**
0x25		Bit 2	**1**

Fonctions 3 & 4 - Read uint16

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

***N** = Quantity of Registers

Figure – Spécs des codes 3 et 4

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**03**	Function	**03**
Starting Address Hi	**00**	Byte Count	**06**
Starting Address Lo	**6B**	Register value Hi (108)	**02**
No. of Registers Hi	**00**	Register value Lo (108)	**2B**
No. of Registers Lo	**03**	Register value Hi (109)	**00**
		Register value Lo (109)	**00**
		Register value Hi (110)	**00**
		Register value Lo (110)	**64**

Figure – Spécs des codes 3 et 4

Fonctions 5 - Write Single Coil

Request

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Figure – Spécs du code 5

Fonctions 6 - Write Single Register

Request

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Response

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 to 0xFFFF

Figure – Spécs du code 6

Fonctions 15 - Write Multiple Coils

Request PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0
Byte Count	1 Byte	**N***
Outputs Value	**N*** x 1 Byte	

***N** = Quantity of Outputs / 8, if the remainder is different of 0 $\Rightarrow$ N = N+1

Response PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0

Error

Error code	1 Byte	**0x8F**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to write a series of 10 coils starting at coil 20:

The request data contents are two bytes: CD 01 hex (1100 1101 0000 0001 binary). The binary bits correspond to the outputs in the following way:

Bit:	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1
Output:	27	26	25	24	23	22	21	20	–	–	–	–	–	–	29	28

Figure – Spécs du code 15

Fonctions 16 - Write Multiple Registers

Request

Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	0x0001 to 0x007B
Byte Count	1 Byte	2 x **N***
Registers Value	**N*** x 2 Bytes	value

***N** = Quantity of Registers

Response

Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 123 (0x7B)

Figure – Spécs du code 16

Fonctions 23 - Read/Write Multiple Registers

Request

Function code	1 Byte	**0x17**
Read Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Read	2 Bytes	0x0001 to 0x007D
Write Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Write	2 Bytes	0x0001 to 0X0079
Write Byte Count	1 Byte	2 x **N***
Write Registers Value	**N***x 2 Bytes	

***N** = Quantity to Write

Response

Function code	1 Byte	**0x17**
Byte Count	1 Byte	2 x **N'***
Read Registers value	**N'*** x 2 Bytes	

***N'** = Quantity to Read

Figure – Spécs du code 23

Attaques sur MODBUS

Sécurité de MODBUS

- Aucune, donc vulnérable à toute attaque pour forger, modifier, bloquer, inverser, etc.
- Sur MODBUS TCP, n'importe quelle attaque en MITM
- Sur MODBUS RTU, la communication est en broadcast comme avec CAN.

Références