الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département : Génie électrique Et Informatique Industrielle

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'Etat en télécommunication

Spécialité
Systèmes des Télécommunications et Réseaux

Développement d'une passerelle Modbus RTU/Modbus TCP pour communication SCADA sécurisée

Réalisé par

BELHASSANI Yasmine
SOLTANI Ourida

Composition du Jury :

Président	M. BOUDIAR Toufik	M.C.A	ENST
Promoteur	Mme. BOUTERFAS Malika	M.C.A	ENST
Co-promoteur	M. CHERIFI Tarek	M.C.B	ENST
Examinateur	M. KHOUMERI El Hadi	M.C.B	ENST
Examinateur	M. BOUCHACHI Islem	M.C.B	ENST

Année universitaire : 2022/2023

Remerciement

Nous sommes profondément reconnaissants envers nos professeurs et encadrants universitaires qui nous ont transmis des connaissances précieuses tout au long de notre parcours académique. Leurs enseignements et leurs encouragements ont été d'une importance cruciale pour façonner nos réflexions et développer nos compétences techniques. Leur expertise et leur disponibilité ont été inestimables et ont grandement contribué à la réussite de ce mémoire.

Nous souhaitons également exprimer notre gratitude envers nos familles et nos amis, qui nous ont soutenus et encouragés tout au long de ce parcours. Leur soutien moral indéfectible nous a aidés à surmonter les moments de doute et de fatigue, et leur présence bienveillante a été une source d'inspiration constante.

Enfin, nous tenons à remercier toutes les personnes qui ont participé de près ou de loin à ce projet. Leurs contributions, qu'il s'agisse de partager leur expertise, d'apporter des suggestions pertinentes ou d'offrir une aide précieuse, ont été essentielles pour la concrétisation de ce mémoire. Leur soutien et leur collaboration ont été d'une valeur inestimable.

La réalisation de ce mémoire a été une expérience enrichissante et formatrice, et nous sommes profondément reconnaissants envers toutes les personnes qui ont joué un rôle dans sa réussite. Leurs efforts conjugués ont permis de mener à bien ce projet et ont contribué à notre développement académique et professionnel.

Table de matière

Liste des tableaux

Liste des figures

Liste des abréviations

SCADA	Supervisory Control and Data Acquisition
ICS	Industrial Control System
PLC	Programable Logic Controller
RTU	Remote Terminal Unit
TCP	Transmission Control Protocol
MTU	Master Terminal Unit
IHM	Interface Homme-Machine
LAN	Local Area Network
GSM	Global System for Mobile Communications
DNP3	Distributed Network Protocol 3
DCS	Distributed Control System
API	Application Programming Interface
IEC	International Electrotechnical Commission
GPRS	General Packet Radio Service
LTE	Long-Term Evolution
ASCII	American Standard Code for Information Interchange
IP	Internet Protocol
UART	Universal Asynchronous Receiver-Transmitter
EIA/TIA	Electronic Industries Alliance/Telecommunications Industry Association
ADU	Application Data Unit
PDU	Protocol Data Unit
LSB	Least Significant Bit
MSB	Most Significant Bit
ID	Identifier

CRC	Cyclic Redundancy Check
MBAP	Modbus Application Protocol
ARM	Advanced RISC Machine
HDMI	High-Definition Multimedia Interface
USB	Universal Serial Bus
GPIO	General Purpose Input/Output
LCD	Liquid Crystal Display
VNC	Virtual Network Computing
GUI	Graphical User Interface
SSH	Secure Shell
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining

Introduction Générale

Les systèmes SCADA (Supervisory Control and Data Acquisition) jouent un rôle essentiel dans la gestion et le contrôle des processus industriels à grande échelle. Ils permettent aux opérateurs de superviser, de surveiller et de contrôler à distance des systèmes de production complexes dans des domaines tels que l'énergie, la fabrication, les infrastructures critiques et bien d'autres. Avec l'évolution rapide de l'automatisation industrielle, les systèmes SCADA sont devenus une composante centrale des systèmes de contrôle industriel [1]. Les systèmes de contrôle industriels (Industrial Control System ICS) englobent un ensemble de technologies et de systèmes qui assurent le contrôle et la supervision des processus industriels. Ils intègrent des dispositifs sur terrain tels que les capteurs, les actionneurs, les contrôleurs logiques programmables (PLC) et les systèmes SCADA. L'objectif des ICSs est de permettre une automatisation et une gestion efficaces des opérations industrielles, ce qui se traduit par des performances optimales, une productivité accrue et une sécurité améliorée dans les environnements industriels complexes [2].

Dans le domaine des systèmes SCADA, il existe plusieurs protocoles de communication utilisés pour l'échange de données entre les dispositifs de terrain et la salle de contrôle. Parmi ces protocoles, Modbus et ses variantes sont largement utilisés pour leur simplicité et leur mise en œuvre facile, ce qui a fait de Modbus le protocole industriel le plus implémenté. Cependant, pour permettre une intégration facile et une communication plus rapide et efficace, il est nécessaire de convertir le mode de communication de liaison série en mode de communication Ethernet. Cette transition offre une plus grande flexibilité, une portée étendue et bénéficie d'un support de développement solide.

Dans le cadre de ce projet de développement, nous nous sommes penchés sur la conception et l'implémentation d'une passerelle sécurisée entre les deux variantes : Modbus RTU et Modbus TCP. Les points suivants soulèvent les aspects clés à explorer :

- Interopérabilité : concevoir une passerelle qui assure une communication fluide entre les dispositifs utilisant les protocoles Modbus RTU et Modbus TCP,
- Gestion de la conversion des données : gérer la conversion des données,
- Sécurité de la passerelle : garantir la sécurité des données lors de la communication.

Notre travail est organisé en trois chapitres :

Le premier chapitre dresse les fondements théoriques pour notre étude des systèmes SCADA dans le contexte particulier de notre projet, à savoir le domaine de gestion des infrastructures énergétiques.

Le deuxième chapitre explore les principes fondamentaux du protocole Modbus, incluant sa structure de messages ainsi que ses différentes variantes, comme Modbus RTU, Modbus ASCII et Modbus TCP. Nous analysons en détail les caractéristiques et les spécificités de chaque variante.

Le troisième chapitre se consacre au développement de la passerelle Modbus RTU-Modbus TCP. Nous détaillons les différentes étapes de conception et implémentation de la passerelle, mettant en évidence ses fonctionnalités clés et les techniques de programmation utilisées.

Dans la conclusion générale, nous récapitulons les résultats principaux obtenus au cours de cette étude. Nous passons en revue les objectifs initiaux du projet et évaluons dans quelle mesure ils ont été atteints. Nous mettons en évidence les contributions et les réalisations clés de notre travail, en soulignant leur importance et leur impact potentiel dans le domaine des systèmes SCADA et des communications protocolaires dans le domaine industriel.

Chapitre I Les systèmes SCADA

I.1. Introduction

Dans chapitre nous abordons les concepts et les définitions liés aux systèmes SCADA, ainsi que leurs principales fonctions. Nous discutons également des aspects fondamentaux des communications dans les systèmes SCADA. Nous mettons en évidence leur importance et leurs différentes applications.

I.2. Introduction au système SCADA

SCADA, pour Supervisory Control And Data Acquisition, est une technologie qui permet à un utilisateur de collecter des données à partir d'une ou de plusieurs installations distantes et d'envoyer des instructions de contrôle limitées à ces installations [3].

Un système SCADA permet à un opérateur situé dans un emplacement central d'un processus largement réparti, tel qu'un champ pétrolier ou gazier, un système de pipelines, un système d'irrigation ou un complexe de production hydroélectrique, d'apporter des modifications aux points de consigne des contrôleurs de processus distants, d'ouvrir ou de fermer des vannes ou des interrupteurs, de surveiller les alarmes et de recueillir des informations de mesure.

SCADA englobe l'interface opérateur et la manipulation des données liées aux applications, mais ne se limite pas à cela. SCADA intègre également des éléments tels que les équipements de communication, les protocoles de transmission de données les unités terminales distantes (Remote Terminal Unit RTU), les contrôleurs logiques programmables (Programable Logic Controller PLC). L'interface opérateurs permet aux utilisateurs de visualiser et d'analyser les données en temps réel, et aussi de recevoir des alarmes et des alertes [1].

Pour illustrer concrètement l'environnement de contrôle et de surveillance des systèmes SCADA, la Figure 1 présente une image d'une salle de contrôle typique utilisée dans ces installations.

Figure 1: Salle de contrôle et de supervision [4].

I.2.1. Processus applicables à SCADA

Les processus applicables à la technologie SCADA sont très variés en milieu industriel. Les exemples ci-dessous illustrent quelques-uns des nombreux processus pouvant bénéficier de l'implémentation de cette technologie.

A. **Industrie pétrolière et gazière** : les installations de production de pétrole ou de gaz (les puits, les systèmes de collecte, les équipements de mesure des fluides et les pompes) sont généralement réparties sur de vastes zones. Ils nécessitent des commandes relativement simples telles que l'allumage de l'extinction des moteurs, et doivent collecter régulièrement des informations de mesure et réagir rapidement aux conditions dans le reste du champ.

B. **Transmission électrique** : les systèmes de transmission électrique peuvent couvrir des milliers de kilomètres carrés (Km^2). Ils peuvent être contrôlés en ouvrant et en fermant des interrupteurs pour répondre presque immédiatement aux variations de charge sur les lignes.

C. **Systèmes hydroélectriques** : les groupes de petites centrales hydroélectriques qui sont activées et désactivées en réponse à la demande des clients, généralement situées dans des endroits éloignés, peuvent être contrôlées en ouvrant et en fermant les vannes de la

turbine. Ils doivent être surveillés en continu et doivent répondre relativement rapidement aux demandes du réseau électrique.

D. **Gestion de l'eau** : les systèmes d'irrigation s'étendent généralement sur de vastes surfaces, et peuvent être gérés en contrôlant l'ouverture et la fermeture de vannes simples. Ils requirent également la collecte de mesures concernant l'eau fournie aux consommateurs [5].

I.3. Architecture SCADA

L'architecture SCADA fait référence à sa structure et ses composants. Cette architecture est illustrée de manière générale dans la figure 2. Du point de vue logiciel, on peut parler d'une architecture en couches, comprenant une couche client, couche de traitement, et une couche de données [6].

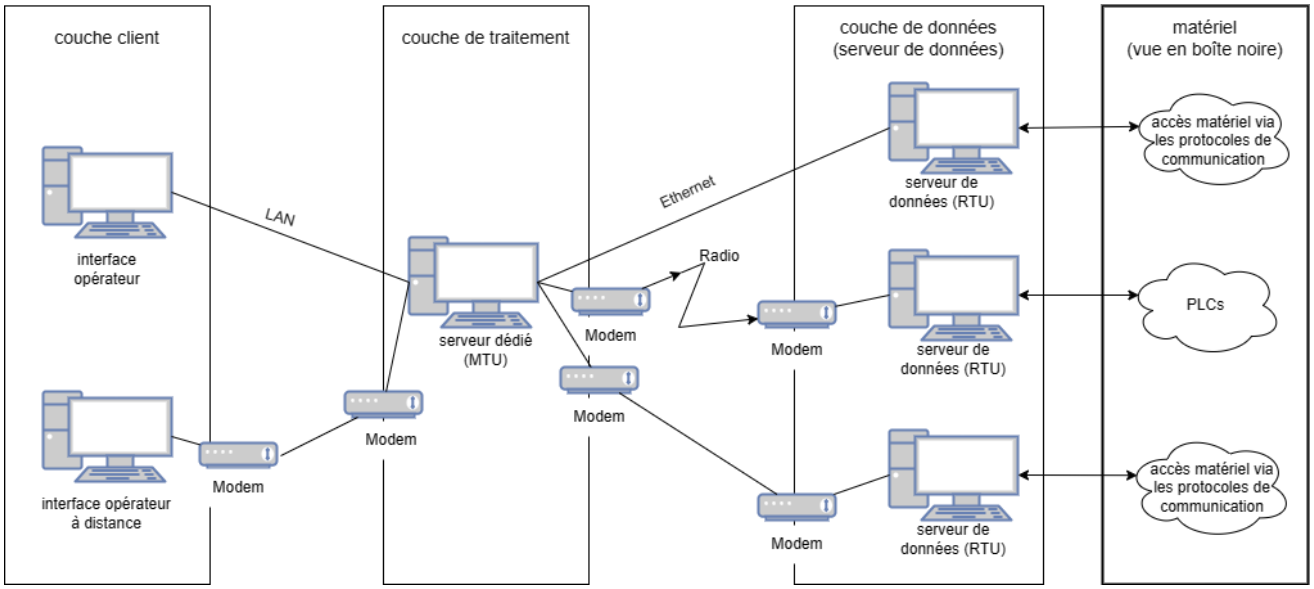

Figure 2: Architecture générale d'un système SCADA [6] .

Cette architecture illustre les aspects suivants :

- La séparation des couches client, de traitement et de données.
- La distinction entre les composants SCADA et les équipements contrôlés.
- Les différents moyens de communication utilisés pour l'échange de données entre la couche client et la couche de traitement, entre la couche de traitement et la couche de données, et enfin entre la couche de données et le matériel [6].

Nous allons maintenant expliquer les tâches accomplies par chacune des couches :

I.3.1. Couche Client

Au centre d'un système SCADA se trouvent les opérateurs qui accèdent au système par le biais d'une interface homme-machine (IHM), également appelée console d'opérateur ou interface utilisateur. La principale tâche de la couche client est d'afficher et de visualiser les informations reçues de la couche de traitement et d'interagir avec l'utilisateur [6].

I.3.2. Couche de traitement

Après avoir mentionné ce qui précède, il est clair que l'IHM interagit directement avec une unité terminale maître (Master Terminal Unit MTU) qui est située dans la couche de traitement. Les avantages de l'utilisation d'une MTU sont les suivants :

- Exécution automatique des tâches sans interaction de l'opérateur,
- Déclenchement d'alertes en cas de dépassement des seuils prédéfinis,
- Réalisation de calculs complexes au niveau du MTU pour éviter la redondance de calculs sur les clients,
- Distribution efficace des données aux clients de manière événementielle afin de minimiser la transmission d'informations via un moyen de communication à faible débit [6].

I.3.3. Couche de données

La couche de serveur de données dans un système SCADA assure la liaison avec le matériel et l'acquisition des données. Les serveurs de données, appelés unités terminales distantes (Remote Terminal Unit RTU), sont connectés aux MTU via différents supports de communication.

Ils sont responsables de la collecte des données à partir des équipements matériels tels que les automates programmables (Programable Logic Controller PLC) via des protocoles de communication [6].

I.4. Fonctions principales d'un système SCADA

Un système SCADA comprend 2 sous-ensembles fonctionnels [7] :

I.4.1. Le sous-ensemble Commande

La fonction de commande d'un système SCADA consiste à exécuter un ensemble d'opérations et le contrôle direct des actionneurs. Ainsi assurant les opérations suivantes :

- Le fonctionnement normal du procédé en l'absence de défaillance,

- La gestion des modes de fonctionnement ou leur reprise,
- Le traitement d'urgence en cas de besoin [7].

I.4.2. Le sous-ensemble Surveillance

La fonction de surveillance vise à détecter les anomalies, trouver leurs causes et mettre en œuvre des solutions pour rétablir les opérations prévues. Et qui a pour objectifs les principaux suivants :

- Identifier les causes et les conséquences d'actions inattendues ou incontrôlables,
- Développer des solutions pour faire face aux imprévus,
- Travailler avec des opérateurs humains pour prendre des décisions clés, recueillir des informations qui ne sont pas directement disponibles et envisager ou expliquer les solutions correctives mises en œuvre [7].

I.5. Communications fondamentales SCADA

La communication joue un rôle crucial dans les systèmes SCADA. Elle permet l'échange de données entre différents éléments du système, tels que les capteurs, les actionneurs, les contrôleurs logiques programmables, les mainframes, les interfaces utilisateur, …etc. la figure 3 représente quelques bases de communication dans les systèmes SCADA.

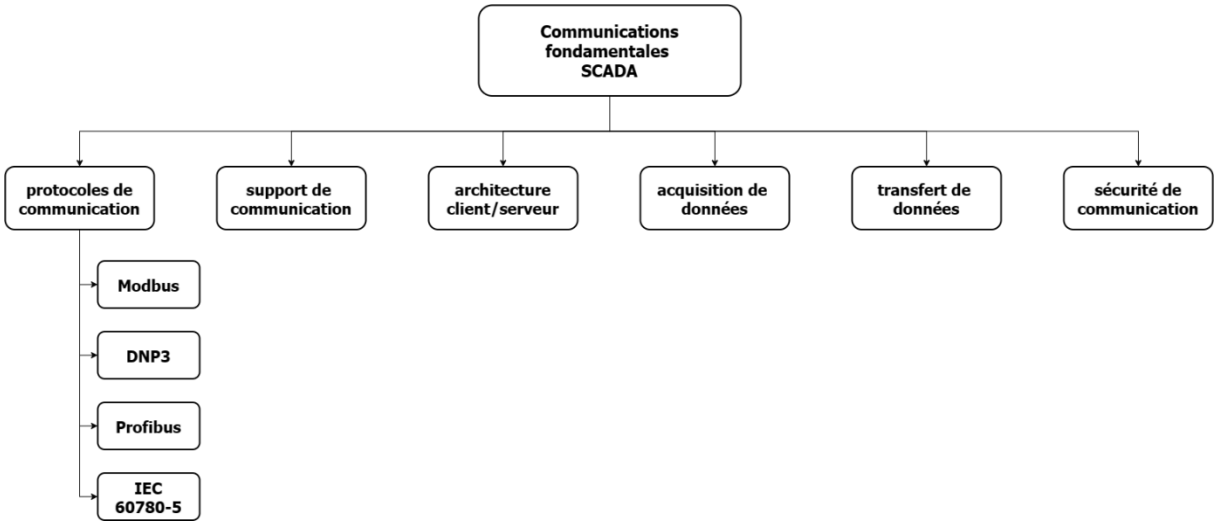

Figure 3: Les éléments de base de la communications fondamentales SCADA.

I.5.1. Protocoles de communication

Les systèmes SCADA utilisent des protocoles de communication spécifiques pour créer des connexions entre les différentes unités du système. Les protocoles couramment utilisés sont Modbus, DNP3, Profibus, et CEI 60870-5.

Modbus

Modbus est un protocole de communication largement utilisé dans les domaines de l'automatisation industrielle et des systèmes de contrôle supervisés (SCADA). Il a été développé dans les années 1970 par Modicon, une société qui fait maintenant partie de Schneider Electric.

Ce protocole de communication est conçu pour permettre l'échange de données entre un dispositif maître (tel qu'un ordinateur central ou un serveur SCADA) et des dispositifs esclaves tels que des capteurs, des actionneurs ou des automates programmables [8]. Une vue détaillée sur Modbus est présentée dans le chapitre II.

DNP3

DNP3 (Distributed Network Protocol Version 3.3) est une norme de télécommunication utilisée pour la communication entre les stations maîtresses, les RTU et d'autres appareils électroniques intelligents. Il garantit l'interopérabilité dans des domaines tels que l'énergie, le pétrole et le gaz, l'eau et les eaux usées, et la sécurité. DNP3 est conçu pour les systèmes SCADA et se concentre sur la transmission fiable de petits paquets de données avec une déterministe séquence d'arrivée. [9].

Profibus

Profibus (Process Field Bus) C'est une norme de réseau en général, utilisé au sein des systèmes de contrôle industriels. Plusieurs systèmes peuvent être utilisés avec la norme Profibus, tels que : le système SCADA, le système numérique de contrôle et de commande, système de contrôle distribué (Distributed Control System DCS), l'automate, l'assemblage et la manipulation des appareils de terrain, le système de contrôle industriel (Industrial Control System ICS) [7].

IEC 60870

La norme IEC 60870 est utilisée dans le domaine de l'électrotechnique et de l'automatisation des centrales électriques pour la commande à distance et l'acquisition de données. Elle permet de contrôler les réseaux de transport et de distribution d'électricité, ainsi que d'autres systèmes de commande à distance. Grâce à l'utilisation de protocoles standard, cette norme assure l'interopérabilité entre les appareils de différents fabricants [10].

Voici ci-dessous le tableau 3 résumant les protocoles cités précédemment.

Tableau 1: Récapitulatif sur les protocoles de SCADA.

Protocol	Description	Application
Modbus	Protocole simple et largement utilisé pour les communications industrielles	Contrôle de processus et appareils industriel
DNP3	Protocole de réseau distribué	Energie, pétrole et gaz, eau et eaux usées et sécurité
Profibus	Protocole de communication de bus de terrain pour l'automatisation industrielle	Automatisation d'usine, contrôle de processus et instrumentation
IEC 60870	Protocole de télécontôle et de téléprotection dans les réseaux électriques	Système d'alimentation électrique

I.5.2. Support de communication

Les systèmes SCADA peuvent utiliser différents types de réseaux pour la transmission de données, tels que les réseaux Ethernet, les réseaux série (RS-232, RS-485), les réseaux cellulaires (GSM, GPRS, LTE), les réseaux sans fil, …etc. Le choix du réseau dépend des exigences spécifiques du système, la distance de communication, la bande passante requise et d'autres facteurs [11].

I.5.3. Architecture client-serveur

Dans les systèmes SCADA, l'architecture client-serveur est couramment utilisée pour la communication. Les ordinateurs centraux ou les serveurs SCADA agissent comme des nœuds centraux, recevant les données des capteurs, envoyant des commandes aux actionneurs et traitant les informations pour l'affichage et l'analyse. Les interfaces utilisateur, les stations de surveillance et les terminaux distants agissent comme des clients qui interagissent avec le serveur SCADA [12].

I.5.4. Acquisition de données

Les systèmes SCADA collectent des données à partir de divers points de surveillance tels que des capteurs, des compteurs et des automates programmables. Ces données sont généralement regroupées en blocs logiques et collectées périodiquement ou en fonction d'événements spécifiques. Les données collectées peuvent inclure des mesures en temps réel, des conditions d'alarme, des événements, des rapports de diagnostic, …etc [13].

I.5.5. Transfert de données

Les données acquises dans les systèmes SCADA sont transférées via les réseaux de communication aux différents nœuds du système. Les données peuvent être transmises de manière synchrone ou asynchrone, en temps réel ou en mode batch. Le transfert des données peut être basé sur des requêtes périodiques, des notifications d'événements ou des demandes spécifiques du système [14].

I.5.6. Sécurité des communications

Les systèmes SCADA doivent disposer de mesures de sécurité pour protéger les communications contre les intrusions, les attaques malveillantes ou les interférences. Cela comprend l'utilisation de mécanismes d'authentification, le cryptage des données, les pares-feux, la segmentation du réseau, les protocoles sécurisé, …etc [15].

Chapitre II Le protocole Modbus

II.1. Introduction

Le protocole Modbus est un standard de communication largement utilisé pour l'échange de données entre les équipements de terrain et les systèmes de contrôle centralisés. Ce chapitre examine les principes essentiels du protocole Modbus, y compris sa structure de messages et les différentes variantes telles que Modbus RTU, Modbus ASCII et Modbus TCP.

II.2. Description générale du protocole Modbus

Depuis 1979, Modbus est devenu le standard de facto pour la communication série dans l'industrie, permettant à des millions de dispositifs d'automatisation de communiquer entre eux. Support de la structure simple et élégante de Modbus ne cesse de croître. La communauté Internet peut accéder à Modbus via le port système réservé 502 de la pile TCP/IP. C'est un protocole de messagerie de couche applicative, positionné au niveau 7 du modèle OSI comme il est montré dans la figure 4. Il permet la communication demande/réponse entre des dispositifs connectés à différents types de séries ou réseaux. Il est actuellement implémenté à l'aide de :

- TCP/IP sur Ethernet,
- La transmission série asynchrone sur différents supports (RS232/422/485, …etc),
- Modbus PLUS, un réseau à passage de jetons à haute vitesse.

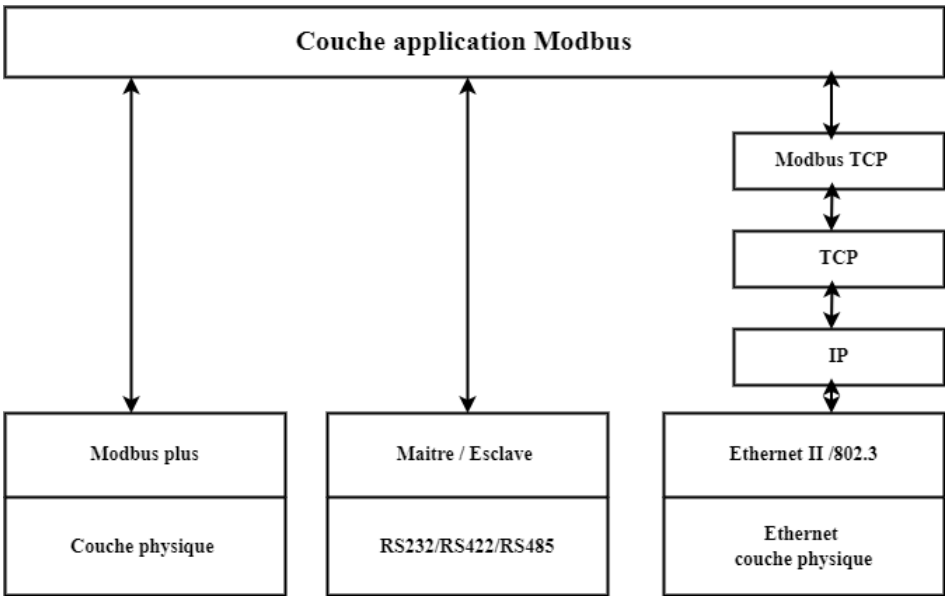

Figure 4: Modèle en couche de communication Modbus [16].

II.3. Architecture du protocole Modbus

Le protocole Modbus permet la communication entre des appareils au sein de différentes infrastructures réseau. Dans l'architecture maître-esclave, l'appareil esclave est connecté au monde physique par le biais de capteurs, d'actionneurs ou d'autres formes de PLC (Programmable Logic Controller). La Figure 5 montre un exemple d'architecture réseau Modbus.

Tous les types de dispositifs sur le terrain, PLCs, IHMs, panneaux de contrôle, pilotes, commandes de mouvement, dispositifs d'E/S, se connectent au même réseau en utilisant différentes implémentations Modbus pour initier une opération à distance. Des passerelles Modbus+ et série sont utilisées comme convertisseurs entre les différents types de bus ou réseaux utilisant le protocole Modbus [16].

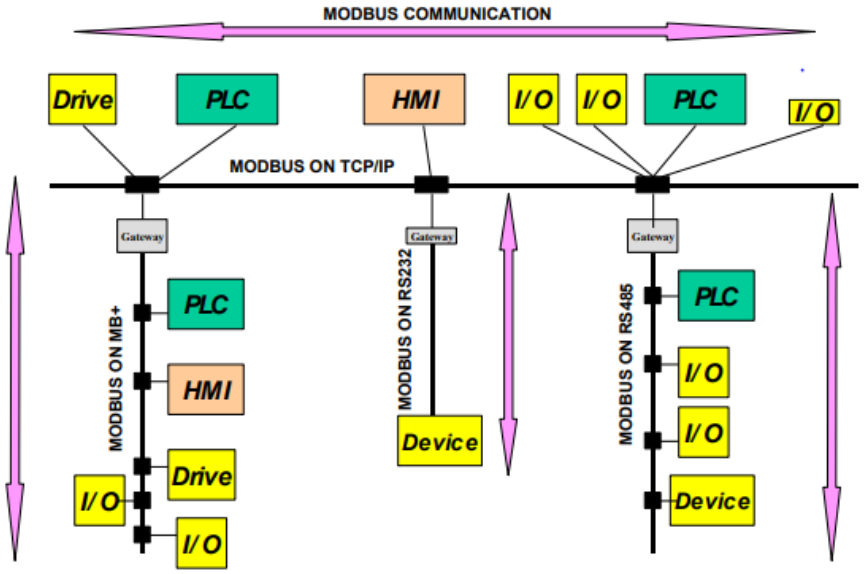

Figure 5: Exemple d'architecture de réseau Modbus [16].

Dans l'environnement SCADA, Modbus est généralement utilisé pour fournir des services de messages de demande et de réponse, permettant au maître d'obtenir des informations sur l'état des appareils esclaves avec un nombre maximal de 274 esclaves.

Au niveau du message, le protocole Modbus continue d'appliquer le principe maître-esclave même si la méthode de communication réseau est pair-à-pair (peer to peer). Si un contrôleur envoie un message, il le fait en tant que périphérique maître et attend une réponse d'un périphérique esclave.

De même, lorsqu'un contrôleur reçoit un message, il construit une réponse en tant que périphérique esclave et la renvoie au contrôleur d'origine.

Tel que représenté sur la figure 6, la communication entre maître/esclave se compose d'un échange de requêtes demande/réponse, avec les deux modes : le mode unicast et le mode broadcast.

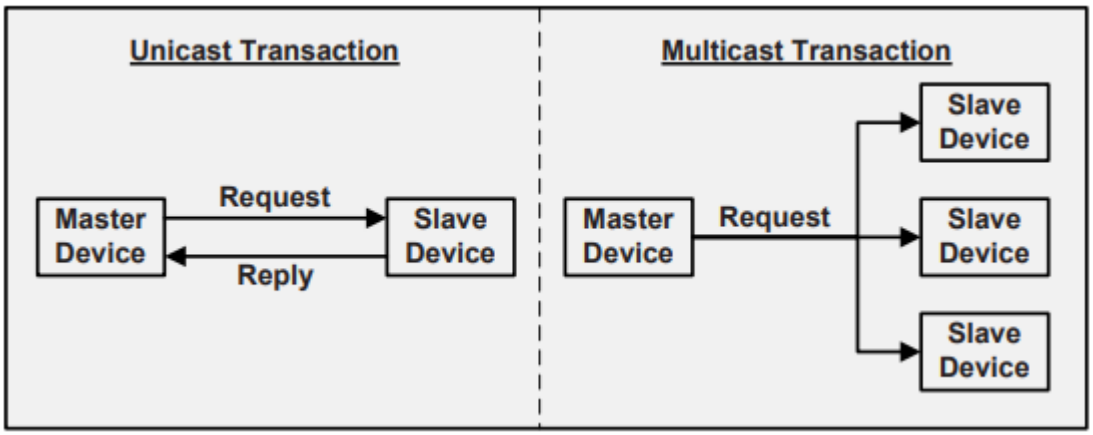

Figure 6:Communications maître-esclave Modbus [17].

II.3.1. Modbus VIA UART

Dans le cadre de la communication du protocole Modbus, l'interface UART (Universal Asynchronous Receiver Transmitter) est utilisée pour la transmission asynchrone des données. Le UART peut être configuré pour prendre en charge divers protocoles de communication, tels que RS232, RS422 et RS485.

RS232(EIA/TIA-232-E)

C'est le plus connu des standards de communication série. Il s'agit d'un système de communication point à point, est utilisé pour une communication full-duplex de données sur de courtes distances, généralement jusqu'à 15 mètres avec vitesse de communication peut atteindre 115 kbits/s. Cependant, le RS232 présente des inconvénients. Il n'est pas adapté aux environnements où il y a beaucoup de bruit ou de parasites, car cela peut perturber la transmission des données [18].

RS422(EIA-422)

C'est une interface série utilise pour les longues distances (1200 mètres), en full-duplex, sa vitesse de transmission peut aller jusqu'à 10 Mbits/s. Les signaux sont envoyés sur 2 fils afin d'augmenter la fréquence de transmission. Il peut supporter jusqu'à 10 récepteurs par ligne

données, mais reste simplement une structure de messagerie indépendante de la couche physique sous-jacente.

RS485(EIA/TIA-485-A)

Une communication série en half-duplex sur de plus longues distances. Il permet de faire communiquer jusqu'à 32 périphériques sur la même ligne de données et sur une distance pouvant aller jusqu'à 1200 m sans répéteurs.

Un message envoyé sur RS232/422/485 se compose d'un bit de démarrage, de plusieurs bits de données, d'un bit de parité et d'un bit d'arrêt.

La figure 7 représente le format de la trame série.

Bit de départ	Données	Parité	Bit de Stop
1 bit	5-8 bits	0-1 bit	1/1.5/2 bit

Figure 7: Format de la trame série.

Le bit de départ est le bit indiquant le début de la transmission, généralement 0.

Bits de données : 5, 6, 7 ou 8 bits de données. Le premier bit est le bit le moins significatif.

Bit de parité : utilisé dans les communications série pour détecter les erreurs de transmission en vérifiant la parité des bits de données.

Bit de Stop : un bit indiquant la fin de la transmission du message, peut prendre les valeurs 1, 1.5 ou 2.

Un exemple des trames est illustré dans la figure 8.

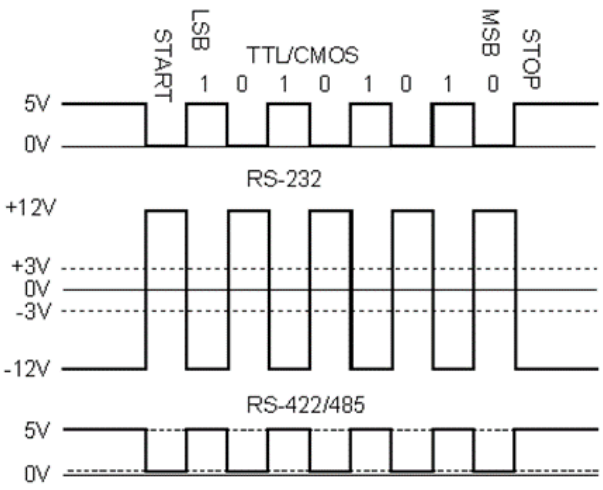

Figure 8: Exemple de donnée transmise dans RS232/422/485.

Sur le bus RS485, le niveau 1 est en bas et le niveau 0 en haut. C'est exactement l'inverse d'une ligne série RS232.

Le start bit est à 1, donc en bas.

Le stop bit est à 0, donc en haut.

Le tableau 2 résume les caractéristiques des support RS232, RS422, RS485

Tableau 2: Caractéristiques des supports physiques.

Port	**RS232**	**RS422**	**RS485**
Type de transfert	Full duplex	Full duplex	Half duplex (2 fils)
Distance maximale	15 mètres à 9600 bps	1200 mètres à 9600 bps	1200 mètres à 9600 bps
Topologie	Point à point	Multipoint	Multipoint
Nombre Max d'appareils connectés	1 récepteur	10 récepteurs	32 récepteurs

II.3.2. MODBUS VIA ETHERNET

Modbus TCP/IP utilise le protocole Ethernet pour la communication sur des réseaux locaux (LAN) ou étendus (WAN). Cela est dû à leur vitesse, leur polyvalence et leur compatibilité avec d'autres

réseaux de niveau professionnel basés sur les mêmes normes de réseau au sein d'un site de production [19].

Les données dans les réseaux Ethernet sont transmises sous forme de trames. Ces trames contiennent les informations à envoyer d'un appareil à un autre via le réseau Ethernet.

Les données sont encapsulées dans la trame Ethernet qui est représentée dans la figure 9, comprend des éléments tels que :

- **Préambule :** Une séquence de 8 octets constituée d'alternances de 0 et de 1. Il permet à l'émetteur et au récepteur de synchroniser leurs horloges.
- **Adresse de destination :** Une adresse MAC de 6 octets qui identifie la carte réseau destinataire.
- **Adresse source :** Une adresse MAC de 6 octets qui identifie la carte réseau émettrice.
- **Type :** Un champ de 2 octets qui spécifie le type de protocole utilisé dans la partie de données de la trame. Pour IPv4 (0x0800) et IPv6 (0x86DD).
- **Data :** La partie des informations transportées par la trame.
- **FCS (Frame Checksum) :** 4 octets qui servent de mécanisme de détection d'erreurs.

Préambule	Adresse de destination	Adresse source	Type	Data	FCS
8 octets	6 octets	6 octets	2 octets	46-1500 octets	4 octets

Figure 9: Trame Ethernet.

II.3.3. Formats de messages Modbus

Le protocole MODBUS définit une unité de données de protocole (PDU) simple, indépendante des couches de communication sous-jacentes. La mise en correspondance du protocole MODBUS sur des bus ou réseaux spécifiques peut introduire certains champs supplémentaires sur l'unité de données d'application (ADU) [16], la figure 10 illustre un format général de la trame Modbus. Modbus définit des règles pour organiser et interpréter les données, mais reste simplement une structure de messagerie indépendante de la couche physique sous-jacente.

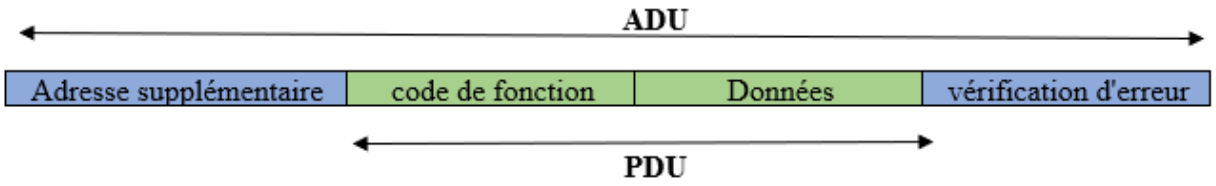

Figure 10:Trame générale de Modbus.

II.3.4. Types des messages Modbus

Le dispositif maître qui initie la transaction construit l'ADU. Une fois que le maître envoie une demande en unicast ou broadcast, les esclaves répondent individuellement à chaque demande qui leur est adressée, sans initier de messages par eux-mêmes.

La demande du maître comprend une adresse, un code de fonction qui définit l'action demandée, données requises et un champ de vérification d'erreur. Le champ de données contient des informations supplémentaires utilisées par le serveur ou l'esclave pour exécuter l'action spécifiée par le code de fonction [16].

D'autre part, la réponse de l'esclave comprend des champs confirmant l'action effectuée, les données à renvoyer et un champ de vérification d'erreur. En cas d'absence d'erreur, la réponse de l'esclave contient les données demandées, mais en cas d'erreur dans la demande reçue ou si l'esclave est incapable d'effectuer l'action demandé [16].

II.3.5. Transaction MODBUS

Les opérations suivantes sont effectuées dans cette procédure [18]:

- **Validation du code de fonction :** Vérifie si un code de fonction valide a été reçu et s'il a été configuré pour effectuer l'action demandée. En cas d'erreur, une exception est renvoyée.
- **Validation de l'adresse de données :** Les données demandées du champ de données de l'unité de données de protocole (PDU) doivent être valides selon la carte mémoire du périphérique. En cas d'erreur, un code d'exception est renvoyé à l'expéditeur.
- **Validation de la valeur des données :** Vérifie la validité de la valeur des données pour déterminer si l'action demandée peut être effectuée. En cas de demande d'écriture, il vérifie

si la valeur des données peut être écrite avant d'exécuter l'action demandée. En cas d'échec de cette vérification, une exception est renvoyée à l'expéditeur.

- **Exécution de la fonction :** Après les vérifications préliminaires, le périphérique exécute l'action demandée en utilisant les données fournies dans l'unité de données de protocole (PDU). En cas d'erreur, plusieurs exceptions peuvent être renvoyées.
- **Envoi de la réponse Modbus :** En cas d'exécution réussie de la commande, une réponse est générée et envoyée au périphérique demandeur.

Les figures 11 et 12 montrent la transaction Modbus lorsqu'il n'y a pas d'erreur et lorsqu'une exception se produit en raison d'une erreur.

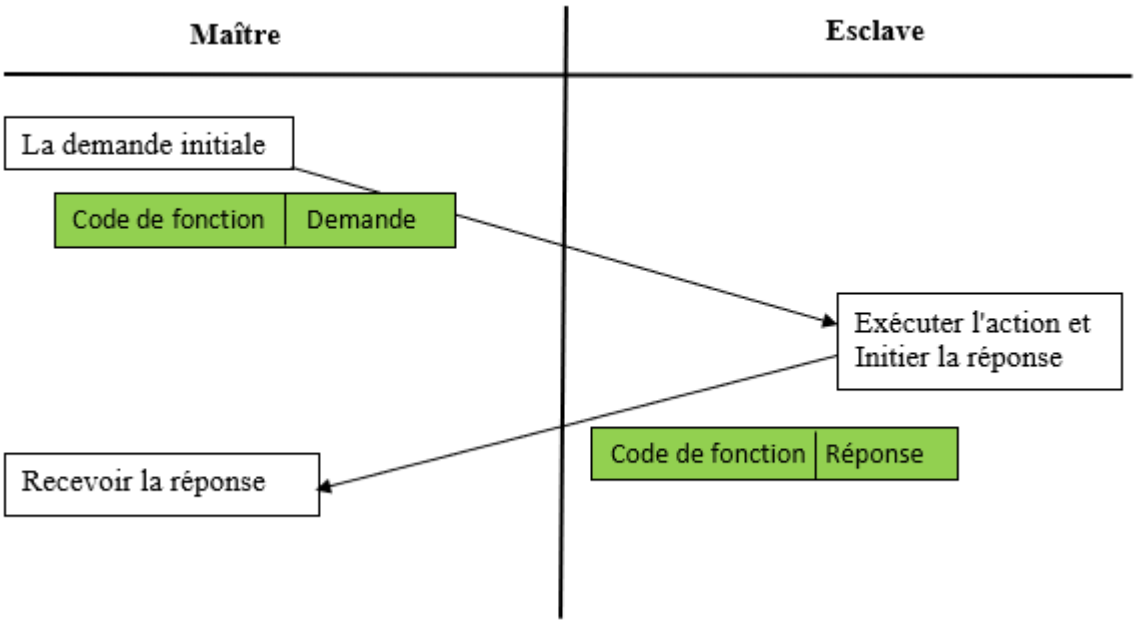

Figure 11:Transaction de Modbus (sans erreur).

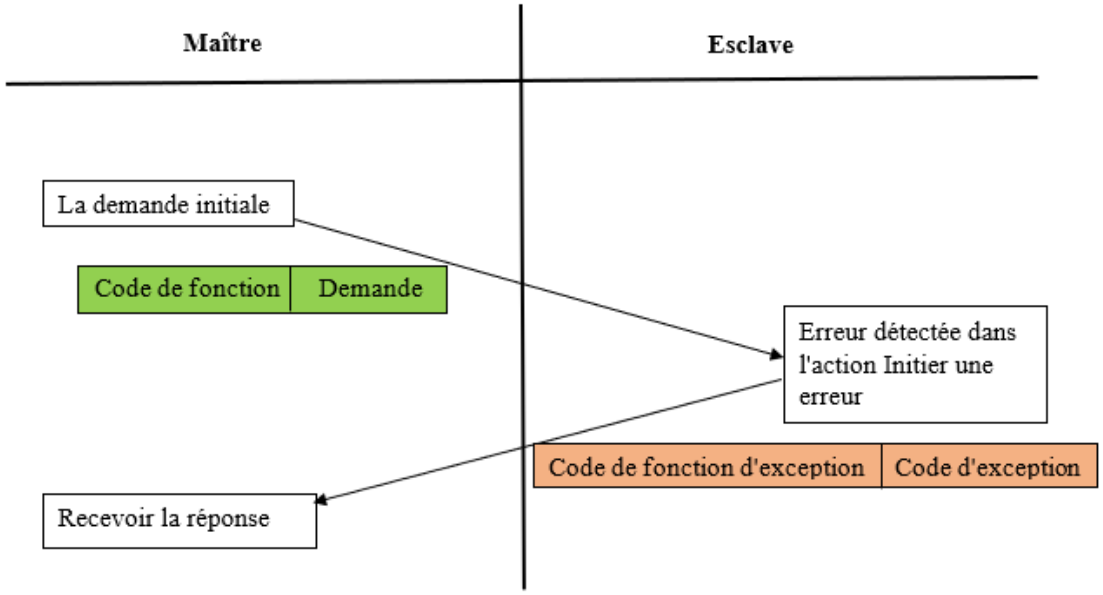

Figure 12:Transaction de Modbus (réponse avec exception).

II.3.6. Codes de fonction MODBUS

Le protocole Modbus fonctionne avec des codes de fonction qui sont envoyés entre le maître et l'esclave. Chaque code de fonction détermine quelle action doit être exécutée. Le tableau 3 présente les codes de fonction utilisés par Modbus [16].

Tableau 3: Les codes de fonction de Modbus. Voir Annexe A.

Code de fonction	Description
01 (0x01)	Lecture de bobines (bits)
02 (0x02)	Lecture d'entrées discrètes (bits)
03 (0x03)	Lecture de registres de données (16 bits)
04 (0x04)	Lecture d'entrées analogiques (16 bits)
05 (0x05)	Écriture d'une seule bobine (bit)
06 (0x06)	Écriture dans un seul registre (16 bits)
15 (0x0F)	Écriture de plusieurs bobines (bits)
16 (0x10)	Écriture de plusieurs registres (16 bits)
23 (0x17)	Lecture/écriture de registres
43 (0x2B)	Lecture/écriture de plusieurs registres

II.3.7. Le modèle des données

Modbus utilise une représentation "big-Endian" pour les adresses et les éléments de données. Cela signifie que lorsqu'une quantité numérique plus grande qu'un seul octet est transmise, l'octet le plus significatif est envoyé en premier [16].

Le modèle de données Modbus est basé sur quatre types de données différents : discrete inputs, coils, input registers and holding registers, qui sont présentés dans le tableau 4.

Tableau 4 : Les quatre types de registre utilisés pour stocker les informations dans Modbus.

Type de registre	Taille	L'accès	Commentaires
Discretes Input	Un bit	Lecture seule	Ce type de données peut être fourni par un système d'E/S (entrée/sortie).
Coils	Un bit	Lecture-écriture	Ce type de données peut être modifié par un programme d'application.
Input Registers	Mot de 16 bits	Lecture seule	Ce type de données peut être fourni par un système d'E/S.
Holding Registers	Mot de 16 bits	Lecture-écriture	Ce type de données peut être modifié par un programme d'application.

Chacun des quatre types de données à une pile de mémoire de registres spécifiée et 10000 adresses désignées. Comme montré dans le tableau 5.

Tableau 5: Plage d'adresses des registres pour les types de registres Modbus.

Type de register	Plage d'adresses
Coils	1-9999
Discrete Inputs	10000-19999
Input Registers	30000-39999
Holding Registers	40000-49999

II.4. Variantes de Modbus

Les contrôleurs peuvent être configurés pour communiquer sur des réseaux Modbus standard en utilisant Modbus RTU, qui prend en charge les transmissions sur les bus série. Ou bien Modbus TCP est une variante de Modbus développée pour fonctionner sur des réseaux modernes utilisant l'IP [19].

II.4.1. Modbus RTU

Modbus RTU (Remote Terminal Unit) il s'agit d'un protocole ouvert qui est compatible avec la vaste majorité des appareils industriels, il est utilisé pour la communication série. Modbus RTU fonctionne sur la couche Data Link du modèle OSI, tandis que les couches physiques utilisent RS-485 et RS-232. Le format de trame Modbus RTU comprend l'Adresse d'esclave, le code de fonction, les données, le contrôle de redondance cyclique (CRC) [20], comme il est montré dans la figure 13.

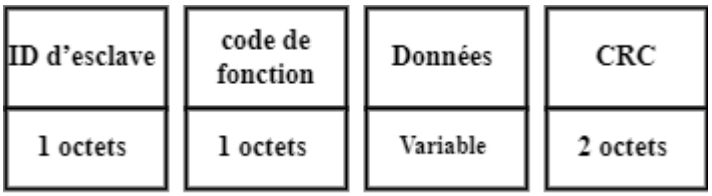

Figure 13:Trame Modbus RTU.

Une trame Modbus RTU est composée des éléments suivants :

- Adresse d'esclave : Un octet qui identifie l'appareil esclave auquel la trame est destinée. Les adresses d'esclave vont généralement de 1 à 247.
- Code de fonction : Un octet qui spécifie le type d'action ou d'opération à effectuer sur l'appareil esclave. Il peut s'agir de lectures, d'écritures, de demandes de statut,… etc. Les codes de fonction vont de 1 à 255, bien que certains codes soient réservés à des fins spécifiques.
- Données : Les données transportées par la trame varient en fonction du code de fonction utilisé. Elles peuvent inclure des adresses de registres, des valeurs à écrire, des données de statut, …etc.

- Contrôle de parité : Un ou deux octets utilisés pour la détection d'erreurs et la vérification de l'intégrité des données. Il peut s'agir d'une parité paire ou impaire, ou d'une somme de contrôle de trame.

II.4.2. Modbus TCP

Modbus TCP est un protocole Modbus qui utilise la communication TCP/IP via EthernetII.

Sa trame encapsule l'entête protocole d'application Modbus (MBAP) et l'unité de données de protocole (PDU) (figure14) dans un segment TCP sans l'octet somme de contrôle, en utilisant un port de communication 502 [20]. Dans Modbus TCP, le modèle de communication utilisé est client/serveur.

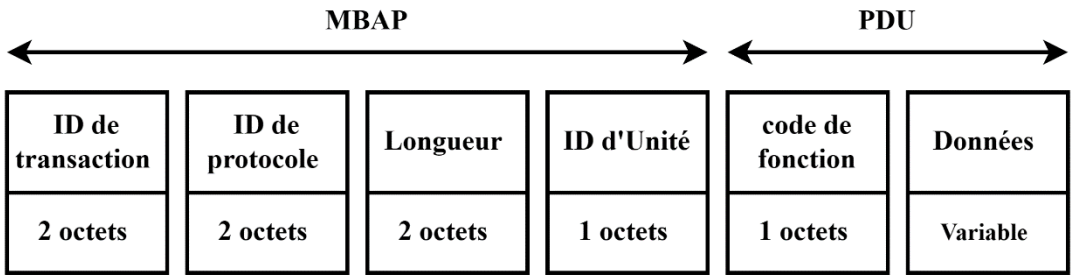

Figure 14:Trame Modbus TCP.

Une brève description des champs de la trame Modbus TCP est représentée dans le tableau 6.

Tableau 6: La structure de Modbus TCP ADU [21].

Champs	Longueur	Description	Client	Serveur
ID de transaction	2 octets	Identification d'une transaction de demande/réponse	Initialisé par le client	Répété par le serveur à partir de la demande reçue ; renvoyé dans la réponse

ID de protocole	2 octets	0 = Protocole Modbus	Initialisé par le client	Répété par le serveur à partir de la demande reçue ; renvoyé dans la réponse
Longueur	2 octets	Nombre d'octets suivants dans l'APDU	Initialisé par le client basé sur la demande	Initialisé par le serveur basé sur la réponse
ID d'esclave	1 octet	Identifie l'esclave associé à la transaction	Initialisé par le client	Initialisé par le serveur
Code de fonction	1 octet	Indique le type d'action à effectuer dans la demande	Spécifié par le client	Interprété par le serveur
Data	Variable	Données spécifiques à la fonction Modbus	Incluses dans la demande du client	Incluses dans la réponse du serveur

Chapitre III Passerelle Modbus RTU / Modbus TCP

III.1. Introduction

Dans ce chapitre nous abordons l'aspect pratique de notre passerelle, qui consiste à mettre en place une solution assure la communication entre les deux protocoles Modbus RTU et Modbus TCP. Nous présentons d'abord les outils nécessaires à cette implémentation (hard et soft). Nous exposons ensuite les étapes de programmation, maitre/esclave pour Modbus RTU et client/serveur pour Modbus TCP. Enfin nous présentons les résultats des tests sur la passerelle pour vérifier son bon fonctionnement et son efficacité dans la communication entre les deux protocoles.

III.2. Solution proposée

Le Schéma de conception générale de notre passerelle est représenté dans la figure 15. Cette solution est implémentée sur un carte Raspberry Pi. Elle est programmée sous environnement Python.

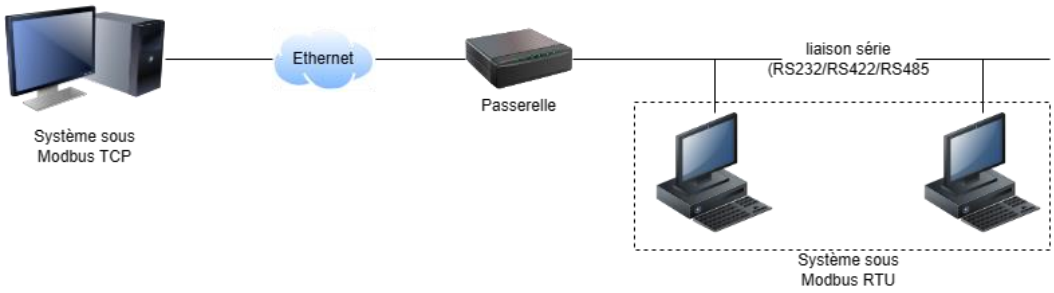

Figure 15:Schéma de conception générale.

Le rôle principal de cette passerelle et d'agir comme un pont entre les deux systèmes, permettant aux appareils qui communiquent via le protocole série Modbus RTU d'interagir avec le réseau Modbus TCP/IP.

Bien qu'il existe des produits répondant à cette exigence [22], notre passerelle personnalisée implémentée sur une carte Raspberry Pi offre une solution plus économique et flexible. Nous avons développé notre propre passerelle avec des capacités de manipulation de code étendues, permettant aux utilisateurs de l'adapter et de la personnaliser selon leurs besoins spécifiques.

III.3. Matériel utilisé

Pour la réalisation de la passerelle Modbus RTU/TCP, nous avons utilisé le matériel suivant :

III.3.1. Carte Raspberry Pi

Le Raspberry Pi est un ordinateur de la taille d'une carte de crédit avec un processeur ARM qui fonctionne sous environnement Linux. Dans notre projet, il s'agit de Raspberry Pi 3 Modèle B.

comme illustré dans la Figure 16 ci-dessous. Elle présente les propriétés suivantes [23] :

1. Processus intégré : 2 GHz quad-core BCM2837 ARMv8 64bit CPU, 1G RAM,
2. Lecteur de cartes Micro SD : qui se trouve sous le bus d'afficheur LCD,
3. Port HDMI : « High Definition Multimedia Interface » permet de relier le RaspberryPI à un dispositif compatible,
4. 4 Ports USB 2.0,
5. Port Ethernet,
6. Sortie audio,
7. Sortie vidéo,
8. 40 broches GPIO,
9. Alimentation : tension 5V, courant minimal 750mA,
10. Bus afficheur LCD.

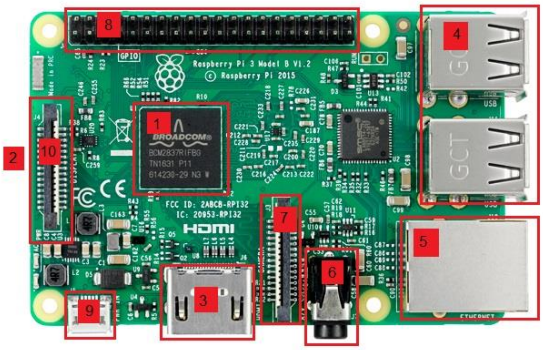

Figure 16:Les composants essentiels de la carte Raspberry pi 3 [23].

Ses caractéristiques répondent aux exigences d'implémentation de notre solution. Le processeur Raspberry Pi joue le rôle de contrôleur dans la passerelle, garantissant une gestion et un contrôle efficaces des données et des dispositifs.

III.3.2. VNC Viewer

VNC est une technologie qui permet d'accéder et de contrôler à distance un autre ordinateur via un réseau [24]. Il permet de visualiser et d'interagir avec cet ordinateur. VNC fonctionne en transmettant l'interface utilisateur graphique (GUI) de l'ordinateur distant via le réseau vers

l'appareil client, où l'utilisateur peut voir et contrôler le bureau distant. La figure 17 représente la fenêtre d'accueil de logiciel VNC Viewer.

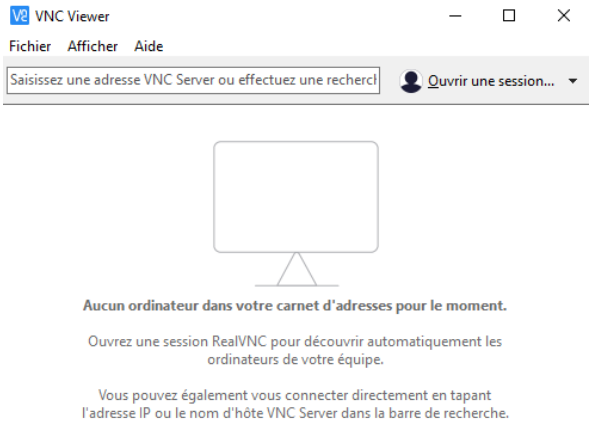

Figure 17: fenêtre d'accueil VNC Viewer.

III.3.3. PuTTy

PuTTy est un émulateur de terminal, une application de transfert de fichiers en réseau et un client open source. Il permet aux utilisateurs de se connecter à un serveur ou à un appareil distant. Il prend également en charge diverses fonctionnalités telles que la gestion de session, l'authentification par clé SSH et la personnalisation du terminal [25]. La figure 18 montre fenêtre d'accueil de logiciel PuTTy.

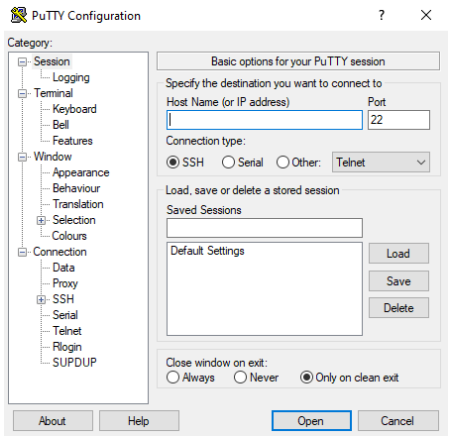

Figure 18: fenêtre d'accueil Logiciel PuTTy.

III.4. Configuration de Raspberry Pi

La configuration d'une carte Raspberry Pi implique plusieurs étapes, notamment l'installation du système d'exploitation, la connexion des périphériques nécessaires et la configuration du réseau. Ces étapes sont présentées en détails dans l'annexe C.

III.5. Langage de programmation utilisé

Pour ce projet, nous avons utilisé le langage Python, qui a la réputation d'être un langage convivial pour les débutants, car il gère une complexité utilisateur importante et permet aux débutants de se concentrer sur la compréhension complète des concepts de programmation plutôt que sur les moindres détails.

La Fondation Raspberry Pi a spécifiquement choisi Python comme langage principal en raison de :

- Sa polyvalence,
- Ses syntaxes simple et facile à apprendre,
- Multitude de ressources et de support disponibles pour son utilisation,
- Disposition d'un riche ensemble de bibliothèques et d'outils qui peuvent être utilisés pour simplifier des tâches complexes [26].

En ce qui concerne Raspberry Pi, Python est préinstallé sur le système d'exploitation Raspbian.

III.5.1. Les bibliothèques utilisées

Dans nos codes Python, nous avons utilisé plusieurs bibliothèques pour la mise en œuvre de la communication Modbus et le développement de l'interface utilisateur. Voici une brève description de ces bibliothèques :

- **PySerial** : permet la communication série,
- **Socket** : permet la communication réseau,
- **UModbus** : fournit des fonctionnalités pour la mise en œuvre du protocole Modbus,
- **Modbus_tk** : offre une interface de programmation pour communiquer avec des périphériques Modbus,
- **PyQt** : permet de créer des interfaces graphiques,
- **Tkinter** : Tkinter est une bibliothèque Python standard pour le développement d'interfaces graphiques,

- **pyCrypto ou pyCryptodome** : une bibliothèque Python utilisée pour le chiffrement et le déchiffrement de données.

III.6. Conception de l'architecture de la passerelle

La conception d'une passerelle Modbus RTU vers Modbus TCP nécessite une réflexion approfondie sur différents aspects de la conception afin d'assurer une communication fluide entre les appareils Modbus RTU et les réseaux Modbus TCP/IP. La figure 19 illustre un schéma de la passerelle et les deux réseaux RTU et TCP.

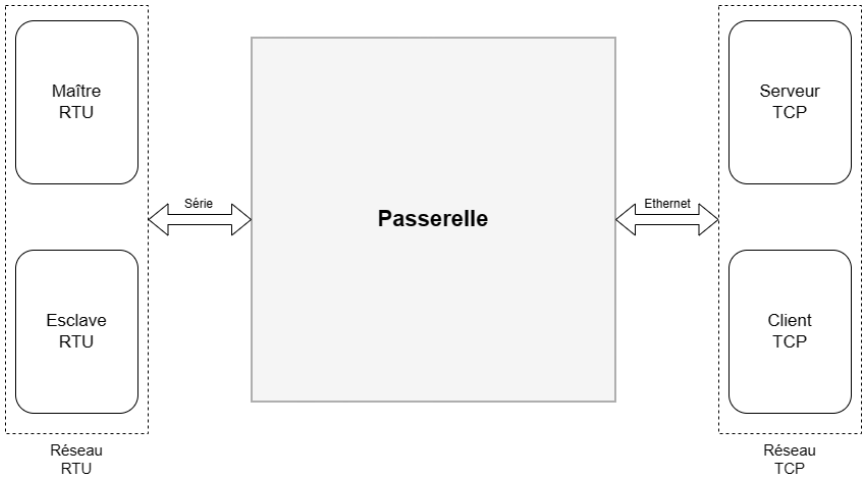

Figure 19:Schéma de la passerelle et les deux réseaux RTU & TCP.

III.6.1. Principe de fonctionnement

Le fonctionnement principal de la passerelle consiste à recevoir des données d'un réseau, les convertir dans le protocole approprié, puis les transmettre à l'autre réseau. Par exemple, Dans le cas d'une passerelle Modbus RTU vers Modbus TCP, elle reçoit les données de l'appareil Modbus RTU via le réseau série, les convertit dans le format Modbus TCP, puis les envoie à l'appareil Modbus TCP via le réseau Ethernet.

III.6.2. Réalisation de la passerelle

Dans cette partie, nous allons aborder la réalisation de la passerelle Modbus RTU/TCP en utilisant les outils et les langages de programmation présentés précédemment. Nous détaillerons les différentes étapes nécessaires à la mise en place de la passerelle.

Bibliothèques utilisées

Dans la passerelle on a utilisé quatre bibliothèques, tel quelles : Socket, PySerial, Modbus-tk, Tkinter.

Configuration des ports de communication

On configure le port série, en utilisant cette syntaxe :

```
serial.Serial('COM1', baudrate=9600, bytesize=8, parity='N', xonxoff=0)
```

Et pour le port TCP, on utilise cette syntaxe :

```
sock= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

L'argument *socket.AF_INET* , spécifie la famille d'adresses à utiliser, dans ce cas, le schéma d'adressage IPv4.

Le deuxième argument *socket.SOCK_STREAM*, spécifie le type de socket à créer. Dans ce cas, SOCK_STREAM indique que la socket sera une socket de type flux (stream), utilisée pour les connexions TCP.

Traitement des données

La passerelle a deux parties de traitement de données, elles sont présentées comme suit :

Requête Maître RTU

Lorsque le maître Modbus RTU envoie une requête à la passerelle, celle-ci convertit cette requête en une requête Modbus TCP en extrant PDU de ADU, ensuite Ajoutant l'entête MBAP à cette PDU. La passerelle transmet ensuite cette requête au Serveur Modbus TCP correspondant. Le Serveur traite la requête et renvoie une réponse à la passerelle. La passerelle reçoit alors la réponse du Serveur Modbus TCP, une autre opération d'extraction de la PDU est effectuée, ensuite l'adresse ou identité d'esclave correspond et le CRC sont ajoutés à cette PDU, Finalement, la réponse est envoyée au maître Modbus RTU.la figure 20 résume cette explication.

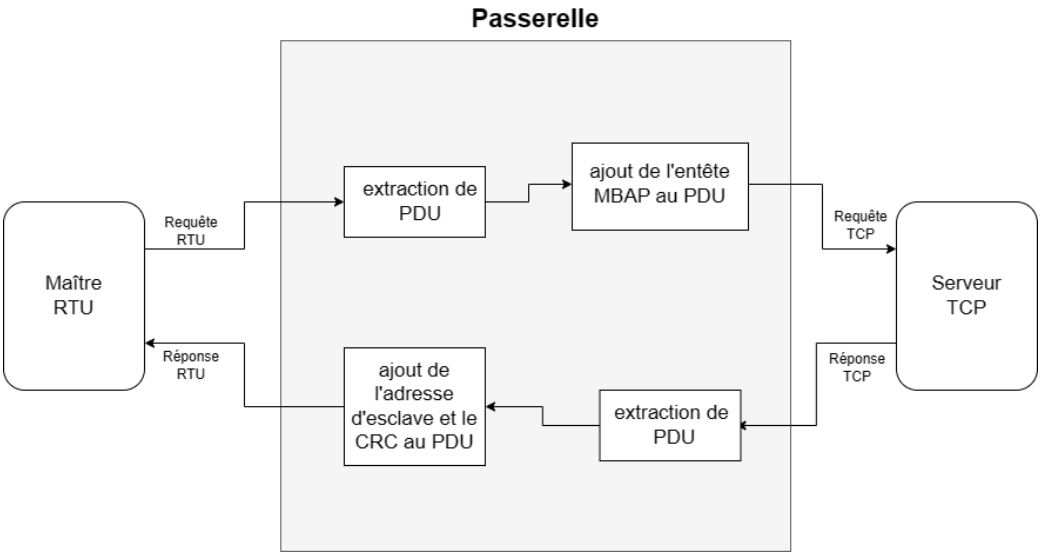

Figure 20:Chemin de la requête RTU et la réponse TCP.

Requête client TCP

Lorsque le client Modbus TCP envoie une requête à la passerelle, celle-ci convertit cette requête en une requête Modbus RTU en extrant la PDU de ADU, ensuite Ajoutant l'adresse ou identité d'esclave correspond et le CRC à la PDU. La passerelle transmet ensuite cette requête à l'esclave Modbus RTU correspondant. L'esclave traite la requête et renvoie une réponse à la passerelle. La passerelle reçoit alors la réponse de l'esclave Modbus RTU, une autre opération d'extraction de la PDU est effectuée, ensuite l'entête MBAP est ajoutée à cette PDU, Finalement, la réponse est envoyée au client Modbus TCP. La figure 21 représente cette explication.

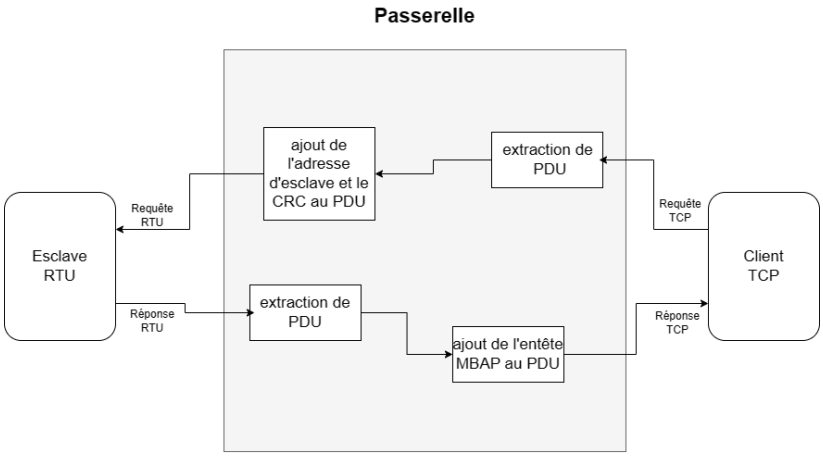

Figure 21:Chemin de la requête TCP et la réponse RTU.

III.7. Mise en œuvre de la communication Modbus RTU

L'implémentation du maître et esclave Modbus RTU a été réalisée en utilisant la bibliothèque PySerial pour la communication série et la bibliothèque Modbus_tk pour gérer le protocole Modbus RTU.

Pour le maître Modbus RTU, les étapes suivantes ont été suivies pour assurer son bon fonctionnement :

- Importation des bibliothèques requises : Les bibliothèques PySerial et Modbus_tk ont été importées dans le code du maître,
- Configuration du port série : Les paramètres du port série tels que le nom du port, le débit en bauds, la parité, …etc. ont été définis en utilisant la classe Serial de PySerial,
- Gestion des requêtes : Le code a été conçu pour gérer les requêtes Modbus RTU entrantes. Cela impliquait la réception des requêtes via le port série, l'extraction des informations nécessaires (adresse de l'esclave, code de fonction, adresse du registre), et l'utilisation de la bibliothèque Modbus_tk pour générer les réponses appropriées.

De même, pour l'esclave Modbus RTU, les étapes suivantes ont été suivies :

- Importation des bibliothèques requises : Les bibliothèques PySerial et Modbus_tk ont été importées dans le code d'esclave,
- Configuration du port série : Les paramètres du port série ont été définis en utilisant la classe Serial de PySerial, de manière similaire à la configuration du maître,
- Envoi des requêtes : Le code du client a été développé pour envoyer des requêtes Modbus RTU au maître via le port série. Les requêtes ont été construites en fournissant les informations nécessaires telles que l'adresse de l'esclave, le code de fonction et l'adresse du registre, en utilisant la bibliothèque Modbus_tk,
- Réception des réponses : L'esclave a été configuré pour recevoir les réponses du maître Modbus RTU via le port série.

Test de communication

Un test de communication a été effectué pour vérifier le bon fonctionnement des deux codes maître et esclave Modbus RTU, ce test est fait en capturant et en analysant les paquets échangés entre le maître et l'esclave Modbus RTU à l'aide d'outils appropriés tels que Modbus poll qui agit comme un maître RTU et Modbus Slave. Il était possible de vérifier la validité des requêtes et des réponses Modbus RTU, assurant ainsi une transmission précise des données.

La figure 22 représente la transmission des données entre le code esclave Modbus RTU développer et le logiciel Modbus Poll.

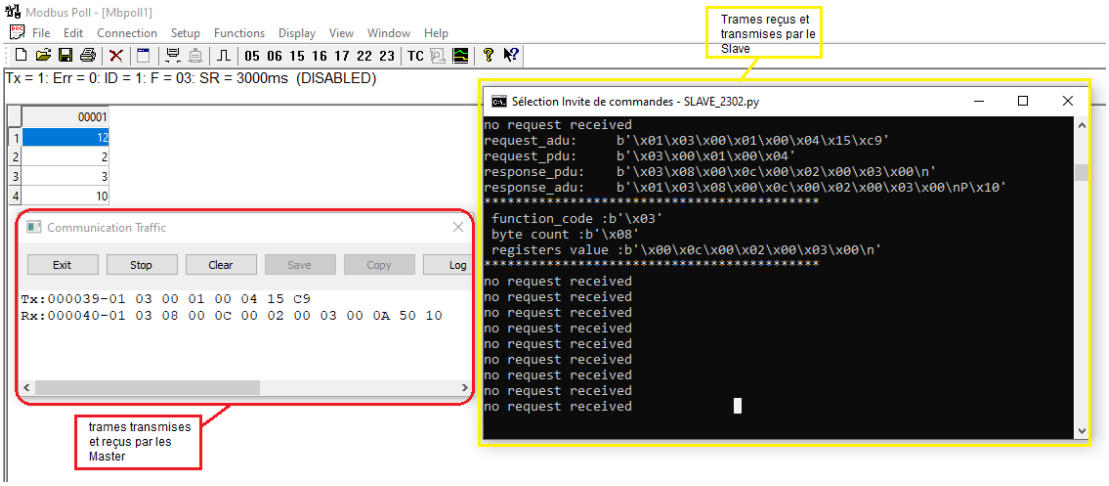

Figure 22:Capture Modbus poll et notre code esclave.

La figure 23 montre la transmission des trames entre le code maître Modbus RTU et le logiciel Modbus Slave.

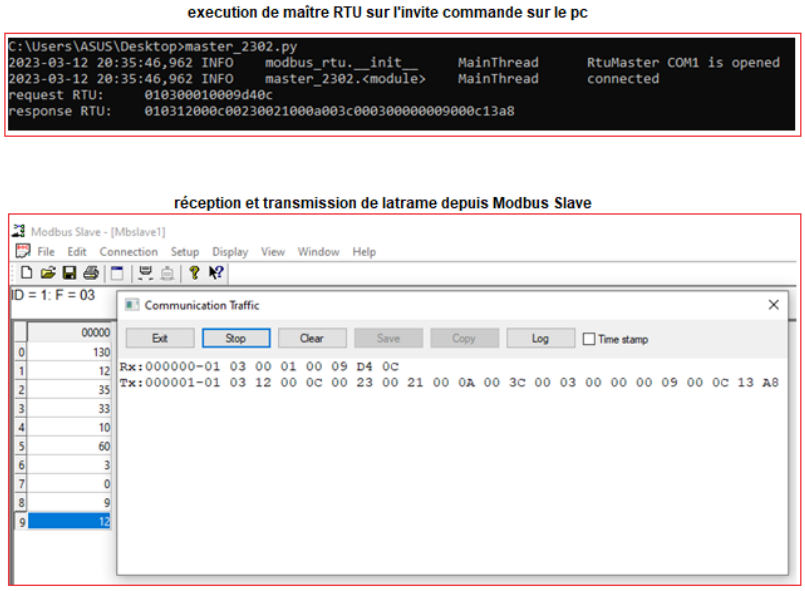

Figure 23:Capture Modbus Slave et notre code maître

III.8. Mise en œuvre de la communication Modbus TCP

La mise en œuvre du client et du serveur Modbus TCP a été réalisée en utilisant la bibliothèque uModbus pour la gestion des requêtes et des réponses Modbus TCP, ainsi que le module socket intégré à Python pour la communication réseau.

Pour commencer, le serveur Modbus TCP a été développé en utilisant le module socket et la bibliothèque uModbus. Les étapes suivantes ont été suivies pour assurer le bon fonctionnement du serveur :

- Importation des modules : Les modules socket et uModbus ont été importés dans le code du serveur,
- Configuration du serveur : Le serveur a été configuré en créant une instance de socket et en liant l'adresse IP et le port de communication sur lesquels il écoute les requêtes des clients Modbus TCP. Les paramètres Modbus tels que l'adresse et les registres ont également été définis en fonction des besoins spécifiques du système,
- Gestion des requêtes : Le code du serveur a été mis en place pour gérer les requêtes entrantes des clients Modbus TCP. Cela implique la réception des requêtes via le socket, l'extraction des informations nécessaires (adresse, registres,… etc.) et l'utilisation de la bibliothèque uModbus pour générer les réponses appropriées.

De même, le client Modbus TCP a été créé en utilisant le module socket et la bibliothèque uModbus. Les étapes suivantes ont été suivies pour assurer le bon fonctionnement du client :

- Importation des modules : Les modules socket et uModbus ont été importés dans le code du client,
- Configuration du client : Le client a été configuré en créant une instance de socket et en spécifiant l'adresse IP et le port de communication du serveur auquel il se connecte. Les paramètres Modbus, tels que l'adresse et les registres, ont également été définis en fonction des besoins spécifiques du système,
- Envoi de requêtes : Le code du client a été développé pour envoyer des requêtes Modbus TCP au serveur via le socket. Les requêtes ont été construites en fournissant les informations nécessaires, telles que l'adresse de l'esclave, le type de fonction Modbus et les registres associés, en utilisant la bibliothèque uModbus,
- Réception des réponses : Le client a été configuré pour recevoir les réponses du serveur Modbus TCP via le socket. Les réponses ont été extraites en utilisant la bibliothèque uModbus pour obtenir les données requises ou les messages d'erreur, le cas échéant.

Test de communication

On utilise Wireshark pour capturer et analyser les paquets échangés entre le serveur et le client. Cela a permis de vérifier si les requêtes et les réponses Modbus TCP étaient correctement formées et si les données étaient transmises comme prévu.

Nous avons établi la communication entre le Client et le serveur Modbus TCP. La figure 24 représente la trame requête que le client a envoyé, et qui a été capturé par le logiciel Wireshark.

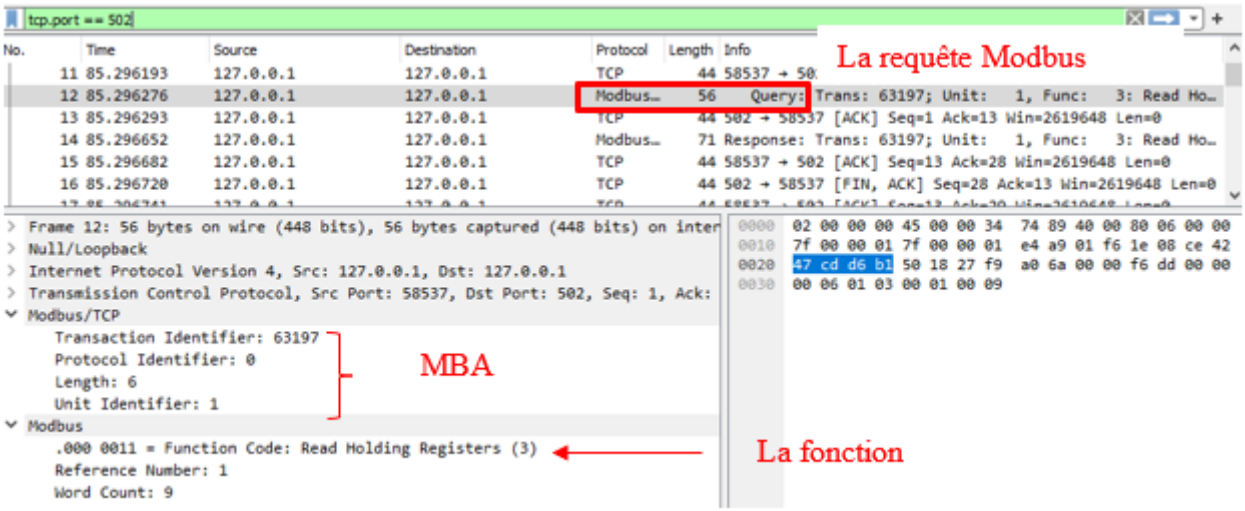

Figure 24:La trame requête capturée par le Wireshark.

La figure 25 représente la trame réponse que le serveur a envoyée, et qui a été capturé par le logiciel Wireshark.

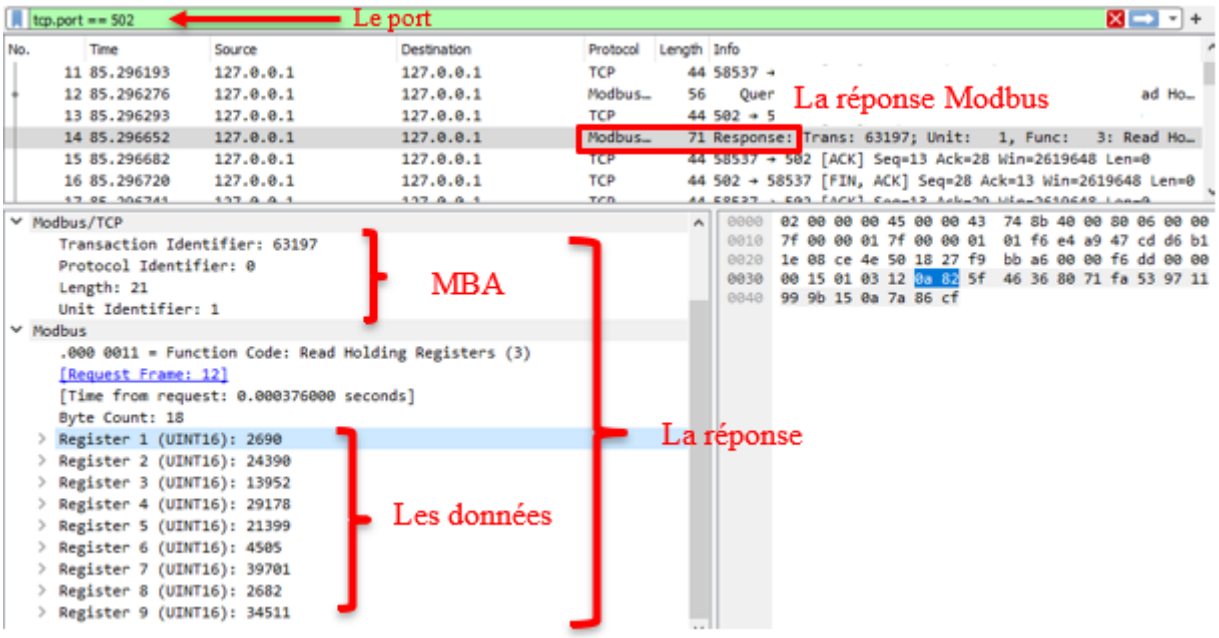

Figure 25:La trame de la réponse capturée par le Wireshark.

III.9. Mesure de sécurité

Pour assurer la sécurité de la passerelle et la communication Modbus, nous avons utilisé la bibliothèque AES (Advanced Encryption Standard) .

AES (Advanced Encryption Standard) est un algorithme de chiffrement symétrique largement utilisé dans le domaine de la sécurité informatique. Il offre une forte protection des données en utilisant des clés de chiffrement de 128, 192 ou 256 bits. L'AES est reconnu pour sa robustesse, sa rapidité et son adoption répandue dans les applications nécessitant une confidentialité et une intégrité des données élevées.

Voici les éléments clés à aborder concernant l'utilisation de l'AES dans la sécurité :

- **Clés de chiffrement :** Le chiffrement AES nécessite l'utilisation d'une clé de chiffrement. Nous avons généré une clé de chiffrement AES de manière sécurisée. La clé est un élément essentiel du processus de chiffrement et doit être suffisamment longue et aléatoire pour garantir la sécurité,
- **Création du chiffreur (Cipher) :** À l'aide de la clé générée et un mode de chiffrement tel que le mode CBC (Cipher Block Chaining) pour garantir l'intégrité et la sécurité des données, nous avons créé un objet chiffreur (cipher) AES. Cet objet chiffreur est responsable de l'encodage et du décodage des données,

- **Chiffrement des données :** Avant d'envoyer des données depuis la passerelle Modbus TCP vers le Modbus RTU, nous avons utilisé le chiffreur AES pour chiffrer les données. Le chiffrement AES assure que les données sont illisibles sans la clé appropriée,
- **Décryptage des données :** Lors de la réception des données du Modbus RTU vers la passerelle Modbus TCP, nous avons utilisé le même chiffreur AES pour décrypter les données. Cela permet de retrouver les données d'origine dans un format lisible.

L'utilisation de l'AES pour chiffrer et déchiffrer les données assure la confidentialité des informations sensibles transitant entre la passerelle et les dispositifs Modbus.

Lorsqu'on envoie une trame cryptée par AES et la capture à l'aide de Wireshark, les données apparaîtront comme étant chiffrées et le protocole utilisé sera TCP plutôt que Modbus TCP. Cela est dû au fait que Wireshark n'a pas la capacité de déchiffrer les données cryptées comme il est présenté dans la figure 27.

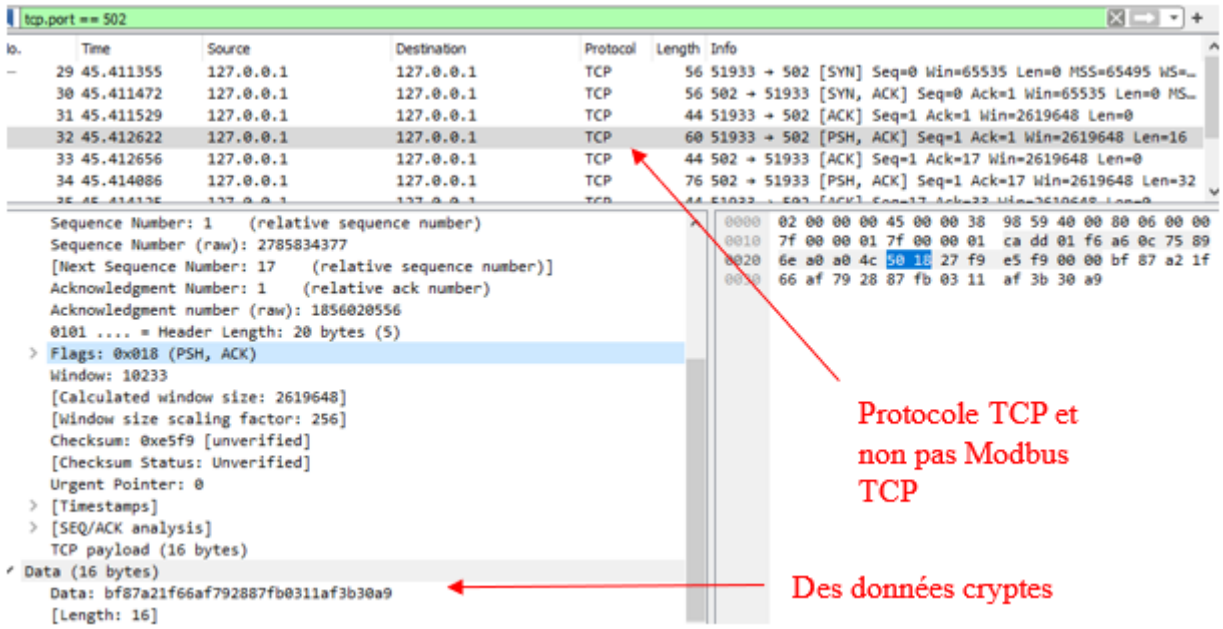

Figure 26:Une trame cryptée capturé par le Wireshark.

III.10. Test et résultats

Pour valider l'implémentation de notre passerelle sécurisée, nous avons effectué des tests entre PC1, la carte Raspberry Pi et PC2. Ces tests ont été réalisés pour vérifier le bon fonctionnement de la passerelle, la sécurité des échanges de données et les performances du système. De plus, nous avons ajouté des fonctionnalités graphiques GUI (Graphical User Interface) pour faciliter la lecture, la visualisation et la gestion des données échangées. Ces interfaces offrent une expérience utilisateur intuitive, permettant aux utilisateurs de superviser et de contrôler efficacement les communications entre les systèmes Modbus RTU et TCP.

III.10.1. Configuration de test

Avant d'effectuer les tests, on site en général le schéma de communication entre les équipements représentés dans la figure 28, avec PC1 (Modbus RTU), Raspberry Pi (passerelle), PC2 (Modbus TCP).

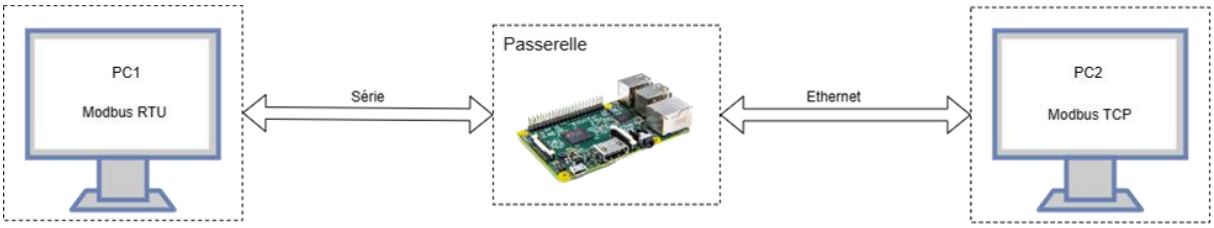

Figure 27:Schéma générale de la communication.

III.10.2. Test de fonctionnalité

Les tests de fonctionnalité visent à vérifier que la passerelle sécurisée fonctionne correctement et effectue les fonctions attendues. Cela implique de tester chaque fonctionnalité de la passerelle, telles que le chiffrement/déchiffrement, le contrôle d'accès, la validation des données.

Vérification de la communication

- Vérification de la communication entre PC1 et la passerelle : Le PC1 envoie des requêtes Modbus RTU au Raspberry Pi, qui les reçoit et les traite correctement. Les réponses de la passerelle sont renvoyées au PC1 avec les données correctes et les valeurs attendues.
- Vérification de la communication entre la passerelle et PC2 : Le Raspberry Pi reçoit les requêtes Modbus TCP du PC2, les traduit en Modbus RTU, les envoie à PC1 et renvoie les réponses à PC2. Les données sont correctement traduites entre les deux protocoles et les réponses sont conformes aux attentes.

Test de sécurité

Les données échangées entre la passerelle et PC2 sont chiffrées à l'aide de l'algorithme AES. Des tests sont effectués pour s'assurer que les données chiffrées sont correctement transmises et déchiffrées par les deux parties sans compromettre la sécurité des informations.

Test d'échange de données

- Échange de données normales : Différents types de données Modbus sont échangés entre PC1 et PC2 via la passerelle sécurisée. Les valeurs sont vérifiées pour s'assurer que les données sont correctement transmises et traitées par toutes les parties impliquées.
- Échange de données limites : Des cas limites, tels que des données incorrectes, des erreurs de transmission ou des erreurs de format, sont testés pour vérifier la robustesse de la passerelle et sa capacité à gérer de telles situations sans compromettre la communication.

La figure 29 nous montre le test de fonctionnalité effectuée par le client Modbus TCP, qui a envoyé une requête Modbus TCP en demandant à l'esclave Modbus RTU une lecture de 2 registres « holding registers ». Sur l'interface d'esclave Modbus RTU la requête reçue et la réponse à envoyer sont affichées. Finalement, le client reçoit la réponse selon sa demande ou la requête qu'il a envoyé.

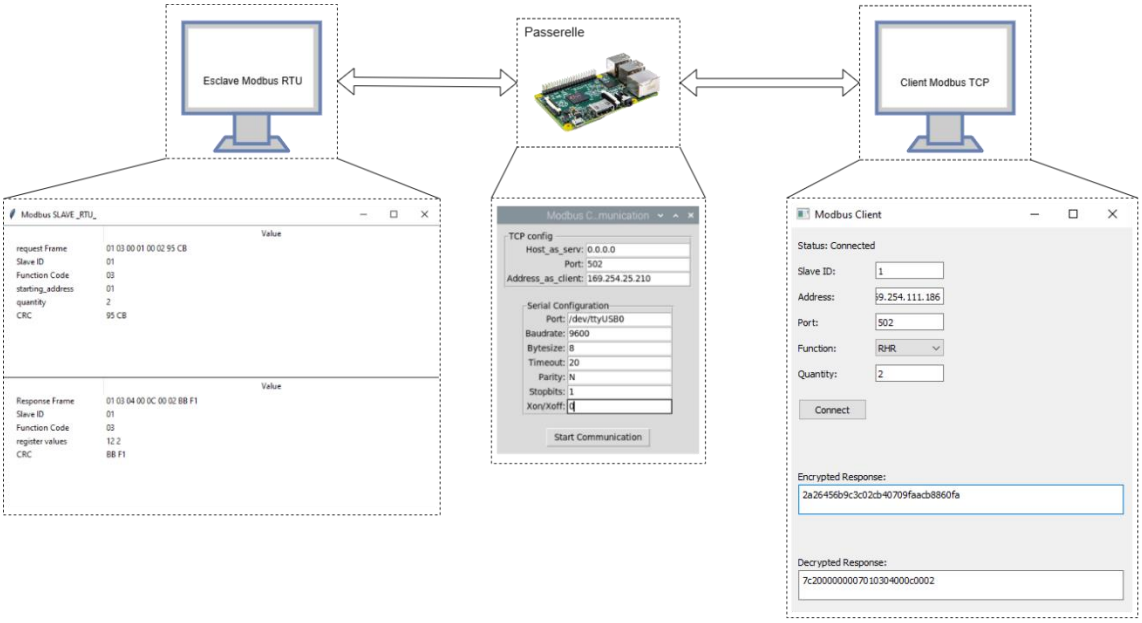

Figure 28:Test de communication entre Client TCP et Esclave RTU.

La figure 30 nous montre le test de fonctionnalité effectuée par le maître Modbus RTU, qui a envoyé une requête Modbus RTU en demandant à l'esclave Modbus RTU une lecture de 6 registres « holding registers ». Sur l'interface du serveur Modbus TCP la requête reçue est affichée. Finalement, le client reçoit la réponse selon sa demande ou la requête qu'il a envoyé.

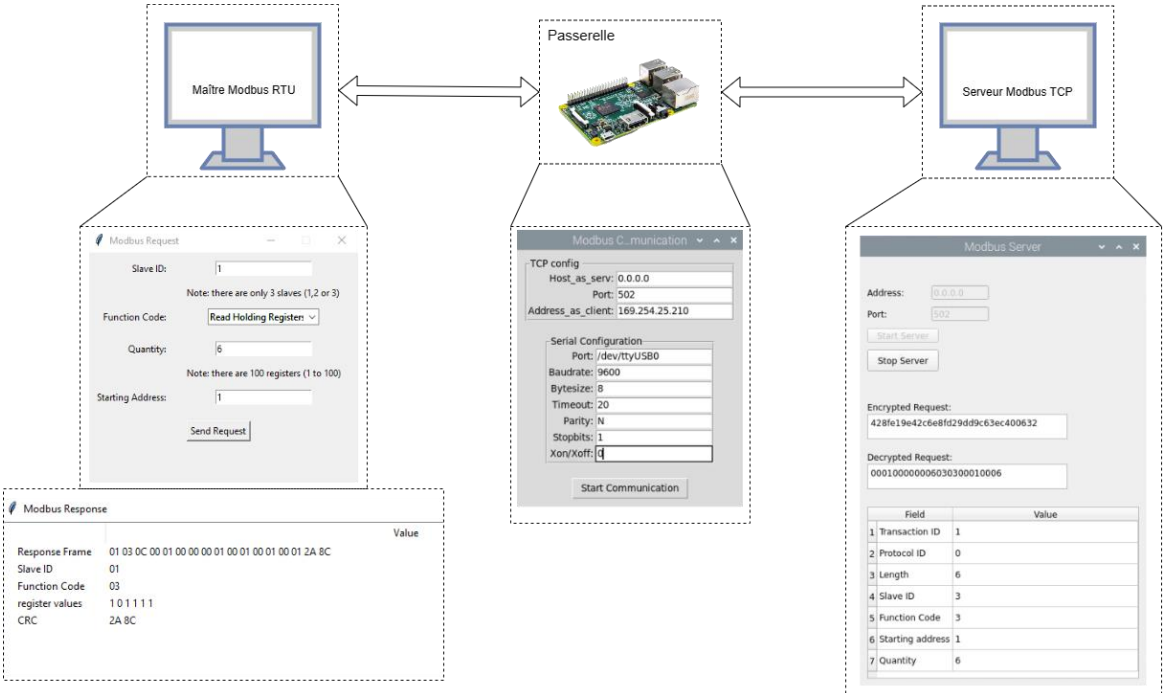

Figure 29:Test de communication entre maître RTU et Serveur TCP.

Conclusion générale

Ce mémoire visait à concevoir et à mettre en œuvre une passerelle sécurisée entre les protocoles Modbus RTU et Modbus TCP, en vue d'une intégration efficace des systèmes. Les principaux aspects explorés dans ce projet étaient l'interopérabilité entre les deux protocoles, la gestion de la conversion des données, la sécurité de la communication.

Nous avons effectué des tests approfondis, en établant une communication entre le maître Modbus RTU et le serveur Modbus TCP, en utilisant la passerelle. Nous avons vérifié la bonne conversion des données et mesuré les performances de la passerelle en termes de délai de transmission. De même, pour le cas de la communication entre le client Modbus TCP et l'esclave Modbus RTU.

Nous avons intégré des fonctionnalités de sécurité en utilisant l'algorithme de chiffrement AES, garantissant ainsi la confidentialité et l'intégrité des données transitant par la passerelle.

Ce projet a atteint avec succès ses objectifs en fournissant une passerelle fonctionnelle entre Modbus RTU et Modbus TCP. Les résultats obtenus ouvrent de nouvelles perspectives pour une intégration plus fluide et sécurisée des systèmes industriels utilisant ces protocoles. Ce projet constitue donc une contribution significative dans le domaine des systèmes SCADA et des communications protocolaires, offrant des avantages pour l'automatisation et la gestion efficace des opérations industrielles.

Références

[1] S. A.Boyer, «definition of scada,» chez *scada:supervisory control and data acquisition*, 2004, p. 9.

[2] P. Zhang, «Definition and functions,» chez *Advanced Industrial Control Technology*, 2014, p. 4.

[3] S. A.Boyer, «What is SCADA?,» chez *SCADA:Supervisory Control And Data Acquisition* , 2004, p. 9.

[4] «procetradi,» 2019. [En ligne]. Available: https://www.procetradi.com/proyecto/scada-project-for-electroperu.

[5] S. A.Boyer, «applicable processes,» chez *SCADA:supervisory control and data acquisition* , 2004, p. 10.

[6] L. S. Jan Stowisek, «General Architecture of SCADA Systems,» *A PVSS Application for Monitoring the Start-up of the Super Proton Synchrotron after Major Breakdowns,* pp. 15-16-17, 2001.

[7] A. M. MEZDOUR Hala, «Etude et réalisation d'un système de supervision sous,» 2018/2019.

[8] modbus.org, «Modbus,» 2012. [En ligne]. Available: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

[9] D. R. E. W. Gordon Clarke, «What is DNP3?,» chez *Pratical Modern SCADA protocols*, 2004, p. 66.

[10] «booWIKI,» [En ligne]. Available: https://boowiki.info/art/systemes-reseau/iec-60870.html.

[11] «FactoryFuture,» [En ligne]. Available: https://www.factoryfuture.fr/tout-savoir-systemes-scada/#:~:text=Les%20syst%C3%A8mes%20SCADA%20actuels%20sont,des%20donn%C3%A9es%20entre%20les%20n%C5%93uds..

[12] «PLClogix,» [En ligne]. Available: https://www.plclogix.com/downloads/PLCTech2_WEB/PLCTech2/pl170018.pl2.

[13] «BuiltIn,» [En ligne]. Available: https://builtin.com/data-science/scada.

[14] D. BARR, «Communications Network,» *Supervisory Control and Data Acquisition (SCADA) systems,* p. 19, 2004.

[15] «Evaluate and strengthen the security of any remaining connections to the SCADA network.,» *21 steps improve cyber security scada networks,* p. 3.

[16] «Modbus Organisation,MODBUS Application Protocol Specification,» 26 April 2012. [En ligne]. Available: http://www.modbus.org . [Accès le 04 may 2023].

[17] S. S. Mauricio Papa, «Security Analysis of Multilayer SCADA Protocols,» *Conference Paper,* pp. 206-222, March 2007.

[18] automation-sense, «COMPRENDRE ET METTRE EN ŒUVRE FACILEMENT LE BUS INDUSTRIEL MODBUS,» automation-sense, 2017.

[19] «Honeywell,» April 2021. [En ligne]. Available: http://www.cindtechs.com/unleashd/technotes/hc900_modbus_tcp.pdf. [Accès le may 2023].

[20] S. Marios, «SCADA Modbus TCP Protocol: Security and Authentication Issues,» October 5, 2020.

[21] E. D. K. J. T. Langill, Industrial Network Security Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems, British Library Cataloguing-in-Publication Data, 2015.

[22] «MOXA,» MOXA, [En ligne]. Available: https://www.moxa.com/en/products/industrial-edge-connectivity/protocol-gateways/modbus-tcp-gateways/mgate-mb3180-mb3280-mb3480-series.

[23] «GoTRONIC,» [En ligne]. Available: https://www.gotronic.fr/art-carte-raspberry-pi-3-b-27826.htm#:~:text=Le%20mod%C3%A8le%20Raspberry%20Pi3%20B%2B,broches%20d%27E%2FS..

[24] «splashtop,» [En ligne]. Available: https://www.splashtop.com/fr/blog/best-vnc-alternative-splashtop-remote-access-control#:~:text=Le%20Virtual%20Network%20Computing%20(VNC,distant%20avec%20votre%20ordinateur%20local..

[25] «Geekflare,» [En ligne]. Available: https://geekflare.com/fr/best-tools-to-connect-unix-server/.

[26] «REAL PYTHON,» [En ligne]. Available: https://realpython.com/python-raspberry-pi/#:~:text=The%20Raspberry%20Pi%20Foundation%20specifically,Python%20on%20the%20Raspberry%20Pi..

Annexes

Annexe A

Les codes de fonction de Modbus.

- **01 (0x01) Lecture de bobines (Read Coils)**

Ce code de fonction est utilisé pour lire de 1 à 2000 états continus de coils dans un dispositif distant. La demande PDU spécifie l'adresse de départ et le nombre de bobines, et le message de réponse est emballé avec une bobine par bit du champ de données. L'état est indiqué par 1=ON et 0=OFF, et le bit de poids faible (LSB) du premier octet de données contient la sortie adressée dans la requête. Si la quantité de sortie n'est pas un multiple de huit, les bits restants dans le dernier octet de données seront remplis avec des zéros. Le champ du nombre d'octets compte la quantité d'octets de données complets.

- **02 (0x02) Lecture des entrées discrètes (Read Discrete Inputs)**

Le code de fonction Lecture des entrées discrètes est utilisé pour lire de 1 à 2000 états continus d'entrées discrètes dans un dispositif distant. La demande PDU spécifie l'adresse de départ et le nombre d'entrées, et les entrées discrètes sont adressés à partir de zéro. Le message de réponse est emballé avec une entrée par bit du champ de données, avec un état indiqué comme 1= allumé; 0=éteint. Si la quantité d'entrées retournée n'est pas un multiple de huit, les bits restants dans le dernier octet seront remplis de zéros. Le champ du nombre d'octets compte la quantité d'octets de données complètes.

- **03 (0x03) Lecture des registres de maintien (Read Holding Registers)**

Un code fonction qui permet de lire le contenu d'un bloc contigu de registres de maintien dans un dispositif distant. La demande PDU spécifie l'adresse du registre de départ et le nombre de registres à lire. Le message de réponse est emballé en utilisant deux octets, avec le contenu binaire justifié à droite à l'intérieur de chaque octet.

- **04 (0x04) Lecture des registres d'entrée (Read Input Registers)**

Ce code est utilisé pour lire de 1 à 125 registres d'entrée contigus dans un dispositif distant. La demande PDU spécifie l'adresse du registre de départ et le nombre de registres à lire, et le message de réponse est emballé avec deux octets contenant le contenu binaire justifié à droit [16].

- **05 (0x05) Write Single Coil**

Il est utilisé pour écrire une seule sortie à l'état ON ou OFF dans un dispositif distant. L'état ON/OFF souhaité est spécifié par une constante dans le champ de données de la demande. La demande PDU spécifie l'adresse de la bobine à forcer. L'état ON/OFF souhaité est spécifié par une constante dans le champ de valeur de la bobine [16].

- **06 (0x06) Write Single Register**

Ce code de fonction est utilisé pour écrire un seul registre de maintien dans un dispositif distant. La demande PDU est utilisée pour écrire un seul registre de maintien dans un appareil distant, en commençant à zéro, et renvoie un écho de la demande une fois que le contenu du registre a été écrit [16].

Annexe B

Codes d'exception

Si un esclave reçoit une demande invalide, il répond avec une réponse d'exception. Ainsi, le maître peut être informé de l'état de l'esclave. Les exceptions typiques sont une fonction non autorisée, un code non autorisé et une adresse de données non autorisée. Si le maître essaie de lire à partir d'un registre non mappé dans la carte mémoire Modbus de l'esclave, une exception sera générée [16].

Tableau 7: Exemples de codes d'exception Modbus [16].

Code d'exception	Signification	Explication
1	Fonction erronée	Le code de fonction reçu dans la requête n'est pas une action autorisée pour le serveur.
2	Adresse incorrecte	L'adresse de données reçue dans la requête n'est pas une adresse autorisée pour le serveur.
3	Donnée incorrecte	Une valeur contenue dans le champ de données de la requête n'est pas une valeur autorisée pour le serveur.
4	Défaut esclave	Une erreur irrécupérable s'est produite pendant que le serveur tentait d'exécuter l'action demandée
5	Acquittement	Le serveur a accepté la demande et est en train de la traiter, mais une longue période de temps sera nécessaire pour le faire.
6	Équipement occupé	Le serveur est en train de traiter une commande de programme de longue durée.
8	Erreur de parité sur la mémoire	Le serveur a tenté de lire le fichier d'enregistrement, mais a détecté une erreur de parité dans la mémoire.

Annexe C

Configuration de la carte Raspberry Pi

Nous avons suivis ces étapes pour configuré notre carte Raspberry Pi

- Télécharger logiciel "Raspberry Pi Imager" pour configurer la carte SD (16 Go) avec le système d'exploitation Raspbian,
- Insertion de la carte SD préparée dans le lecteur de carte Raspberry Pi,
- Connecter la carte Raspberry Pi avec un ordinateur en utilisant un câble Ethernet et à l'aide de PuTTy pour accéder à la ligne de commande de la Raspberry Pi. En utilisant le nom associé à la carte Raspberry Pi (raspberrypi.local) pour se connecter via SSH, dans PuTTy. Cliquant sur "Open" pour établir la connexion SSH comme illustré dans la figure 18. Une fois connecté à la Raspberry Pi via PuTTy, on peut exécuter des commandes de configuration, installer des logiciels supplémentaires, …etc., en fonction de nos besoins,
- Puisque nous allons utiliser le logiciel VNC Viewer, et pour cela il faut configurer VNC Server su la carte RaspberryPi, nous entrons la commande « sudo raspi-config » sur le terminal afficher sur PuTTy et configurons VNC.la figure 31 montre cette configuration de VNC sur RaspberryPi,

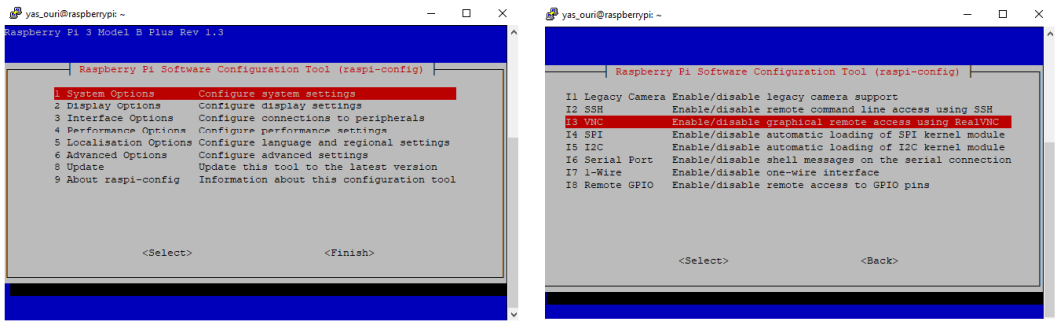

Figure 30: Configuration de VNC Viewer sur Raspberry Pi.

- Une fois VNC Server est configuré, on accède à la Raspberry Pi à distance en connectant la Raspberry Pi à un ordinateur à l'aide d'un câble Ethernet. Sur cet ordinateur, on ouvre VNC Viewer et entre l'adresse IP de la Raspberry Pi pour établir la connexion à distance. Maintenant on peut accéder à l'interface graphique de la Raspberry Pi et interagir avec elle. Cette étape est représentée dans la figure 32.

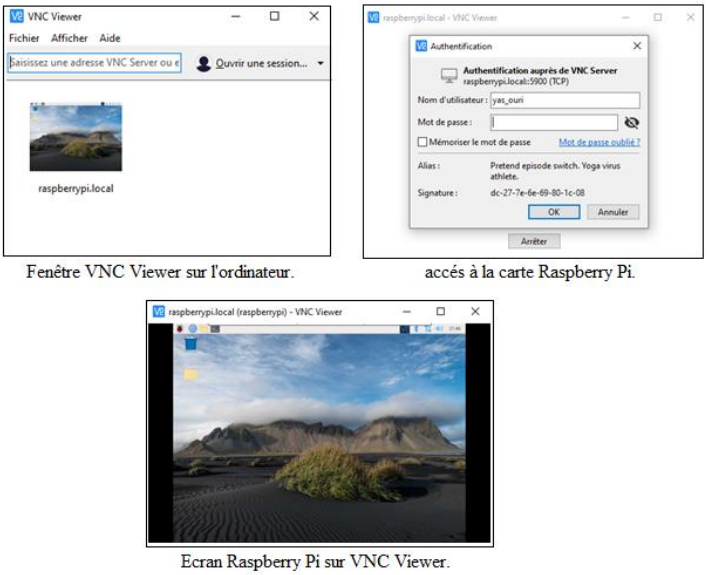

Figure 31: Ecran d'accueil de la Raspberry Pi.

ملخص

تترافق تطورات الأنظمة الصناعية في كثير من الأحيان مع تغيرات في بروتوكولات الاتصال المستخدمة في معدات التحكم. يمكن أن يشكل ذلك تحديًا في ضمان التوافق والتواصل بين المعدات القديمة والتقنيات الجديدة. في هذا السياق، يهدف مشروعنا إلى تقديم بوابة اتصال قادرة على ربط بروتوكولات مختلفة. تحديدًا، ركزنا على بوابة بين بر Modbus TCP و Modbus RTU .

كلمات مفتاد بروتوكول Modbus RTU Modbus TCP ، إطار، بوابةModbus TCP، Modbus RTU، Modbus، SCADA الاتصال،

Résumé

L'évolution des systèmes industriels s'accompagne souvent de changements dans les protocoles de communication utilisés par les équipements de contrôle. Cela peut poser des défis pour assurer l'interopérabilité et la compatibilité entre les équipements anciens et les nouvelles technologies. Dans ce contexte, notre projet vise à proposer une passerelle de communication capable de faire le lien entre différents protocoles. Plus précisément, nous nous sommes concentrés sur la passerelle entre les protocoles Modbus RTU et Modbus TCP.

Mots clés : Modbus RTU, Modbus TCP, Trame, passerelle Modbus RTU Modbus TCP, protocole de communication, SCADA, Modbus.

Abstract

The evolution of industrial systems often involves changes in the communication protocols used by control equipment. This can pose challenges in ensuring interoperability and compatibility between legacy equipment and new technologies. In this context, our project aims to propose a communication gateway capable of bridging different protocols. Specifically, we focused on the gateway between Modbus RTU and Modbus TCP protocols.

Keywords: Modbus RTU, Modbus TCP, Frame, Modbus RTU Modbus TCP gateway, communication protocol, SCADA, Modbus.