REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

Laboratoire CENTRELEC R&D

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Étude et réalisation d'un nœud de communication IEC 61850

NEMDIL Ahmed

REBIKA Samir

Sous la direction de M. Mourad ADNANE MCA.

Présenté et soutenue publiquement le (30/06/2019)

Composition du jury :

Président	**M. Rabah SADOUN**	**Prof.**	**ENP**
Promoteur	**M. Abdelmoumen LARFI**	**Ingénieur.**	**CENTRELEC**
Promoteur	**M. Mourad ADNANE**	**MCA.**	**ENP**
Examinateur	**M. Mohamed Oussaid TAGHI**	**Dr.**	**ENP**

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

Laboratoire CENTRELEC R&D

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Étude et réalisation d'un nœud de communication IEC 61850

NEMDIL Ahmed

REBIKA Samir

Sous la direction de M. Mourad ADNANE MCA.

Présenté et soutenue publiquement le (30/06/2019)

Composition du jury :

Président	**M. Rabah SADOUN**	**Prof.**	**ENP**
Promoteur	**M. Abdelmoumen LARFI**	**Ingénieur.**	**CENTRELEC**
Promoteur	**M. Mourad ADNANE**	**MCA.**	**ENP**
Examinateur	**M. Mohamed Oussaid TAGHI**	**Dr.**	**ENP**

ENP 2019

Dédicace

On dédie ce travail à nos chers parents et grands parents,

A nos familles,

A nos frères et soeurs,

A nos amis
Et à tous ceux qui ont fait confiance à Nos Capacités pendant notre parcours.

Remerciements

Nous remercions Dieu de nous avoir donné courage, santé et soutien durant les moments les plus difficiles.

Nous souhaitons exprimer notre sincère gratitude à notre conseiller, **M. ADNANE Mourad**, pour sa patience, sa motivation et sa compréhension au cours des trois dernières années. Ses conseils et questions pertinentes nous ont aidés tout au long de la recherche et de la rédaction de ce projet. Nous n'aurions pas pu imaginer avoir un meilleur conseiller et mentor.

Nous remercions également **M. LARFI Abdelmoumen** pour son temps, ses précieux conseils et pour nous avoir donné la chance de travailler sur un projet d'une si grande importance.

Nous sommes également reconnaissants à toute l'équipe de **CENTRELEC** et plus particulièrement à **M. BOUTOUCHE Hicham** pour sa disponibilité, ses précieux conseils et ses encouragements ; **M. TALEB Abdelkrim** et **M. MOHAREM Youcef** pour leur patience et leur disponibilité.

Nous remercions sincèrement les membres du jury, **M. SADOUN Rabah** et **M. TAGHI Mohamed Oussaid** , qui ont généreusement offert leur temps, leur soutien, leurs conseils et leur bonne volonté durant ces trois dernières années.

Nous remercions tout particulièrement nos parents de nous avoir incités sans relâche à explorer tous les sujets qui nous intéressaient.

Dernier mais pas des moindres ; nous voudrions remercier nos familles et amis pour leur soutien constant. Rien de tout cela n'aurait pu arriver sans eux.

ملخّص

توحيد الاتصالات في الشبكة الذكية ضروري لتسهيل العمليات المعقدة لوظائف نظام الطاقة الحديثة. تطبيق معيار اللجنة الكهروتقنية الدولية 61850 في مثل هذا النظام يفي بالقدرة على إدارة عدد كبير من الأجهزة وتمكينها من التواصل مع بعضها البعض بطريقة سريعة وموثوق بها لضمان التحكم والحماية الكاملين.

تعرض هذه الوثيقة دراسة وتنفيذ لعقدة اتصال بحسب معيار اللجنة الكهروتقنية الدولية 61850 مبنية على مجموعة من البرامج مفتوحة المصدر. بعد ذلك، يقدم نتائج الاختبار من وجهات نظر مختلفة باستخدام طرق مختلفة.

الكلمات المفتاحية: اللجنة الكهروتقنية الدولية 61850، عقدة اتصال، أنظمة الطاقة الذكية.

Abstract

Standardization in smart grid communications is necessary to facilitate complex operations of modern power system functions. The implementation of the IEC 61850 standard in such a system fulfil the capability to manage a large number of devices and enable them to communicate with one another in a fast and reliable way so as to guarantee the full control and protection. This document presents a study and an implementation of an IEC 61850 communication node built from a combination of open source software. After that, it presents the test results from different perspectives using various methods.

Key words : IEC61850, communication node, intelligent power systems.

Résumé

La normalisation des communications par réseau intelligent est nécessaire pour faciliter les opérations complexes des fonctions des systèmes d'alimentation modernes. La mise en œuvre de la norme IEC 61850 dans un tel système permet de gérer un grand nombre de périphériques et de leur permettre de communiquer entre eux de manière rapide et fiable afin de garantir le contrôle et la protection complète. Ce document présente une étude ainsi qu'une mise en œuvre d'un nœud de communication IEC 61850 construit à partir d'une combinaison de logiciels à source ouverte. Ensuite, il présente les résultats du test sous différentes perspectives en utilisant diverses méthodes.

Mots clés : IEC 61850, nœud de communication, systèmes d'alimentation intelligents.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

API	Application Program's Interface
DA	Data Attribute
DO	Data Object
GOOSE	Generic Object Oriented Substation Event
HAL	Hardware Abstraction Layer
HMI	Human Machine Interface
IED	Intelligent Electronic Device
IP	Internet Protocol
LAN	Local Area Network
LD	Logical Device
LN	Logical Node
MAC	Media Access Control
MMS	Manufacturing Message Specification
OSI	Open System Interconnection
PDU	Protocol Data Unit
SA	Substation Automation
SAS	Substation Automation System
SCL	System Configuration Language
SV	Sampled Values
TCP	Transmission Control Protocol
XML	Extensible Markup Language

Introduction

Dans le contexte du développement industriel, le domaine de l'électronique et en particulier de la communication numérique est largement présent, cela est dû au fait de la facilité d'installation, de la capacité de stockage et de traitement.
Dans une sous-station industrielle, les machines (moteurs) sont contrôlées à travers des dispositifs électroniques programmables représentés sous forme de systèmes embarqués qui sont interconnectés dans un réseau. Par conséquent, un protocole de communication est nécessaire pour permettre l'échange des informations entre ces différents périphériques.
Des fournisseurs internationaux multiples tels qu'ABB et SIEMENS présentent divers protocoles de communication pour leurs produits. Cependant, ces dernières années, l'accent est mis sur la norme IEC 61850 en raison de son caractère inclusif pour la définition des différents niveaux de sous-stations ainsi que son concept de modélisation basé sur l'orienté-objet.
Le but de ce travail est l'étude et la mise en œuvre d'un nœud de communication conforme aux protocoles du niveau industriel définis par la norme IEC 61850 en utilisant des solutions Open Source.
Ce document est organisé comme suit :

- **Chapitre 1 (Aperçu global sur la norme IEC 61850)**

Dans ce chapitre, on introduit les concepts de base de la norme IEC 61850 en présentant le principe général ainsi que les notions fondamentales y afférant.

- **Chapitre 2 (Protocoles de communication IEC 61850)**

Dans ce chapitre, les protocoles de communication de la norme IEC 61850 sont présentés. Chaque protocole est étudié en détail selon des perspectives différentes.

• **Chapitre 3 (Choix techniques)**

Dans ce chapitre, on aborde les choix matériels et logiciels considérés en fonction des besoins et des spécifications demandées dans le cahier des charges.

• **Chapitre 4 (Implémentation)**

Dans ce chapitre, on présente les procédés de développement ainsi que les étapes nécessaires à l'implémentation.

• **Chapitre 5 (Test et résultats)**

Dans ce chapitre, nous présentons les résultats détaillés des tests effectués sur le nœud de communication implémenté, et cela pour les différents cas de figures existants.

• **Conclusion générale**

Dans la conclusion générale, on énumère les principaux résultats de notre travail et on présente des perspectives.

• **Annexe**

L'annexe précise les étapes d'installation et de configuration des hardwares et softwares utilisés et certains concepts parallèles ainsi que des scripts expliqués.

Chapitre 1

Aperçu global sur la norme IEC 61850

1.1 Introduction

Au cours de la dernière décade, la « numérisation » de l'entreprise a connu une croissance exponentielle. Les services publics, industriels, commerciaux et même résidentiels sont entrains de transformer tous ses aspects technologiques vers le domaine numérique.

À l'avenir, chaque pièce d'équipement devrait posséder un type de réglage, de surveillance et/ou de contrôle. Afin de pouvoir gérer le grand nombre d'appareils et assurer l'échange d'informations entre eux, un nouveau modèle de communication était nécessaire. Ce modèle a été développé et normalisé en tant que IEC 61850.

Ce premier chapitre a pour but d'introduire les notions de base concernant la norme IEC 61850 ainsi que les concepts associés.

1.2 Définition

De nos jours, des fonds importants sont investis dans les systèmes électriques afin d'améliorer les performances de production et de distribution d'électricité. L'objectif est d'inclure l'automatisation et le contrôle intelligent dans l'industrie électrique, qui est l'un des aspects fondamentaux lié à l'étude de l'automatisation des sous-stations.

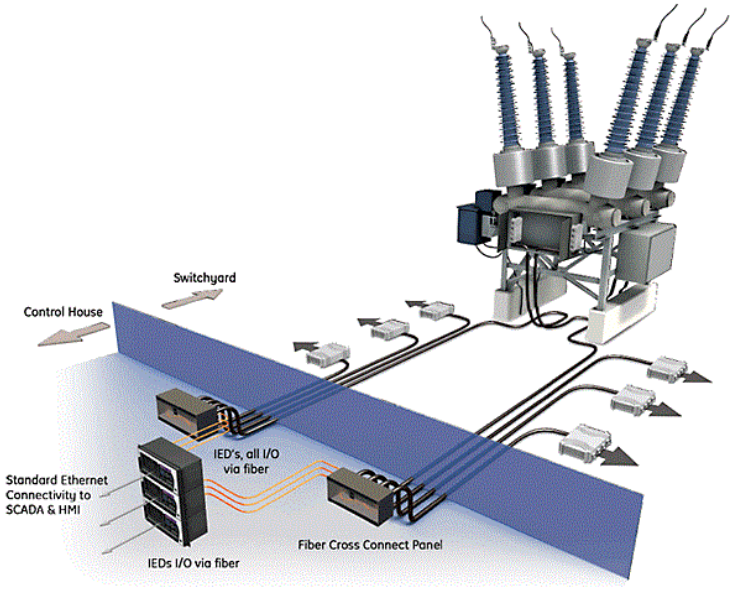

FIGURE 1.1 – Exemple d'un système d'automatisation de sous-station

Source :`http://blogspot.com/`

• **IEC 61850**

La norme internationale IEC 61850 a été introduite en 2003 et définit des protocoles standard pour la communication et l'interopérabilité des appareils, en tenant compte d'une structure de référence pour les systèmes d'automatisation de sous-stations. La norme définit une conception architecturale destinée à organiser l'intelligence des fonctions de protection, de surveillance, de contrôle et d'automatisation dans la sous-station. Cet aspect est réalisé grâce à un ensemble de modèles abstraits et de protocoles de communication, ainsi qu'à des méthodes d'élaboration indépendantes de la plateforme matérielle, des méthodes de communication et des outils de configuration.[1]

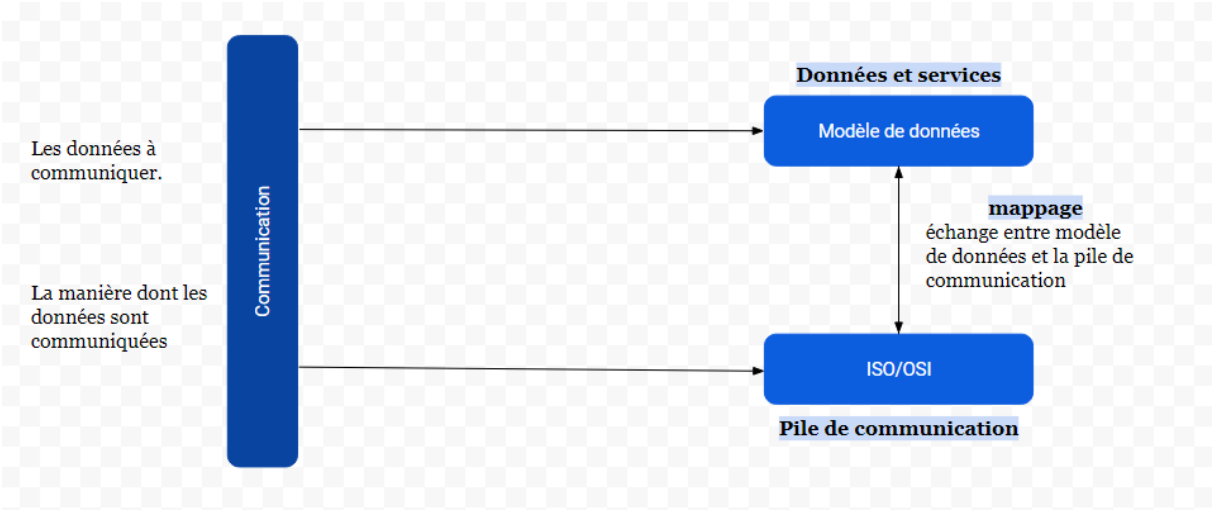

FIGURE 1.2 – La séparation entre le modèle de données et le coté communication dans la norme IEC 61850

• **Dispositif électronique intelligent (IED)**

Un dispositif électronique intelligent (IED) est un terme utilisé dans l'industrie de l'énergie électrique pour décrire les contrôleurs à microprocesseur des équipements du système d'alimentation (ou simplement des relais intelligents), tels que les disjoncteurs, les transformateurs et les batteries à condensateurs. Les IED reçoivent les données des capteurs et des équipements de puissance et peuvent émettre des commandes de contrôle, telles que le déclenchement de disjoncteurs s'ils détectent des anomalies de tension, de courant ou de fréquence, ou la commande des niveaux de tension. Les types courants d'IED incluent les dispositifs de relais de protection, les contrôleurs de changeur de prise en charge, les contrôleurs de disjoncteurs, les commutateurs de batterie à condensateurs, les contrôleurs de ré-enclenchement, les régulateurs de tension,

etc. La figure 1.3 montre un IED commercial de la marque SEL.[2]

FIGURE 1.3 – Un IED de la marque SEL

Source :`http://www.directindustry.com/`

• **Système SCADA**

Le système SCADA (contrôle de surveillance et acquisition de données) est un système de surveillance et de contrôle à distance fonctionnant avec des signaux codés sur des canaux de communication (utilisant généralement un canal de communication par station distante). Le système de commande peut être combiné à un système d'acquisition de données en ajoutant l'utilisation de signaux codés sur des canaux de communication. Cela est fait pour acquérir des informations sur l'état des équipements distants pour des fonctions d'affichage ou d'enregistrement. La figure 1.4 présente un exemple d'un système SCADA chez SONATRACH.[1]

FIGURE 1.4 – Un système SCADA pour les Pipelines de SONATRACH

Source :http://trcat.com/

• **L'interface homme-machine HMI**

L'HMI est une interface utilisateur ou un tableau de bord qui connecte une personne à une machine, un système ou un périphérique. Bien que le terme puisse techniquement s'appliquer à tout écran permettant à un utilisateur d'interagir avec un périphérique, les HMI sont le plus souvent utilisées dans le contexte d'un processus industriel. Les HMI peuvent être utilisés pour afficher visuellement des données, suivre le temps de production, les tendances, les tags et surveiller les entrées/sorties de la machine, etc. La figure 1.5 montre une HMI commerciale de la marque SIEMENS.[3]

FIGURE 1.5 – HMI de la marque SIEMENS

Source :https://cloudinary.com/

1.3 Modèle de sous-station

L'automatisation de sous-stations (SA) fait référence à un système qui supervise les commandes et associe les composants de distribution d'énergie installés dans une sous-station. La norme IEC 61850 a été définie à l'origine pour les systèmes d'automatisation de sous-station (SAS), y compris les applications de protection.[4] La norme IEC 61850 divise la sous-station en 3 niveaux :

Niveau de processus (Process Level) : Contient les équipements qui servent comme entrées/sorties à distance, il comprend les équipements de poste de manœuvre, des capteurs et des actionneurs. Il utilise le bus de processus qui permet la communication entre l'équipement de processus et les dispositifs de niveau supérieur (IED en général).

Niveau de la baie (Bay Level) : Situé au-dessus du niveau du processus, il est composé de disjoncteurs et d'isolateurs associés, de commutateurs de terre et de transformateurs d'instrument, tous combinés dans un ou plusieurs IED. À ce niveau, les IED fournissent toutes les fonctions du niveau de la baie, telles que la commande (commande des outputs), la surveillance (indications d'état, les valeurs mesurées) et la protection.

Niveau de station (Station Level) : Le niveau le plus élevé dans un SAS, qui contient la station SCADA pour l'acquisition des données et le contrôle (exemple : à partir de la HMI). Il contient également les fonctions orientées station utilisées pour contrôler et surveiller la station via le bus de station connecté au niveau de la baie.[1]

La figure 1.6 présente un schéma descriptif de la sous-station défini selon le standard IEC 61850. La norme IEC 61850 détermine que toutes les connexions de communication des deux bus utilisent Ethernet (ISO / IEC 8802-3) comme technologie de communication de base.

1.4 Les domaines de la norme IEC 61850

Comme décrit dans la section précédente, le standard IEC 61850 fournit une architecture de référence de la sous-station à travers différents niveaux. Une autre classification existe ; Cette dernière est associée à une échelle supérieure dans laquelle la norme définit le modèle de données, la configuration et la communication pour les différents dispositifs des trois niveaux.

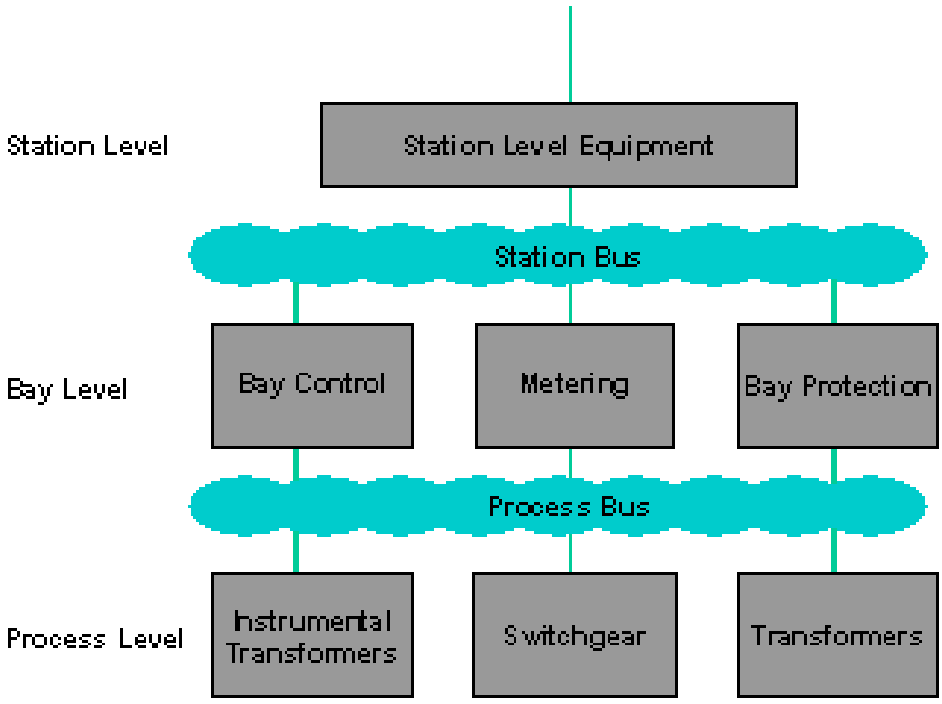

FIGURE 1.6 – Schéma descriptif de la sous-station selon IEC 61850

Source :https://www.nettedautomation.com/

1.4.1 Modèle de données

IEC 61850 modélise les informations communes et les fonctions disponibles dans les dispositifs physiques, qui sont décrites par la norme à l'aide d'une approche de modélisation orientée objet.

Le modèle de données (data model) a une hiérarchie spécifique illustrée dans figure 1.7. Les périphériques physiques (principalement les IED) dans une SAS englobent de nombreux périphériques logiques (LD) représentant un groupe de fonctions avec leurs informations d'entrée et de sortie. De même, il existe des nœuds logiques (LN) qui sont regroupés dans chaque périphérique logique (LD) afin de représenter les informations produites et utilisables par une fonction unique d'un périphérique logique (LD).

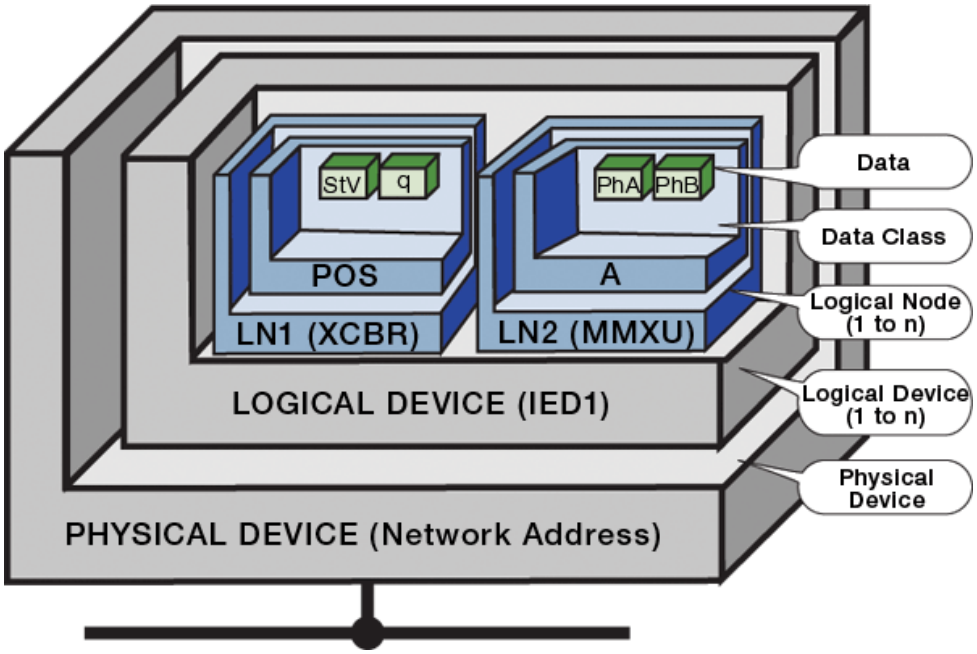

FIGURE 1.7 – IEC 61850 Data Model

Source :https://blogs.dnvgl.com/

Selon le nœud logique (LN), une classe de données (également appelée objet de données : DO) est définie, elle porte un nom associé spécifique. La classe de données peut ainsi avoir des paramètres (attributs de données : DA ou simplement data) permettant de décrire une fonction, valeur ou un état spécifique.

Globalement, le LD existe dans le périphérique physique, le LN existe dans le LD et le DO existe dans le LN, ainsi que le DA (s'il est définit) existe dans un DO. La figure 1.8 décrit un exemple de périphérique physique constitué de LD nommée «relay1» pour la protection, avec deux LN, le premier définit une unité de mesure «MMXU1» pour laquelle nous voulons déterminer la valeur du courant électrique. Le deuxième est un disjoncteur « XCBR2 » pour lequel nous voulons déterminer sa position (soit ouvert ou fermé).[5]

Un modèle de périphérique IEC 61850 est décrit par un fichier de configuration écrit en langage SCL. Il contient les définitions des paramètres LD, LN ainsi que les paramètres d'échange d'informations. Il utilise la même sémantique pour assurer l'interopérabilité. La partie configuration est décrite dans la sous-section 1.4.2.

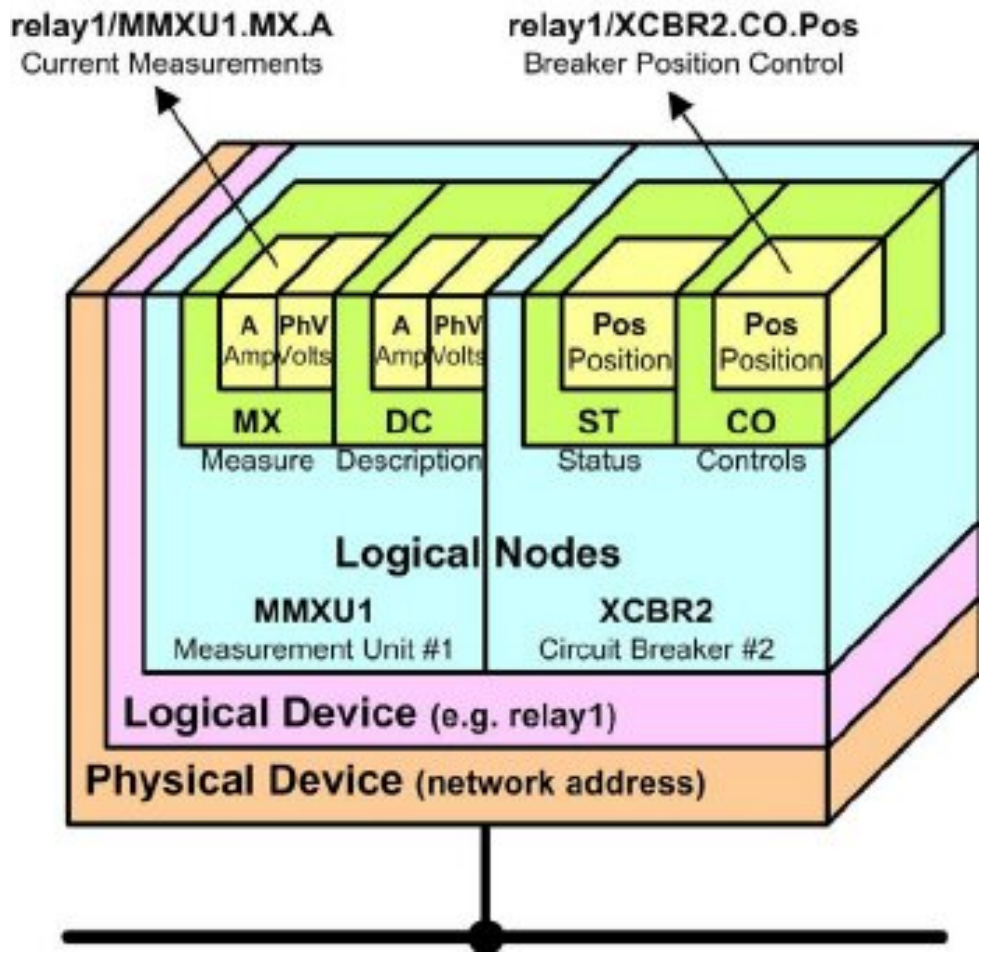

FIGURE 1.8 – Lecture d'une donnée à partir du Data Model

Source :https://www.researchgate.net/

1.4.2 Ingénierie (configuration)

Afin de configurer le IED, la norme IEC 61850 introduit un format de fichier pour décrire la configuration. Le langage utilisé dans ce format de fichier est le langage SCL (Substation Configuration Languge) basé sur XML. SCL spécifie une hiérarchie de fichiers de configuration permettant de décrire plusieurs niveaux du système de manière normalisée et non ambiguë.

SCL peut être utilisé pour restructurer l'ensemble du système d'alimentation afin d'éliminer la configuration manuelle, de réduire les incompréhensions entre les capacités et les exigences du système, d'améliorer l'interopérabilité du système final, d'éliminer les erreurs de saisie manuelle des données et d'accroître considérablement la productivité et l'efficacité des ingénieurs de système d'alimentation[6]. Il existe cinq types de fichiers SCL. Ils sont présentés dans le tableau 1.1.

TABLE 1.1 – Types des fichiers SCL

Nom	Extension	Description
System Specification Description	.SSD	Il décrit les fonctions du SAS et les types requis de LN, mais sans description spécifique de l'IED.
IED Capability Description	.ICD	Il décrit les capacités fonctionnelles d'un type d'IED. Il ne devrait donc comporter qu'une section IED.
System Exchange Description	.SED	Description de l'interface pour un autre projet pour qu'il puisse utiliser le projet spécifié.
System Configuration Description	.SCD	Il décrit les IED utilisés dans le projet, le flux de données entre eux et les modèles requis.
Configured IED Description	.CID	Description de la partie communication d'un IED instancié. Il ne contient que les informations sur l'IED que le configurateur d'IED doit connaître.

Source :`https://www.semanticscholar.org/`

Certains fichiers peuvent être utilisés dans des cas spécifiques de modification de configuration ou simplement de descriptions SAS.Le fichier SCL est divisé en cinq sections :

• **En-tête :** identification du fichier de configuration SCL, informations sur la version et options pour le mappage des noms sur les signaux.

• **Description de sous-station :** description de la structure fonctionnelle d'un poste.

• **Description de l'IED :** configuration d'un IED, des dispositifs logiques, des nœuds logiques et des descriptions de leurs objets de données.

• **Description du système de communication :** décrit les points d'accès IED connectés à un réseau.

• **Modèles de type de données :** utilisés pour définir les types de nœud logique et des données.[7]

1.4.3 Communication

Les services de communication IEC61850 sont mappés dans différents protocoles de communication selon le modèle OSI (Open Systems Interconnection) en tenant compte du modèle de données défini par la norme elle-même. Ces protocoles répondent à différents besoins dans une SAS, tout en maintenant l'interopérabilité des périphériques de différents fournisseurs. Dans ce qui suit, nous allons donner une brève introduction sur les trois protocoles de communication offert par la norme IEC 61850, plus de détails seront abordés dans le chapitre 2.

Les principaux protocoles de communication de la norme IEC 61850 sont les suivants :

• **Manufacturing Message Specification (MMS)**

Protocole de communication unicast basé sur un mécanisme client-serveur permettant l'interchangeabilité des données d'application entre le niveau de la station et le niveau de la baie.

• **Generic Object Oriented Substation Events (GOOSE)**

Protocole de communication multicast qui fonctionne sur le mécanisme éditeur-abonné (publisher-subscriber). Il est principalement utilisé pour envoyer des messages urgents, en particulier entre des périphériques au niveau de la baie (IED). L'un de ses avantages est la facilité de configuration par rapport aux connexions câblées.

• **Sampled Value (SV) - Valeur échantillonnée**

Comme GOOSE, le protocole SV utilise un mécanisme éditeur-abonné, il est utilisé principalement pour envoyer les mesures de courant et de tension de manière numérique sur le bus de processus.

1.5 Conclusion

Dans ce chapitre, nous avons présenté les concepts de la norme IEC 61850 qui constitueront la base de notre travail.

Dans le chapitre suivant, nous allons nous plonger dans la partie communication où nous expliquerons les protocoles IEC 61850 en détails, ce qui correspond à la portée de notre travail.

Chapitre 2

Protocoles de communication IEC 61850

2.1 Introduction

L'idée principale de la norme IEC 61850 consiste à interagir entre différents constructeurs via une connexion physique (réseau local Ethernet LAN), afin de pallier le manque d'interopérabilité lorsque de nombreux constructeurs proposent leur propre protocole de communication ou structure de données. Ce chapitre traite les différentes parties de la pile de communication basée sur la norme IEC 61850 (qui inclut MMS, GOOSE et SV) pour une implémentation dans un SAS.

2.2 Manufacturing Message Specification (MMS)

2.2.1 Caractéristiques du protocole MMS

La pile MMS offre une communication prenant en charge des modèles d'objet normalisés et des services de communication fournissant de nombreuses fonctions de transfert d'informations (ex. fonctions de lecture / écriture) et de données de contrôle et de supervision (ex. fonctions d'état et de contrôle) entre appareils en réseau et / ou applications informatiques.

MMS garantie la communication à l'aide d'un modèle client-serveur (voir figure 2.1). Un client est une application ou un dispositif réseau (par exemple, un système de surveillance ou un centre de contrôle) qui demande des données ou une action au serveur.

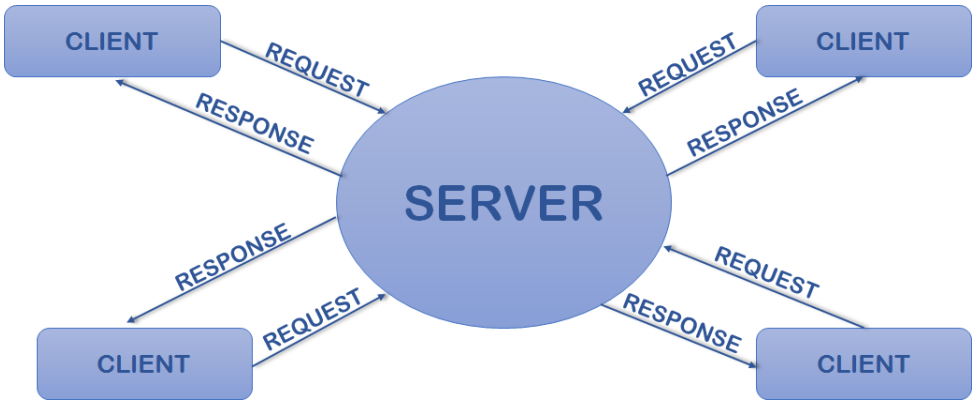

FIGURE 2.1 – Schéma montrant le modèle client-serveur

Source :`https://msatechnosoft.in/`

Un serveur (vu à partir de l'approche IEC 61850) est un appareil ou une application qui contient la description du modèle de données et ses objets (appelés variables) auxquels le client MMS peut accéder. Le client envoie des demandes de service MMS et le serveur répond à ces demandes.
MMS définit les éléments suivants :

• Un ensemble d'objets standards qui existent dans chaque périphérique, sur lesquels des opérations telles que lecture, écriture, signalisation d'événement, etc. peuvent être exécutées.

• Le niveau serveur est considéré comme un périphérique de fabrication virtuel (VMD : Virtual Manufacturing Device) et est l'objet principal, tous les autres objets tels que les variables, les domaines, les journaux, les fichiers, etc. relèvent du VMD.

• Un ensemble de messages standards échangés entre un poste client et un poste serveur dans le but de surveiller et / ou de contrôler ces objets.

• Un ensemble de règles de codage pour mapper ces messages en bits et en octets lors de la transmission (ex. codage par ASN.1 et BER Encoding).

Tout cela repose sur une communication de moindre priorité (fonctions en temps réel non nécessaires) avec des messages d'informations opérationnelles générales et une priorité moyenne en comparant avec les protocoles GOOSE et SV.[8]

2.2.2 Utilisation du protocole MMS

MMS est principalement utilisé pour les messages non urgents qui ne nécessitent pas de transmission en temps réel spécifiquement. Il est basé sur une communication unicast entre le client (qui demande) et le serveur (qui répond).

De nombreux services peuvent être fournis par un serveur MMS à un ensemble de clients de nombre limité. Ces services fournissent une gestion complète et distante pour l'utilisateur (c'est-à-dire le client), ils comprennent :

• **Découverte du modèle de données :** Description du modèle de données brut présent sur le serveur (définie par le fichier ".icd"). Chaque utilisateur devrait disposer du modèle de données dans un IED afin de gérer et de contrôler le périphérique.

• **Lire et / ou écrire des objets de données :** Activer la lecture / écriture sur une donnée nouvelle / existante présente sur le serveur.

• **Contrôler les dispositifs :** Les IED peuvent être contrôlés par un client (contrôle manuel distant) de différentes manières (direct ou sélection avant utilisation) à différents niveaux de sécurité (normal et renforcé).

• **Créer des ensembles de données (Data sets) :** Les ensembles de données sont utilisés pour rassembler des objets de données spécifiques souhaités qui seront utilisés ultérieurement dans les rapports ou simplement pour être lus collectivement.

• **Rapports :** Les rapports peuvent servir d'indications au client avec la possibilité de maintenir la séquence des événements. Ils permettent l'envoi des événements uniquement lors d'une requête (contrôlé par le RCB "Reports Control Block" : block de contrôle de rapports). Les rapports prennent également en charge la possibilité d'interroger des objets de données spécifiques à tout moment et de les recevoir périodiquement dans certaines conditions définies.

• **Manipulation des messages GOOSE :** Le client peut gérer le transfert des messages automatiques (les message GOOSE) dans un IED à partir d'un bloc de control GoCB (Goose Control Block) via une configuration dynamique des services de lecture et d'écriture MMS.

2.2.3 Structure de la pile MMS

Pour fournir la connectivité au réseau, un modèle de pile OSI est requis. Le protocole MMS est mappé à l'aide du modèle OSI illustré dans la figure 2.2, toutes les couches du modèle sont utilisées, de la couche d'application à la couche physique.

Chaque couche offre des protocoles différents, dans le cas du protocole MMS, nous avons la figure 2.3 qui décrit la pile entière.

MMS ne spécifie pas comment adresser les clients et les serveurs mais s'appuie sur le schéma d'adressage des protocoles sous-jacents. En pratique, les clients et les serveurs sont identifiés par leur adresse IP et le MMS est encapsulé sur TCP, port 102.

2.2.4 Mécanisme de transmission MMS

Comme mentionné dans la section précédente, MMS offre une communication unicast basée sur un modèle client-serveur. En outre, elle est mappée sur la pile OSI et utilise principalement le protocole TCP/IP (au sein de la couche de transport). Par conséquent, un mécanisme d'accusé de réception (Handshake Mechanism) existe.

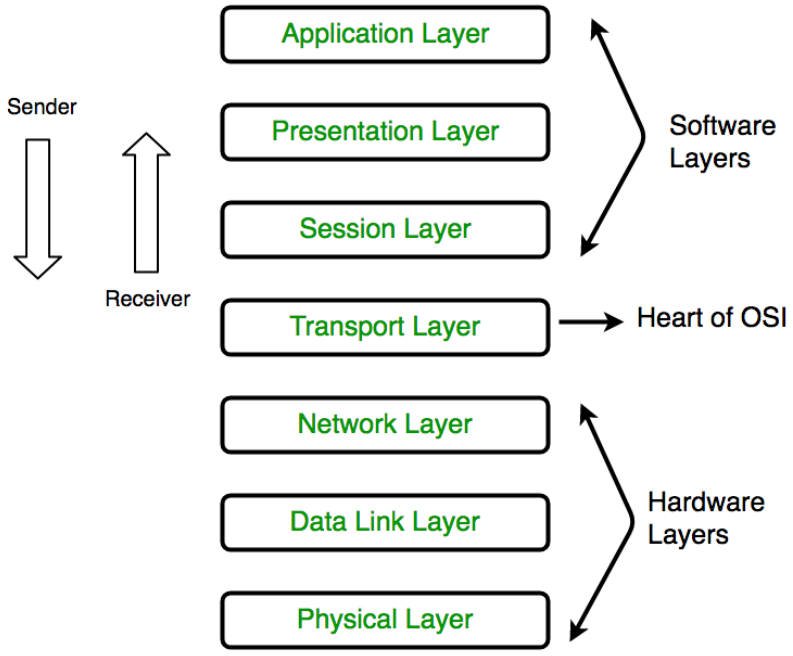

FIGURE 2.2 – Pile de modèle OSI

Source :https://contribute.geeksforgeeks.org/

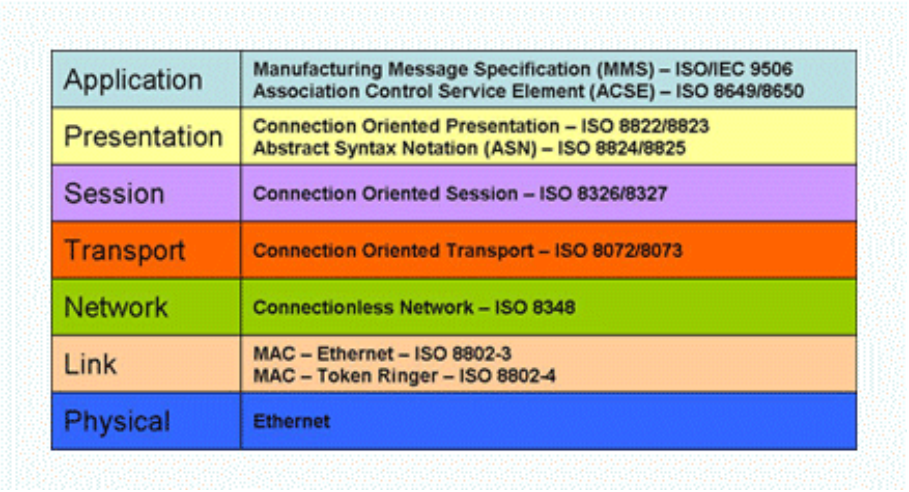

FIGURE 2.3 – La pile de communication MMS

Source :http://www.xelasenergy.com/

Les numéros de séquence TCP (SEQ) et d'accusé de réception (ACK) permettent un transfert de données fiable et ordonné pour les flux TCP. Le numéro de séquence est envoyé par le client TCP, indiquant la quantité de données envoyée pour la session (également appelé numéro de commande d'octet). Le numéro d'accusé de réception est envoyé par le serveur TCP, indiquant qu'il a reçu les données cumulées et qu'il est prêt pour le segment suivant.

Les numéros TCP (SEQ et ACK) sont coordonnés les uns avec les autres et constituent des valeurs clés lors de l'établissement de la liaison TCP, de la fermeture de TCP et, bien entendu, lors du transfert de données entre le client et le serveur.

Les unités PDU les plus fréquentes sont initiation-demande, initiation-réponse, confirmation-demande, confirmation-réponse, conclusion-demande, conclusion-réponse et non confirmée. Ces unités s'affirment bien avec le concept d'accusé de réception qui existe dans le protocole TCP/IP.[9]

La figure 2.4 montre un exemple de séquence TCP et de numéros d'accusé de réception dans un diagramme de flux TCP. La variable clé est la longueur du segment TCP pour chaque segment TCP envoyé dans la session.

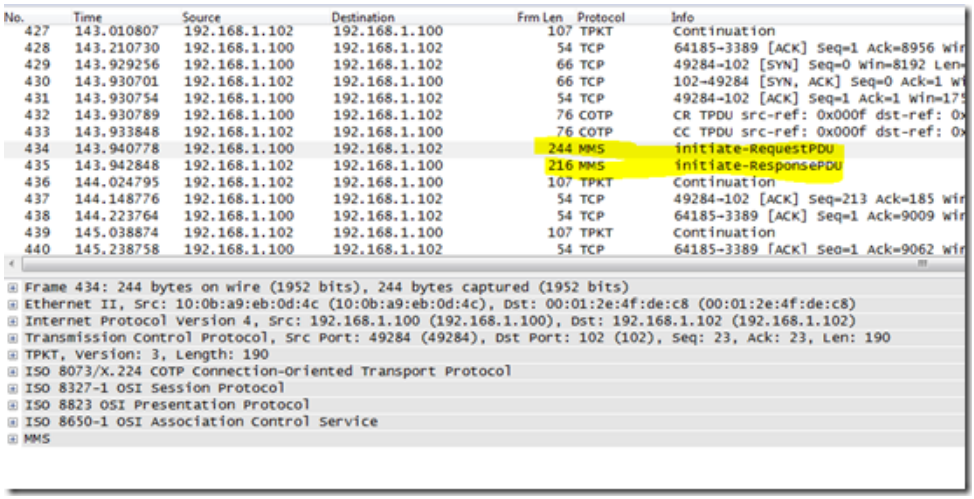

FIGURE 2.4 – Capture d'écran montrant le mécanisme de transmission MMS capturée par l'application Wireshark

Source :`https://pbs.twimg.com/`

2.2.5 L'anatomie d'un message MMS

Le format des PDU MMS est décrit à l'aide de la notation ASN.1 et codé à l'aide des règles de codage de base (BER) pour la transmission (voir Annexe A page 86). Dans la figure 2.5, la première ligne orange contient des informations telles que le type de protocole de messagerie (MMS), l'adresse IP du nœud de destination (200.129.11.14, l'adresse IP du nœud collecteur) et des informations de message (confirmation de demande).

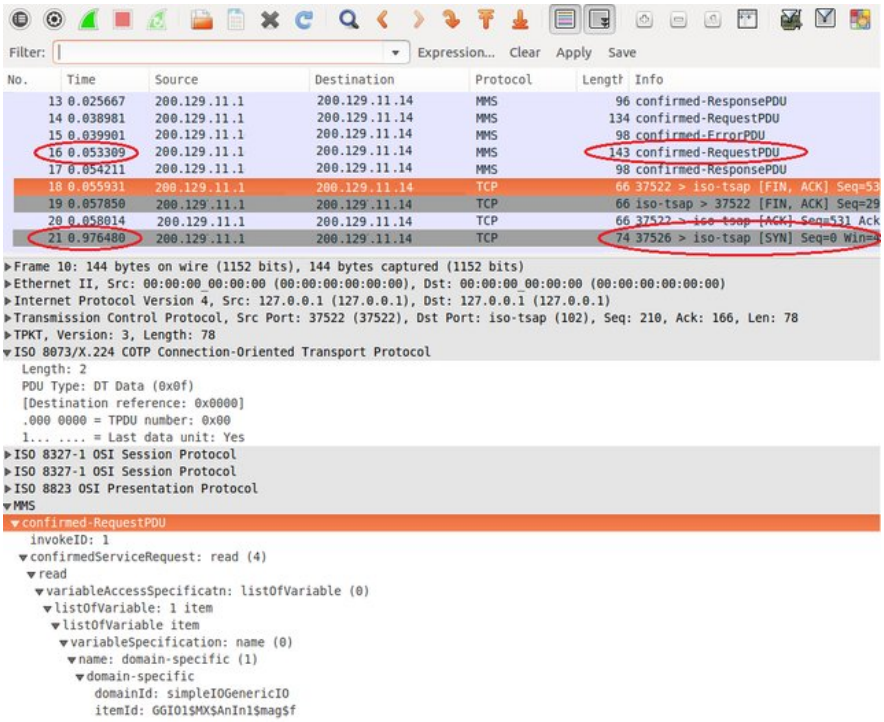

FIGURE 2.5 – Capture d'écran de la demande IEC 61850 confirmée capturée par l'application «Pcap»

Source :`https://www.researchgate.net/`

La capture d'écran montre une séquence de paquets contenant le numéro de séquence, l'heure de l'événement, les nœuds source et destination, le type de protocole, et informations de paquet. Les deux marques rouges dans les séquences 16 et 21 indiquent le début d'une demande et la réponse correspondante.[10]

2.3 Generic Oriented Object Substation Event (GOOSE)

2.3.1 Caractéristiques du protocole GOOSE

La messagerie GOOSE est un protocole d'échange de données rapide, non routable et fiable entre les IEDs (au niveau de la baie). Il définit un service générique d'événement de sous-station (GSE : Generic Substation Event) qui peut fournir une distribution rapide et fiable des valeurs de données d'entrée et de sortie, y compris des valeurs numériques ou analogiques. Le protocole GOOSE permet aux éditeurs / abonnés (i.e. IEDs) de communiquer via le réseau local LAN. La figure 2.6 décrit un schéma qui montre le modèle éditeur / abonné, tandis-que la figure 2.7 indique la différence entre un modèle client / serveur et un modèle éditeur / abonné.[11]

le mécanisme éditeur / abonné est une forme de communication asynchrone utilisée dans les architectures sans serveur. Dans un modèle éditeur / abonné l'éditeur envoie des messages de multidiffusion à tous les appareils qui lui sont connectés et à tout appareil doté d'un rôle d'abonné qui capte les messages qu'il souhaite recevoir. Tout message publié est immédiatement reçu par tous les abonnés. La messagerie éditeur / abonné peut être utilisée pour activer des architectures événementielles ou pour découpler des applications afin d'améliorer les performances, la fiabilité et l'évolutivité.

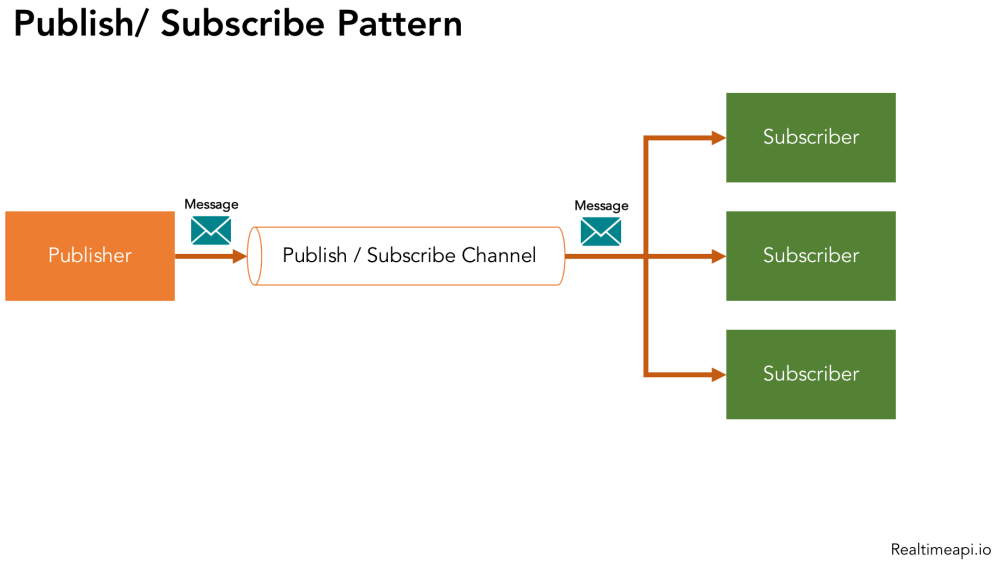

FIGURE 2.6 – Schéma qui montre le modèle Publisher/Subscriber

Source :`https://realtimeapi.io/`

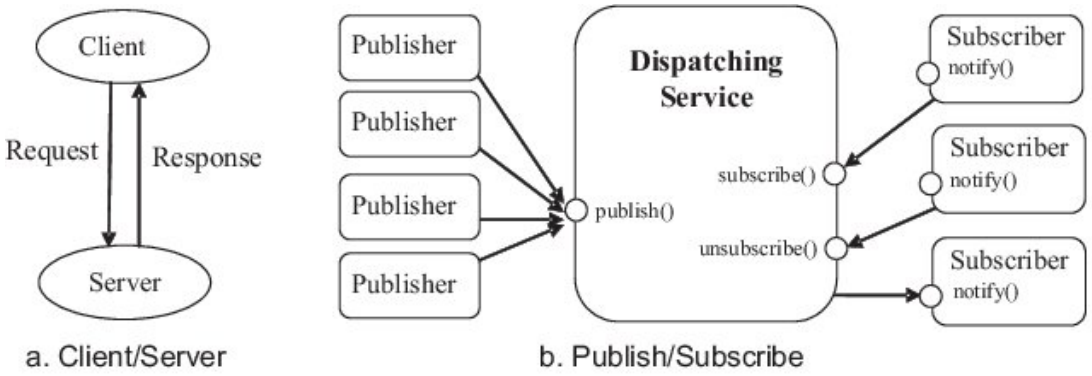

FIGURE 2.7 – La différence entre client/serveur et éditeur/abonné

Source :`https://www.researchgate.net/`

Le protocole GOOSE offre de nombreux avantages pour une communication fiable et sécurisée. Ils comprennent :

• Le protocole GOOSE utilise l'Ethernet standard pour la communication. À mesure qu'Ethernet évolue, les progrès de son développement se répercutent sur la communication GOOSE et en renforceront les avantages.

• Ethernet remplace la connexion en cuivre câblée point à point entre différents IED, ce qui induit moins d'encombrement.

• GOOSE n'utilise pas de mécanisme de prise de contact (Handshake Mechanism), les exigences en matière de rapidité des fonctions d'automatisation des stations sont donc améliorées.

• Le protocole GOOSE utilise le mode de communication multidiffusion qui permet à plusieurs IED de recevoir les mêmes données en même temps.

• Pour assurer le plus haut niveau de fiabilité, les messages GOOSE sont répétés tant que l'état persiste. Ces messages de pulsation sont envoyés en continu sur le réseau avec un temps de cycle long, ce qui garantit que les périphériques récemment activés connaissent les valeurs d'état actuelles de leurs périphériques homologues sur le réseau. Chaque paquet a une durée fixe autorisée à vivre (TAL : Time Allowed to Live) sur le réseau.

• Pour maximiser la fiabilité et la sécurité, le message GOOSE a un paramètre "temps de maintien" appelé "Hold Time", qui définit la durée de vie du message et ensuite expirera, sauf si le même message d'état est répété ou si un nouveau message est reçu avant l'expiration du délai du Hold Time.

• Un seul message GOOSE d'un IED individuel peut contenir toutes les données requises liées au système de protection, alors que l'approche câblée nécessite une connexion spécifique à une fonction et réduit donc le trafic réseau pendant les conditions de défaut.

• Les messages GOOSE utilisent également la configuration avancée des trames Ethernet telles que le réseau local virtuel (VLAN) et le balisage prioritaire. Cette priorité permet de filtrer le message, en particulier lorsque le trafic est élevé, ce qui peut se produire en cas de panne.

2.3.2 Utilisation du protocole GOOSE

Contrairement au MMS, le protocole GOOSE n'utilise pas d'accusé de réception lors de la communication. Le message est répété plusieurs fois selon une configuration prédéfinie afin de conserver le caractère d'urgence des messages GOOSE, qui est son objectif principal.

En outre, le GOOSE permet la transmission de données définies par des data sets, il est configuré de manière statique par un fichier ".icd" ou dynamiquement par manipulation du bloc de contrôle GOOSE mappé MMS via les services de lecture et d'écriture MMS comme était déjà mentionné.

Le message GOOSE n'est pas une commande dans le sens où il ne dit à aucun périphérique récepteur ce qu'il doit faire, mais qu'il indique simplement qu'un nouvel événement est survenu, en quoi consiste cet événement et à quel moment. L'utilisateur configure l'IED pour effectuer un ensemble d'actions en fonction du message GOOSE reçu.

2.3.3 Structure de la pile GOOSE

En raison de l'urgence des messages et de l'aspect temps réel, le protocole GOOSE est directement mappé sur la couche liaison (Ethernet) comme l'indique la figure 2.8. Cela réduit considérablement le temps de traitement, et également convient le mieux à une fonction de protection extrêmement critique dans la sous-station.

Comme le protocole GOOSE repose directement sur Ethernet, il ne s'applique qu'aux réseaux locaux LAN et n'est pas routable par les routeurs IP.

2.3.4 Mécanisme de transmission GOOSE

Afin de garantir la fiabilité et l'actualité de GOOSE, le mécanisme de retransmission de séquence basé sur la technologie de multidiffusion Ethernet est appliqué dans le cas d'une communication GOOSE.

Par rapport au MMS, GOOSE a deux caractéristiques remarquables :

• Sa rapidité d'exécution supérieure et sa grande efficacité qui réponds aux exigences définies dans la transmission d'informations liées à la protection des relais.

• Sa fonction d'émission et de réception multicast permet le partage des mesures (tensions / courants) ainsi que des dispositifs actionneurs (sectionneurs, disjoncteurs)

GOOSE transmet en réalisant des cycles d'adresses de multidiffusion Ethernet.

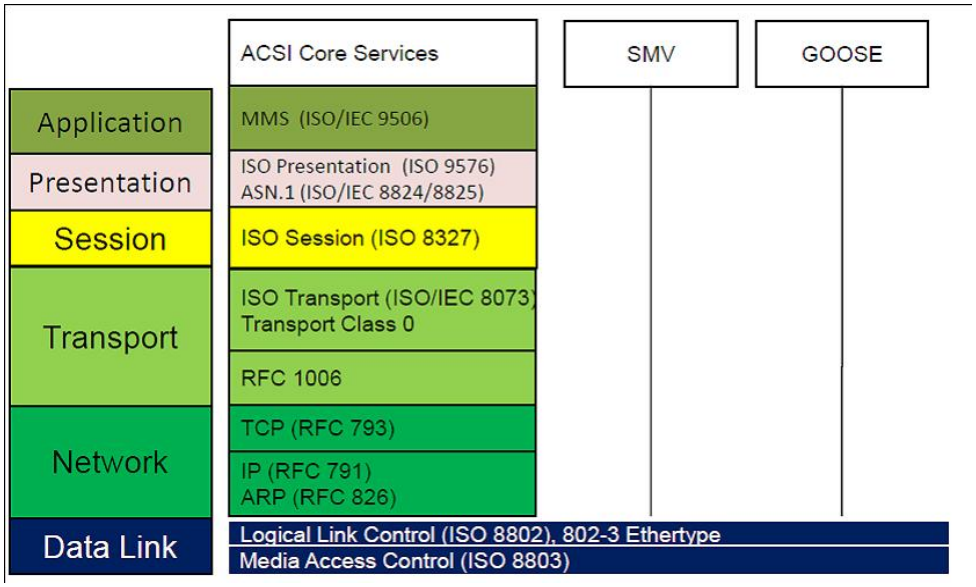

FIGURE 2.8 – Structure de la pile de communication GOOSE et SV

Source :`http://www.xelasenergy.com/`

Lorsque les données restent identiques, la transmission peut se produire rapidement trois fois en un cycle. GOOSE peut également transmettre l'ensemble des données en un seul cycle, comme indiqué dans la figure 2.9.

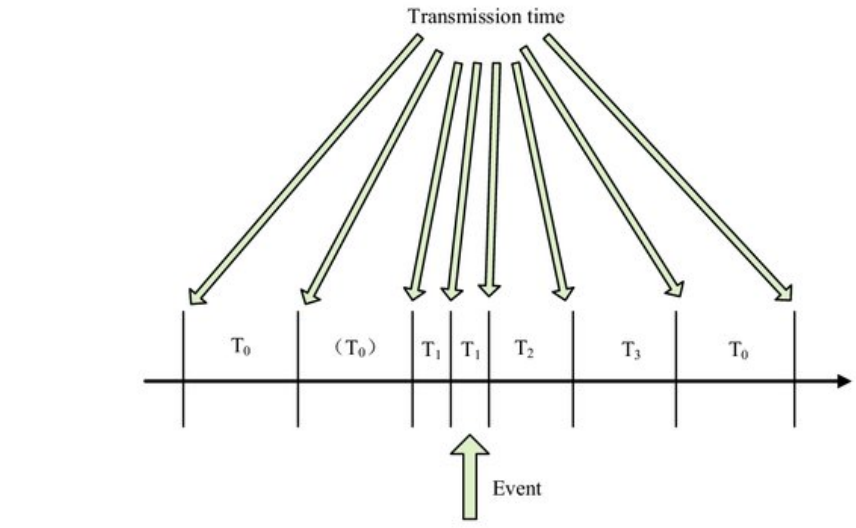

FIGURE 2.9 – Mécanisme de transmission d'un message GOOSE

Source :`https://www.researchgate.net/`

T_0 fait référence à l'intervalle de transmission stable dans le temps. Une fois que les données ont été modifiées, la mise à jour des données et la transmission répétée ont lieu quel que soit le mode de transmission de la trame précédente.

L'intervalle de retransmission et l'intervalle de pulsation les plus rapides peuvent être définis. Si T_0 est égal à 5s et T_1 à 1ms, les applications de verrouillage associées correspondant aux intervalles peuvent être étendues à 10 secondes et à 100ms. T_2 et T_3 font référence au temps de retransmission dans le processus d'un état fluctuant à un état stable. Aucune définition spécifique n'est donnée pour cette période.[12]

2.3.5 L'anatomie d'un message GOOSE

Comme le montre la figure 2.10, le datagramme GOOSE (i.e. PDU) commence par l'adresse MAC de destination, qui est une adresse de multidiffusion réservée aux applications IEC 61850 commençant toujours par 01-0c-cd, suivie d'une adresse MAC source de six octets. Ceci est l'adresse MAC de l'IED d'édition (Publisher). Un message GOOSE a un ID de réseau local virtuel (ID VLAN) IEEE 802.1Q et un type Ethernet unique (Ethertype) 88-B8.

Le champ ID d'application (APPID) est un champ de quatre octets que les IED abonnés utilisent pour identifier les messages auxquels ils s'abonnent. Le champ (Length) représente la longueur du datagramme GOOSE global moins huit octets et est suivi de deux champs réservés laissés de côté par la norme pour une utilisation future.

Le champ GOOSE PDU est lui-même composé de douze sous-champs qui suivent également le schéma de codage BER ASN.1 modifié. La PDU GOOSE comprend les éléments suivants :

- **gocbRef :** référence au bloc de contrôle GOOSE.
- **timeAllowedtoLive :** durée d'attente d'un destinataire avant de recevoir un message retransmis.
- **datSet :** nom du Data set associée.
- **goID :** ID de publication de l'IED.
- **t :** horodatage indiquant un nouvel événement GOOSE.
- **stNum :** compteur incrémenté à chaque événement GOOSE.
- **sqNum :** compteur incrémenté à chaque message GOOSE répété.
- **test :** spécifie si un message est / n'est pas destiné à être testé.
- **confRev :** nombre de fois que l'ensemble de données a été modifié.

- **ndsCom :** nécessite un champ de mise en service.
- **numDataEntries :** nombre d'éléments de données dans allData.
- **allData :** données réelles envoyées (bool, integer, float, etc.).[11]

Destination MAC Address		Source MAC Address		Priority Tagging/VLAN ID
Ethertype (88B8)		APPID		Length
Reserved 1		Reserved 2	Tag	Length	goosePDU
Tag	Length	gocbRef	Tag	Length	timeAllowedtoLive
Tag	Length	datSet	Tag	Length	goID
Tag	Length	t	Tag	Length	stNum
Tag	Length	sqNum	Tag	Length	test
Tag	Length	confRev	Tag	Length	ndsCom
Tag	Length	numDatSetEntries	Tag	Length	allData
Tag	Length	Data 1 (Boolean)	Tag	Length	Data 2 (Float)
●●●●●●	●●●●●●		Tag	Length	Data N

FIGURE 2.10 – Structure du datagramme GOOSE

Source :`https://www.researchgate.net/`

2.4 Sampled Values (SV)

2.4.1 Caractéristiques et utilisation du protocole SV

Le protocole SV est assez similaire au protocole GOOSE, il est directement lié à la couche liaison et utilise un modèle éditeur / abonné, dans lequel un éditeur transmet des données non acquittées à des abonnés. Les données SV transmises sont exclusivement des mesures numériques du courant et / ou de la tension des dispositifs d'alimentation électrique.

Le SV est principalement utilisé pour l'acquisition d'informations brutes, en particulier celles mesurées par les transformateurs de mesure. Les flux SV transportent des échantillons numérisés des mesures analogiques à une fréquence d'échantillonnage définie codée dans des trames Ethernet multidiffusion. Ces messages sont également critiques. Le « IED » spécial qui convertit les lectures analogiques des transformateurs d'instrument en paquets SV s'appelle une unité de fusion (MU).

Un paquet SV contient 8 valeurs instantanées : trois valeurs de phase et une de neutre pour la tension et le courant à un taux d'échantillonnage spécifié de 80 ou 256 échantillons par cycle[13]. Les éléments suivants définissent les caractéristiques principales du protocole SV :

• Les messages SV sont à temps critique, par conséquent, aucun accusé de réception n'est envoyé. Comme dans GOOSE, les couches liaison et application sont directement liées, ce qui améliore les performances temporelles du transfert de données. Cependant, contrairement à GOOSE, le même message n'est pas retransmis en SV.

• Le protocole SV permet au éditeurs de publier en continu les paquets de données à des débits spécifiques définis par l'utilisateur.

2.4.2 L'anatomie d'un message SV

Le message SV est une trame structurée contenant 4 mesures de courant et 4 mesures de tension. Les fréquences d'échantillonnage du signal de mesure doivent également être enregistrées. Elles ont la valeur de 4000 Hz (80 échantillonnages sur une période) pour la protection par relais et pour des objectifs de la comptabilité commerciale, et 12800 Hz (256 échantillonnages sur une période) pour le contrôle de la qualité de l'énergie. Une trame Ethernet normalisée doit alors être formée pour transmettre le message SV. Le format de la trame est illustré dans la figure 2.11.a.

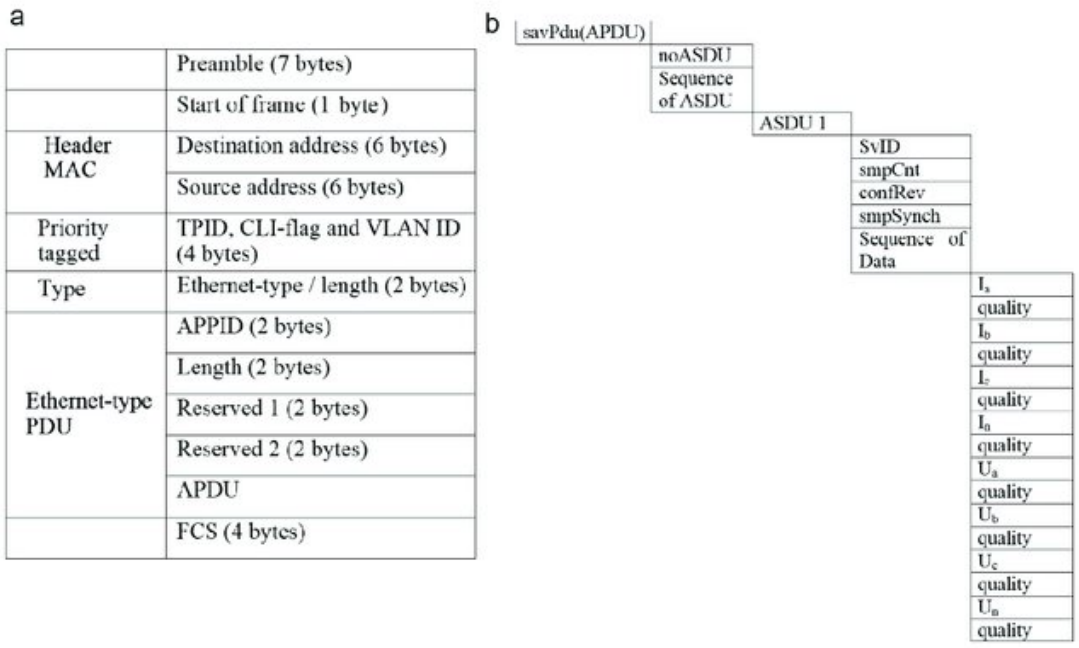

FIGURE 2.11 – Structure d'un message SV

Source :https://www.researchgate.net/

La structure du APDU est indiquée dans le format de la trame. Cette structure contient un groupe de mesures de courant et de tension. Il est représenté sur la figure 2.11.b pour la trame transmettant 80 échantillons sur une période (SV80). La différence entre les trames SV80 et SV256 réside dans le nombre d'APDU, présenté dans le message. Lorsque la fréquence d'échantillonnage est de 80 échantillonnant, il y a 1 ASDU dans le message, lorsque la fréquence d'échantillonnage est de 256 échantillonnant, il y a 8 ASDU dans le message.[14]

2.5 Conclusion

Dans ce chapitre, nous avons présenté la partie communication de la norme IEC 61850, en présentant ses trois protocoles par des différentes perspectives.

Dans le chapitre suivant, nous allons discuter les choix techniques (matériels et logiciels), qui sont pris en considération pour l'implémentation du nœud de communication IEC 61850.

Chapitre 3

Choix techniques

3.1 Introduction

L'implémentation d'un nœud de communication IEC 61850 (en tant que client ou un IED « serveur ou éditeur / abonné ») nécessite, d'une part, un matériel spécifique ayant la capabilité de prendre en charge une connexion Ethernet et d'élaborer un traitement multitâche à grande vitesse. D'autre part, elle a besoin aussi d'un logiciel dédié qui gère les applications en temps réel et qui correspond au matériel choisi afin d'obtenir de bonnes performances.

Dans ce chapitre, nous décrirons les choix techniques du matériel et du logiciel utilisés en se concentrant sur leurs caractéristiques.

3.2 Hardware

3.2.1 La carte de développement BeagleBone Black

La carte BeagleBone Black (BBB) a été sélectionnée comme contrôleur électronique principal qui gère les communications avec un IED (cas d'un serveur MMS ou éditeur / abonné GOOSE). En raison de ses spécifications, il s'agit d'un matériel approprié pour établir une communication IEC 61850 fiable et aussi rapide que souhaitée. La figure 3.1 montre la carte BeagleBone Black et ses caractéristiques matérielles.

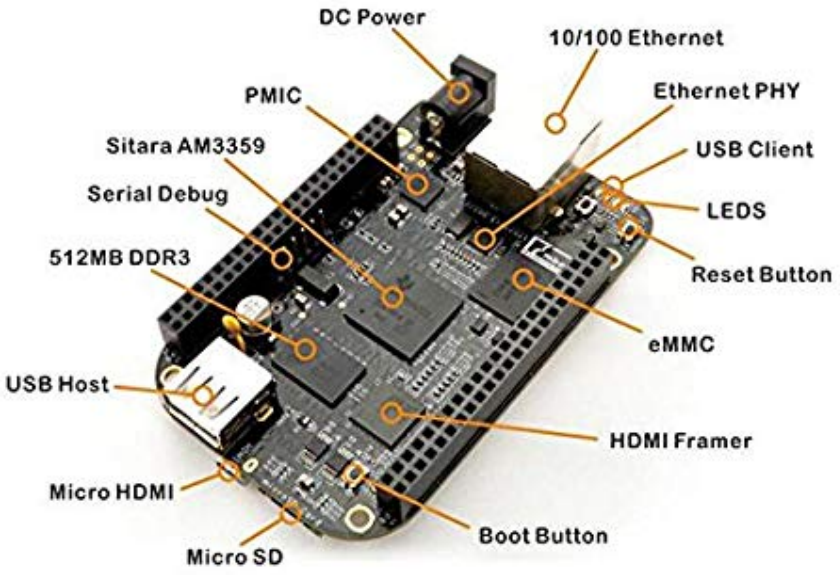

FIGURE 3.1 – Caractéristiques BeagleBone Black

Source :`https://beagleboard.org/`

3.2.2 Caractéristiques de la carte BeagleBone Black

BeagleBone Black est une plate-forme de développement peu coûteuse et prise en charge par la communauté pour les développeurs. Elle permet de démarrer Linux en moins de 10 secondes et peut lancer l'utilisateur sur le développement en moins de 5 minutes avec un seul câble USB.[15]

La carte présente les caractéristiques principales suivantes :

- 512 Mo de RAM DDR3
- 4 Go de stockage flash eMMC intégré 8 bits
- Accélérateur à virgule flottante NEON
- 2x microcontrôleurs 32 bits PRU

Caractéristiques de connectivité :

- Client USB pour l'alimentation et les communications
- USB host
- Ethernet connection
- Micro HDMI
- 2x 46 pin headers

Compatibilité logicielle

- Debian
- Android
- Ubuntu
- Cloud9 IDE sur Node.js avec la bibliothèque BoneScript

3.2.3 Description du microprocesseur AM335x 1GHz ARM Cortex-A8

La carte BeagleBone Black est équipée d'un processeur AM335x 1 GHz ARM Cortex-A8 qui prend en charge les systèmes d'exploitation de haut niveau avec des options d'interface industrielle améliorées.

Le microprocesseur AM335x possède les périphériques suivants :

- Deux ports USB 2.0 DRD (Dual-Role Device) haute vitesse avec PHY intégré
- Deux MAC industriels Ethernet Gigabit (10, 100, 1000 Mbps) prenant en charge les interfaces MII, RMII, RGMII et MDIO.
- Deux ports CAN (Controller-Area Network)
- Six UART

- Trois interfaces maître et esclave I2C
- Quatre banques de broches d'E / S à usage général (GPIO)
- Huit Timers à usage général 32 bits

Le microprocesseur AM335x contient les sous-systèmes présentés dans le diagramme de blocs fonctionnels avec une brève description de chacun dans la figure 3.2.

En outre, il possède un Sous-système d'unité programmable en temps réel et sous-système de communication industrielle (Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem « PRU-ICSS ») qui sont séparés du cœur ARM, permettant un fonctionnement et une synchronisation indépendants pour une efficacité et une flexibilité accrues. Le PRU-ICSS permet des interfaces de périphériques supplémentaires et des protocoles en temps réel tels que EtherCAT, PROFINET, EtherNet / IP, PROFIBUS, Ethernet Powerlink, Sercos, etc.

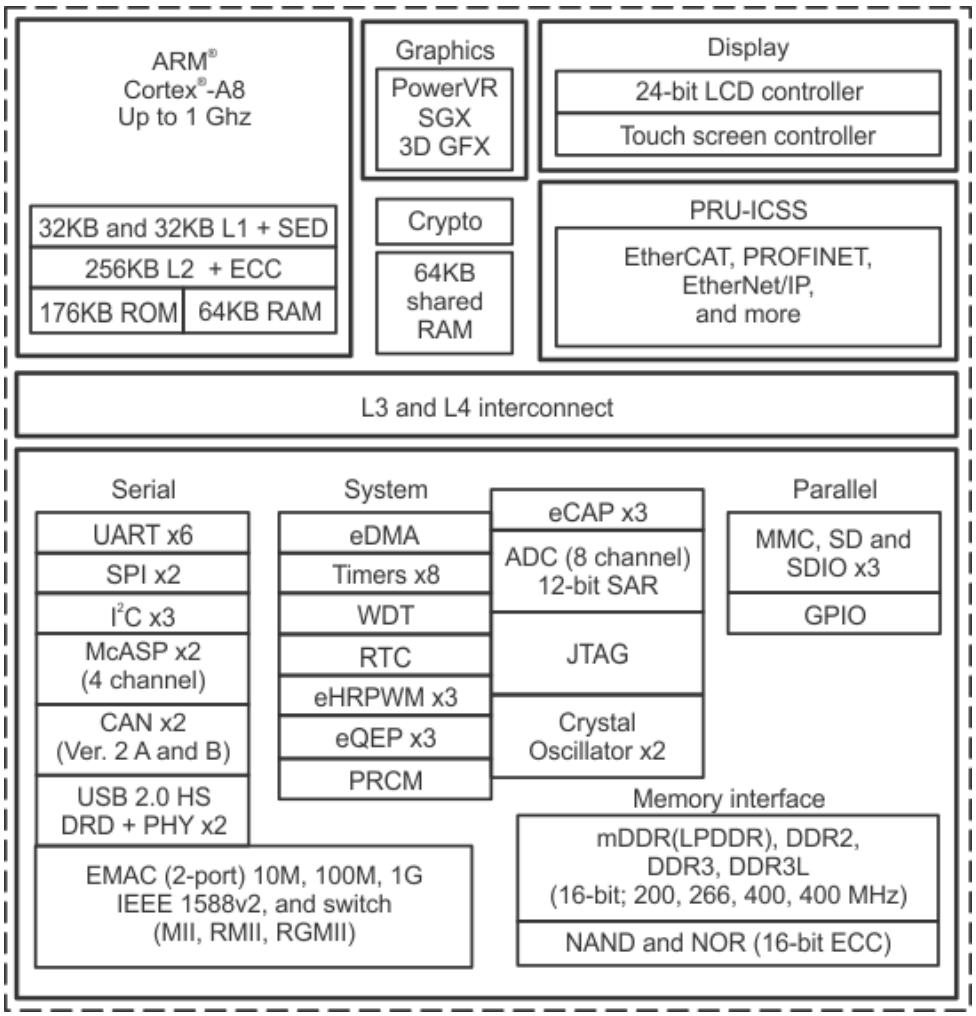

FIGURE 3.2 – Diagramme de blocs fonctionnels du microprocesseur AM335x

Source :`http://www.ti.com/`

De plus, la nature programmable du PRU-ICSS, ainsi que son accès aux broches (pins), événements et à toutes les ressources système sur puce (SoC), offre une flexibilité dans la mise en œuvre de réponses rapides et en temps réel, d'opérations de traitement de données spécialisées et d'interfaces périphériques personnalisées.[16]
Plusieurs périphériques existent à l'intérieur du PRU-ICSS dont on site :

• Un port UART avec broches de contrôle de flux, prend en charge jusqu'à 12 Mbps

• Un module Enhanced Capture (eCAP)

• Deux ports Ethernet MII prenant en charge Industrial Ethernet, tels que EtherCAT

• Un port MDIO

• Deux Programmable Real-Time Units (PRUs)

3.3 Software

3.3.1 La bibliothèque libIEC61850

libIEC61850 est un projet Open Source fournissant une bibliothèque pour les protocoles de communication IEC 61850 / MMS, IEC 61850 / GOOSE et IEC 61850 / Sampled Values écrits en langage C et elle contient aussi une version compilée développée en Python.

Cette bibliothèque fournit une implémentation des données et des fonctions du standards IEC 61850. En plus du protocole MMS, elle prend également en charge la communication intra-sous-station via GOOSE. La bibliothèque contient également un wrapper .NET pour permettre à la bibliothèque d'être facilement utilisée dans des langages de haut niveau tels que C #.

Cette bibliothèque peut fonctionner sur des systèmes embarqués, des systèmes Linux embarqués ainsi que sur des ordinateurs de bureau exécutant Linux, Windows ou MacOS.

3.3.2 Fonctionnalités de la bibliothèque libIEC61850

La dernière version de libIEC61850 (version 1.3.3) contient de nombreuses améliorations de la stabilité et corrections de bugs, elle offre les fonctionnalités suivantes :

• Pile de protocoles ISO complète au-dessus de TCP / IP

• Implémentation statique du modèle IED par génération de code C à partir d'un fichier SCL

• Création dynamique du modèle d'IED par appels API ou par fichier de configuration

• Services de lecture et d'écriture pour les variables MMS simples et complexes

• Services de navigation de modèles (GetServerDirectory, GetDeviceDirectory…)

• Services de Data sets, y compris la création et la suppression dynamiques

• Manuel de référence de l'API

• Service de rapport mis en tampon et sans tampon (Bufferd and Unbufferd Reports)

• Tous les modèles de contrôle IEC 61850 sont pris en charge par le serveur et le client

• Code d'éditeur et d'abonné GOOSE pouvant également être utilisé de manière autonome.

• Couche d'abstraction matérielle et implémentations pour les systèmes de style BSD POSIX (Linux), WIN32, (Mac OS X, FreeBSD)

• Outil de conversion java permettant de convertir les fichiers SCL en modèles IED statiques (deux fichiers : « .c » et « .h ») ou en fichiers de configuration du serveur (fichier « .cfg »).

• Convient également pour une utilisation dans des systèmes embarqués à contraintes de ressources.

libIEC61850 prend en charge deux systèmes de building différents. Le système traditionnel basé sur `make` qui fonctionne bien avec le toolchain basée sur GNU et le nouveau système basé sur `cmake`. Avec le système de building basé sur `cmake`, il est également possible d'effectuer des constructions avec Visual Studio ou d'autres toolchain, en particulier pour la plate-forme Windows. Jusqu'à présent, le système `make` est requis pour la compilation croisée (cross-compiling) pour les systèmes embarqués Linux.[17]

3.3.3 Manuel de référence de l'API pour libIEC61850

L'API de libIEC61850 est divisée en quatre parties : client, serveur, éditeur / abonné et une partie commune. Les trois premières parties partagent également les éléments communs de la quatrième partie. En outre, il existe deux API différentes pour le client et le serveur. Il existe une API MMS "de bas niveau" et une API de niveau supérieur, IEC 61850.

3.3.3.1 IEC 61850 Client et MMS Client API

L'API client IEC 61850 est une API de haut niveau permettant d'accéder à des périphériques conformes à IEC 61850. L'API permet d'accéder aux services MMS pris en charge par la norme IEC 61850. D'autre part, l'API client MMS n'est pas spécifique à IEC 61850, mais il s'agit d'une API client MMS générique qui fournit uniquement les fonctionnalités requises par le mappage MMS IEC 61850. Cette API peut être utilisée si un accès de bas niveau aux services MMS est requis.

Éventuellement les deux API fournissent de nombreuses fonctions utilisées pour une implémentation côté client, ces fonctions sont regroupées comme suit :

- Fonctions générales de traitement des connexions et types de données.
- Fonctions de gestion des blocs de contrôle SV.
- Fonctions de gestion des blocs de contrôle GOOSE
- Services de gestion de rapports, fonctions et types de données liées
- Fonctions de service d'accès aux données (lecture / écriture)
- Fonctions de service et types de données du Data sets
- Fonctions du service de contrôle
- Services de découverte de modèles
- Fonctions des services de journalisation, des types de données et des définitions liés.
- Fonctions, types de données et définitions liés au service de fichiers

3.3.3.2 IEC 61850 Server API et MMS Server API

La prise en charge du serveur pour IEC 61850 inclut la génération du modèle d'appareil MMS à partir du modèle de données statique IEC 61850. L'API du serveur IEC 61850 prend également en charge le modèle de contrôle IEC 61850. Elle est conçue pour générer une très faible surcharge et peut être utilisée pour créer de très petites applications serveur MMS conformes à la norme IEC 61850. Le serveur n'analyse pas le fichier SCL (XML) au moment de l'exécution. Cependant, le fichier SCL sera analysé au moment de la construction par un outil qui génère du code C pour la définition du modèle de données IEC 61850 et d'autres paramètres du fichier SCL. Cela libérera le code serveur de la charge d'un analyseur de fichiers XML. De plus, le système cible ne nécessite pas de système de fichiers pour stocker le fichier SCL.

L'API du serveur MMS fournit une API MMS générique. Les fonctions spécifiques à la IEC 61850 ne sont pas prises en charge dans cette API. En cas d'implémentation de périphériques conformes à la norme IEC 61850, l'API du serveur IEC 61850 devrait être utilisé.

Les fonctions API utilisées pour une implémentation côté serveur sont regroupées et elles comportent :

- Fonctions de gestion et de configuration générale du serveur
- Gestion de la connexion et authentification du client
- Accès au modèle de données et mise à jour des données
- Gestion des groupes de paramètres
- Traitement du modèle de contrôle
- Traitement du bloc de contrôle des valeurs échantillonnées côté serveur (SVCB)
- Gérer l'accès externe au modèle de données et au contrôle d'accès
- Définitions générales des modèles de données, fonctions d'accès et d'itération
- Fonctions générales de création de modèles dynamiques
- Créer des modèles de données à l'aide de fichiers de configuration
- Fonctions d'assistance pour créer des classes de données communes (CDC) à l'aide de l'API de modèle dynamique
- Interface de fournisseur de service (SPI) pour les implémentations de stockage de journaux

3.3.3.3 IEC 61850 GOOSE et SV API

Les API GOOSE et SV sont également divisées en deux parties, une partie éditeur, qui inclut un ensemble de fonctions responsables de la configuration de base et permettant la publication de messages (messages GOOSE ou SV). Les principales fonctions utilisées pour implémenter un éditeur IED sont :

• `GoosePublisher_create` : elle crée une instance d'éditeur.

• `GoosePublisher_publish` : permet à un éditeur de commencer à envoyer (c'est-à-dire à publier) des messages.

• `GoosePublisher_destroy` : libère la mémoire des instances d'éditeur inutilisées.

D'autre part, la partie de l'abonné présente quelques différences. En plus du fait d'avoir une instance d'abonné qui se connecte à l'éditeur, une autre instance a lieu, elle s'appelle Receiver (un récepteur). Un Receiver est responsable du traitement de tous les messages GOOSE / SV pour une seule interface Ethernet. Un objet `Goose/SVSubscriber` est associé à un flux de données GOOSE / SV identifié par son APPID et son adresse Ethernet de destination. L'objet `Goose/SVSubscriber` est utilisé pour installer un gestionnaire de rappel (callback function) `Goose/SVUpdateListener` qui est appelé pour flux reçue.

Similairement à un éditeur, un récepteur et un abonné sont associés à un ensemble de fonctions dont on mentionne :

• `Goose/SVSubscriber_setListener` : définit une fonction de rappel qui sera appelée à la réception d'un message GOOSE / SV.

• `Goose/SVReceiver_start` : démarre le récepteur GOOSE / SV dans un thread séparé.

• `Goose/SVReceiver_addSubscriber` : Ajoutez un abonné à cette instance de destinataire. Elle ne peut être appelé qu'avant la fonction de démarrage "Start".

3.3.3.4 IEC 61850 Common Parts API

La libIEC61850 possède des fonctions de parties communes utilisées lors de la mise en œuvre d'un client / serveur ou d'un éditeur / abonné IEC61850. Ces fonctions peuvent être classées en deux groupes :

Un groupe qui rassemble les fonctions liées aux conteneurs de données, notamment :

• Définition du type de données `MmsValue` et ses fonctions de traitement : Cette partie définit une structure complexe appelée `MmsValue` qui contient différents types de variables de données (Bool, Integer, Float, etc.). Elle a ses propres fonctions permettant de gérer un tel type (ex. `MmsValue_create` : pour créer une instance de `MmsValue`, `MmsValue_getInt32` : utilisée pour extraire un entier de 32 bits détenu par l'instance de `MmsValue`).

• Définition du type de données `LinkedList` et ses fonctions de traitement : Les listes chaînées sont principalement utilisées pour les ensembles de données, elles peuvent accepter tout type de variable (ex. Entier, flottant, chaîne, etc.). En outre, elle a des fonctions spécifiques telles que `LinkedList_add` pour ajouter une nouvelle variable de données et `LinkedList_contains` utilisé pour vérifier si une donnée particulière est incluse dans une liste chaînée.

• Spécifications du type de données `MmsVariableSpecification` : Gère un type `MmsVariableSpecification` qui, comme un type `MmsValue`, permet de créer une copie d'une variable `MmsValue` existante ayant les mêmes spécifications (ex. type de données, taille, valeur, etc.).

Le deuxième groupe englobe des différentes fonctions liées aux spécifications de la norme IEC 61850 :

• Options de déclenchement : Utilisées dans les rapports (ex. déclenchement périodique).

• Options de rapport : Par exemple : raison de l'inclusion d'une donnée dans un rapport.

• Originator categories (orCat) : Pour informer sur l'origine d'un contrôle.

• Définition pour le type "addCause" : Utilisé dans les modèles de contrôle.

• Définitions et fonctions liées aux contraintes fonctionnelles (FC) : Utilisées pour afficher une FC (fonction : `FunctionalConstraint_toString`) ou pour analyser une chaîne traitée comme une représentation FC (ex. la fonction : `FunctionalConstraint_fromString`).

• Définitions et fonctions liées à la qualité des attributs de données : Permettent de vérifier l'état de qualité d'un attribut de données (ex. GOOD, BAD, etc.).

• Définitions et fonctions liées au type de données Horodatage (heure UTC) IEC 61850 : Utilisé pour gérer l'heure et la synchronisation (fonctions telles que `Timestamp_getTimeInSeconds`, `Timestamp_setClockNotSynchronized`, etc.)

• Définitions et fonctions liées au type de données IEC 61850 Dbpos (un CODED ENUM)

3.4 Conclusion

Dans ce chapitre, nous avons d'abord introduit le BeagleBone Black qui hébergeait la partie de communication IED choisie en raison de ses caractéristiques de traitement des threads multiples, de tâches en temps réel et de diverses capacités de communication via un port Ethernet. Après cela, nous avons présenté les différentes fonctionnalités du logiciel open source (libIEC61850) en expliquant les parties des API utilisées comme référence initiale sur laquelle nous nous sommes basés.

Dans le chapitre suivant, nous présenterons l'implémentation de notre nœud de communication IEC 61850 divisée en différentes parties en fonction du type de communication.

Chapitre 4

Implémentation

4.1 Introduction

L'implémentation de l'interface de communication IEC 61850 consiste à travailler sur deux côtés séparés. Le premier côté est lié au niveau de la station, où une application basée sur un ordinateur de supervision et de contrôle existe (application coté client). L'autre côté comprend une application au niveau de la baie destinée à gérer le comportement des IED (application serveur et éditeur / abonné).

Dans ce qui suit, nous allons montrer les étapes suivies pour l'implémentation des deux cas discutés ci-dessus.

4.2 L'implémentation de l'application client IEC 61850

Une application cliente pour un environnement IEC 61850 appartient au niveau de la station. Elle est utilisée pour demander différents types de requêtes à un serveur.

Au début, le client doit être configuré pour se connecter à un serveur (ou à plusieurs serveurs à la fois) en saisissant l'adresse IP de l'IED souhaité (serveur). Ce processus nécessite un ensemble d'instances définies de libIEC61850 liées aux connexions client du réseau. Les instances les plus importantes sont `IedConnection` utilisé pour créer une structure client contenant une seule connexion IED (serveur) et `IedClientError` qui informent sur l'occurrence d'erreur en cas d'échec de la connexion à un IED. De plus, il existe des fonctions de contrôle permettant le traitement des connexions créées. La fonction `IedConnection_connect` (ou "méthode" si elle est observée du point de vue de la programmation d'objet orienté) prend quatre arguments, les deux premiers sont les instances `IedConnection` et `IedClientError`, le troisième est l'adresse IP de destination et le dernier est le port de connexion, qui est généralement le port TCP : 102. Si une erreur apparaît lors de l'établissement de la connexion (serveur indisponible ou adresse IP incorrecte), la fonction `IedConnection_connect` modifie la valeur du deuxième argument (`IedClientError`) en un motif (macro) d'erreur correspondant, sinon cet argument est égal au macro `IED_ERROR_OK` qui indique que la connexion a abouti.

Après avoir établi une connexion avec réussite à un serveur, le client peut commencer à demander des requêtes. Les messages créés sur le réseau pendant la communication client-serveur sont codés à l'aide du codage ASN 1 et BER. La bibliothèque libIEC61850 fournit des fonctions qui permettent le codage des messages envoyés (telle que la fonction `ber_encodeAsn1PrimitiveValue`). Ces fonctions sont

appelées lors de l'envoi de messages. Pour les messages reçus, d'autres fonctions sont impliquées dans le processus de décodage (telle que la fonction `ber_decode`).

4.2.1 Application CLI pour le client IEC 61850

L'application CLI (Command Line Interface) du client IEC 61850 a été développée à l'aide de la bibliothèque libIEC61850 mentionnée dans le chapitre 3. Elle permet à son utilisateur d'interagir avec différents services fournis par n'importe quel serveur IEC 61850 conformément à la norme. Il existe six services que l'utilisateur de l'application peut demander à un serveur.

4.2.1.1 Le service : Get server data model

Il permet à l'utilisateur de découvrir le modèle de données présent sur le serveur (IED) afin de disposer de l'architecture de données et de réaliser diverses actions telles que la lecture, l'écriture, le contrôle, etc.

Dans l'application développée, l'action "Get server data model" affiche les différentes LN disponibles et permet à l'utilisateur de choisir l'un d'entre eux afin d'afficher les données qu'il contient (DO, DA, RCB, GoCB, etc.) classées selon leur FC.

Pour ce service, un ensemble de fonctions spécifiques est utilisé, principalement la fonction `IedConnection_getDeviceModelFromServer` qui récupère le modèle de données sur le serveur et le stocke dans l'instance `IedConnection` appartenant au client. De plus, la fonction `IedConnection_getLogicalNodeVariables` permet le traitement des données sous forme de listes chaînées afin de les afficher d'une manière formelle.

La figure 4.1 montre la réponse d'un serveur IEC 61850 (fournie par la bibliothèque libIEC61850) à la requête "Get server data model". Dans la figure 4.2, on voie la réponse du server avec les données (DO et DA) classée selon leurs FC.

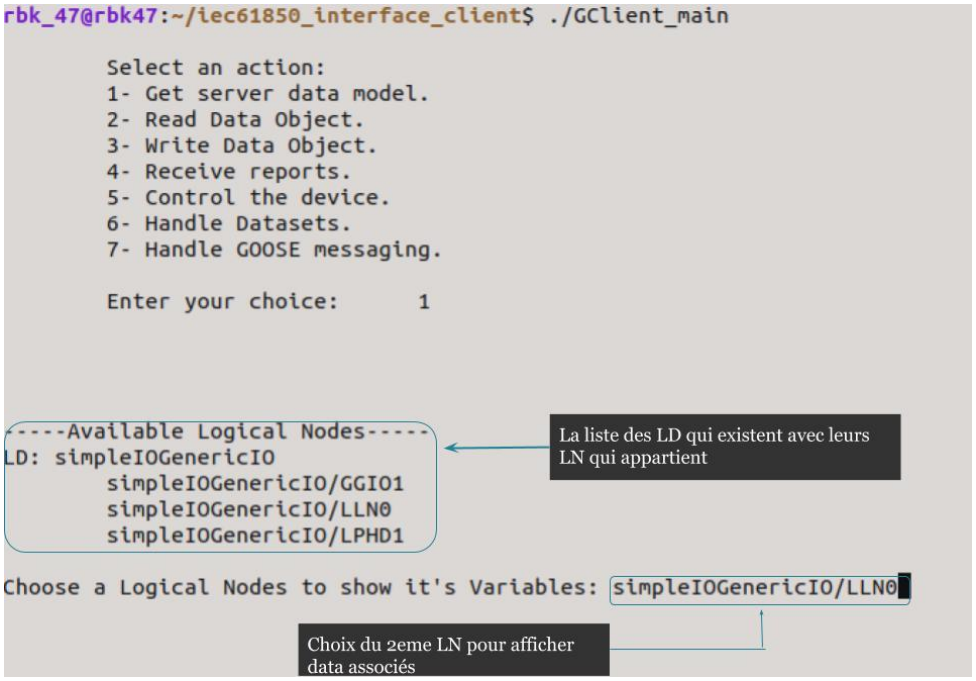

FIGURE 4.1 – Réponse du serveur à la requête "Get server data model"

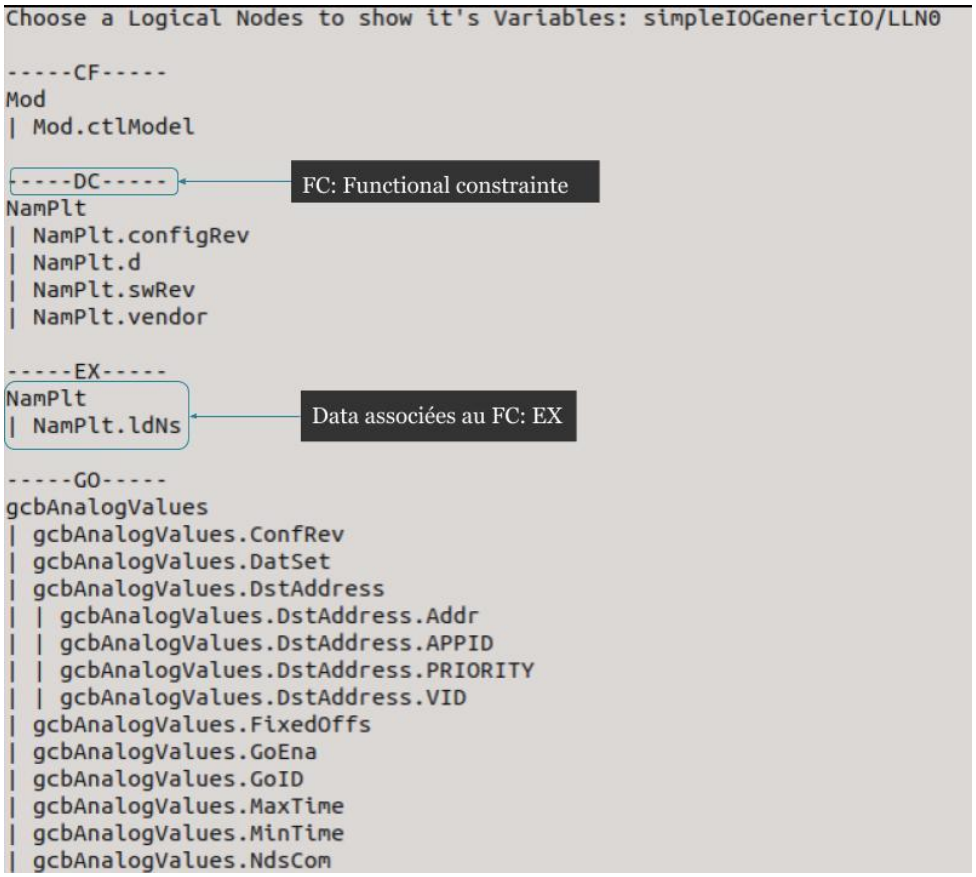

FIGURE 4.2 – Réponse du server après le choix du LN

4.2.1.2 Le service : Read/Write Data Object

Selon le type de DA, l'utilisateur est informé de l'état des composants électrotechniques du IED, ainsi que des valeurs de mesure analogiques de la tension et / ou du courant.

L'option "Read Data Object" demandera à l'utilisateur de saisir une référence de donnée sous le nom "LD/LN.DO.DA". Une fois la référence saisie, une demande de lecture est envoyée au serveur. Si la référence entrée existe, le serveur répond par la valeur de donnée en fonction de son FC, sinon un message d'erreur est affiché sur l'écran.

La fonction qui prend en charge le processus de lecture est `IedConnection_readObject`, elle prend comme argument principal le nom de donnée à lire et son FC et renvoie une instance `MmsValue` qui contiendra la valeur de donnée correspondante.

La figue 4.3 montre la réponse du serveur à une demande de lecture. Le serveur répond par le FC de data entée en référence et sa valeur.

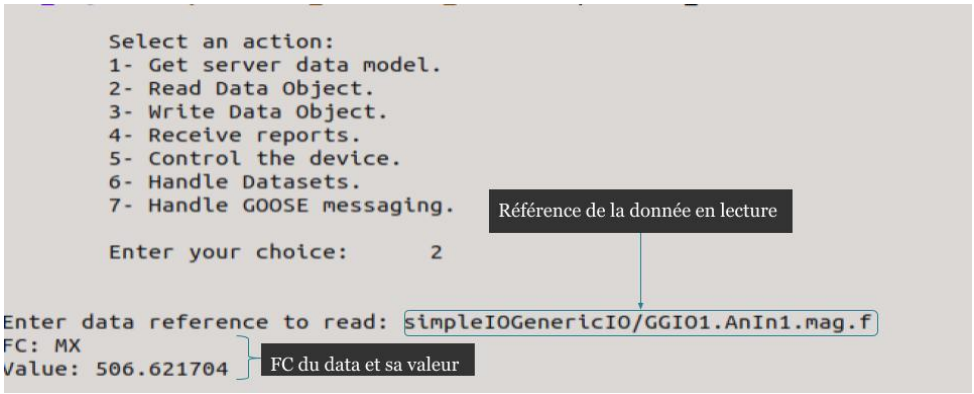

FIGURE 4.3 – Réponse du serveur pour une demande de lecture

Le mécanisme de la demande d'écriture est similaire, il est demandé à l'utilisateur de saisir une référence de données afin de lui donner une nouvelle valeur. Cependant, certaines données ne sont pas inscriptibles en raison d'une restriction d'accès en écriture afin d'empêcher les erreurs de fonctionnement.

Pour les fonctions utilisées dans le processus d'écriture, elles font partie des fonctions `MmsValue_set`. Selon le type de valeur DA, la fonction pertinente est appelée avec le type de données approprié (Boolean, Integer, Float, etc.).

Dans la figure 4.4, l'utilisateur change un DA de type chaîne de caractères (Vendor Name), et par suite vérifie que le changement est bien établi.

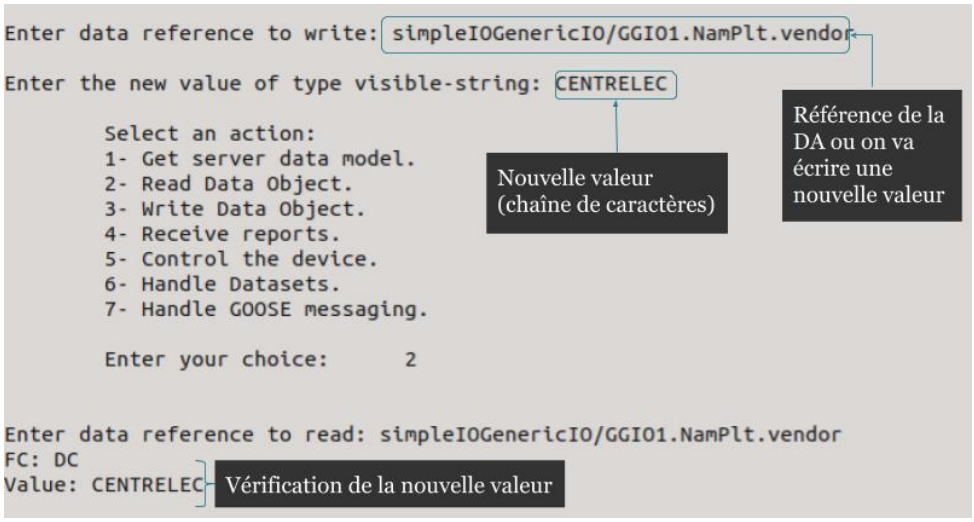

FIGURE 4.4 – Processus d'écriture à partir de l'application cliente

4.2.1.3 Le service : Receive reports

La réception de rapports est plus complexe que la lecture d'un seul objet point de vu logique de programmation, mais elle est très utile pour éviter plusieurs demandes de lecture. Elle permet à un serveur d'envoyer des données en fonction des événements et sans demande explicite du client. Les données envoyées et les événements à l'origine des rapports sont configurés via des blocs de contrôle de rapport (RCB).

La sélection de l'option « Receive reports » demande à l'utilisateur de sélectionner un RCB qui enverra les rapports. Le RCB est toujours associé à un ensemble de données spécifique qui doit être situé sur le même nœud logique que le RCB. De plus, un RCB a sa propre pré-configuration définie sur le serveur. Si la sélection du RCB a réussi (RCB réservé), une nouvelle instance appelée `ClientReportControlBlock` est créée et contient les valeurs de lecture du RCB existant sur le serveur. De plus, de nouvelles options sont disponibles. Ces options comprennent l'affichage de la configuration RCB, l'émission d'une interrogation générale, la définition des options de déclenchement, la définition de la période d'intégrité, etc. Chaque option, lorsqu'elle est sélectionnée, appelle un ensemble de fonctions qui permet la gestion en vertu des besoins du service.

Il existe de nombreuses fonctions développées à utiliser avec les rapports. Les plus importantes sont : `IedConnection_getRCBValues` qui obtient la configuration RCB et le groupe de fonctions `ClientReportControlBlock_set` qui permet de modifier les valeurs de configuration RCB (telles que l'option de déclenchement, la période d'intégrité, etc.).

Dans la figure 4.5, on voie les différentes options que le service des rapports offre. La figure 4.6 montre les valeurs d'un RCB après avoir choisie la réception d'un rapport en interrogation générale, on voie que les rapports sont activés ainsi que l'option de déclenchement est réglée. On peut aussi avoir le nom de data set associée à ce rapport.

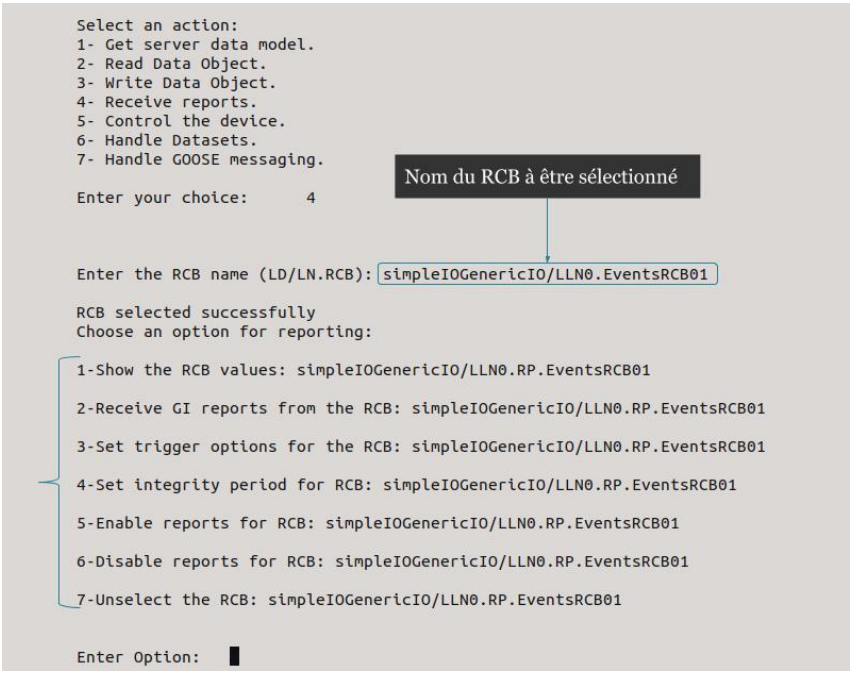

FIGURE 4.5 – Réponse du serveur pour l'option "Receive reports"

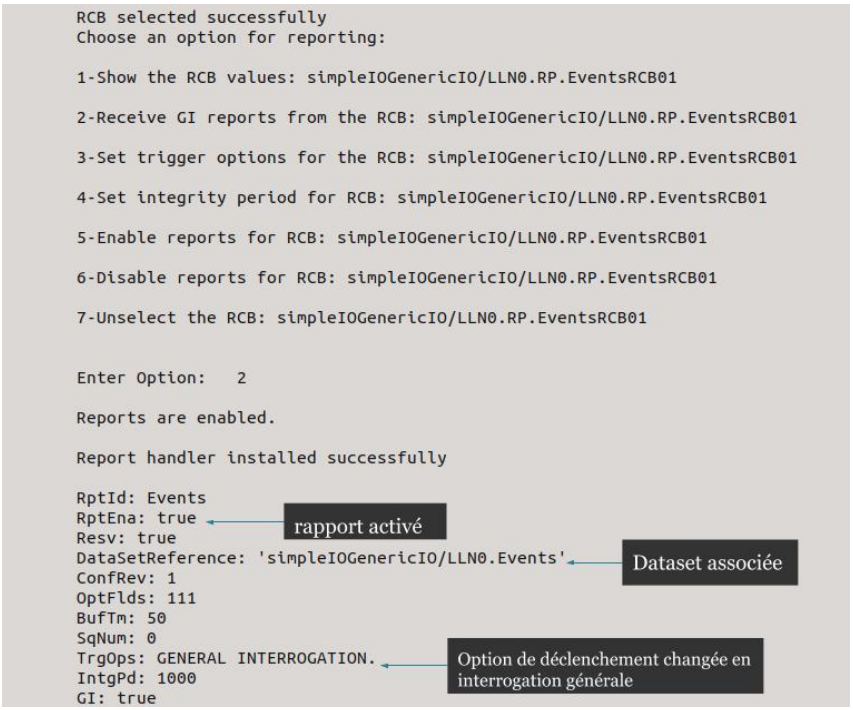

FIGURE 4.6 – Illustration de réception d'un rapport GI

4.2.1.4 Le service : Control the device

Le contrôle du IED par le client est possible en traitant des DO ayant un FC spécifique défini comme "CO" (Contrôle). Ce type de DO a un champ spécial appelé "ctlModel" qui indique le type de contrôle qui peut être appliqué à ce DO (contrôle direct avec une sécurité normale, contrôle direct avec une sécurité renforcée, contrôle SBO « sélectionné avant utilisation » avec une sécurité normale et contrôle SBO avec sécurité renforcée). Une fois que le DO contrôlable est sélectionné avec succès, un ensemble d'options est affichées et cela change par rapport à la valeur "ctlModel". Les options disponibles sont "Statut des données" qui affichent le statut des données sélectionnées (valeur) et l'un des types de contrôle mentionnés ci-dessus. L'utilisateur a la possibilité de sélectionner l'option, qui à son tour appelle la fonction correspondante.

Ces fonctions correspondantes incluent le groupe de la classe `ControlObjectClient`, qui fournit différentes méthodes (telles que créer, utiliser, sélectionner, etc.).

La figure 4.7 montre une lecture de l'état d'un DO, dans ce cas il a un type booléen avec une valeur de « 0 » (vu comme état ouvert d'un disjoncteur associé à ce DO). Dans la figure 4.8, l'utilisateur contrôle ce DO en mettant sa valeur à « 1 » (i.e. état fermé).

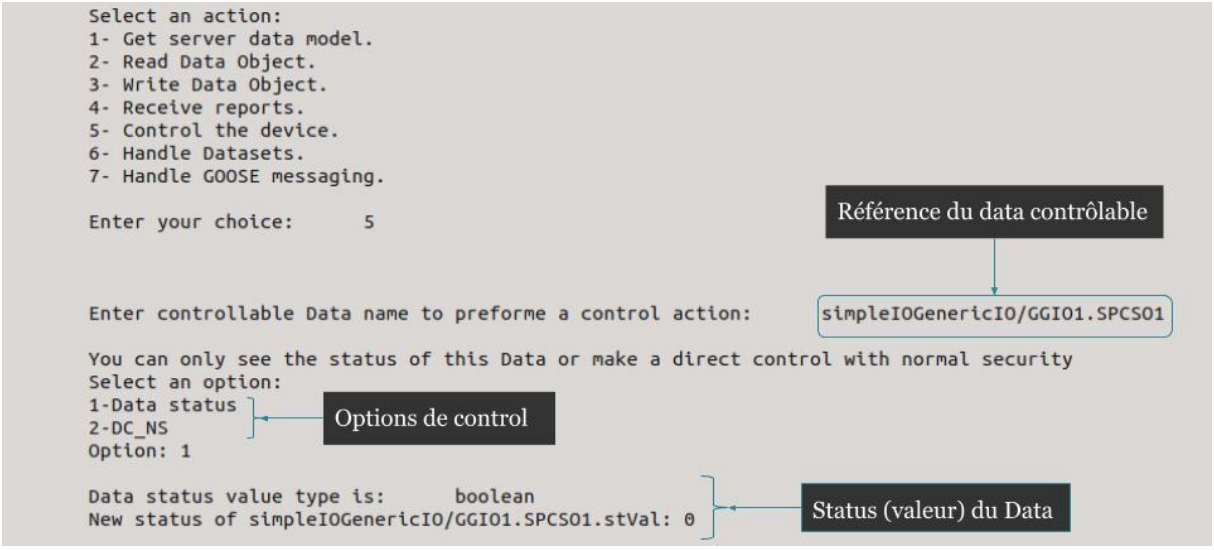

FIGURE 4.7 – Illustration de l'action contrôle -statue du data-

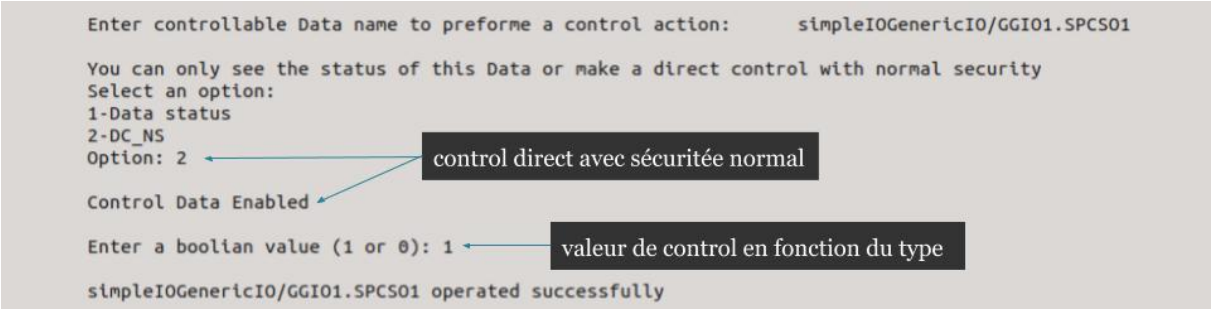

FIGURE 4.8 – Illustration de l'action contrôle -contrôle direct à sécurité normal-

4.2.1.5 Le service : Handel Datasets

Les data set sont utilisés pour simplifier l'accès aux groupes de variables fonctionnels. Ils sont destinés à être utilisés en relation avec les rapports et les messages GOOSE. L'application client développée prend en charge les services suivants liés à l'ensemble de données :

• **Read data set values :** En saisissant la référence du data set, il appelle une fonction de lecture qui affiche les DO / DA qui lui sont associés.

• **Define a new data set :** Autorisez la création d'un nouvel ensemble de données en indiquant son nom et les DO souhaités à y ajouter.

• **Delete an existing data set :** Supprime uniquement les ensembles de données créés par un client. Les ensembles de données prédéfinis à partir des fichiers ICD ne peuvent pas être supprimés.

• **Read the directory of the data set :** Affiche la liste des variables sous un certain ensemble de données.

Les services de data set sont fournis à l'aide des fonctions telles que `IedConnection_readDataSetValues` pour la lecture, `IedConnection_createDataSet` pour la création d'un nouveau data set, `IedConnection_deleteDataSet` pour supprimer un data set effaçable et `IedConnection_getDataSetDirectory` pour obtenir la liste des données associées à un data set. La figure 4.9 montre la lecture d'un data set qui contient 4 DO ayant un FC : « ST ». La lecture comporte le type de chaque DO (dans notre cas c'est des booléen) et leurs valeurs respectives.

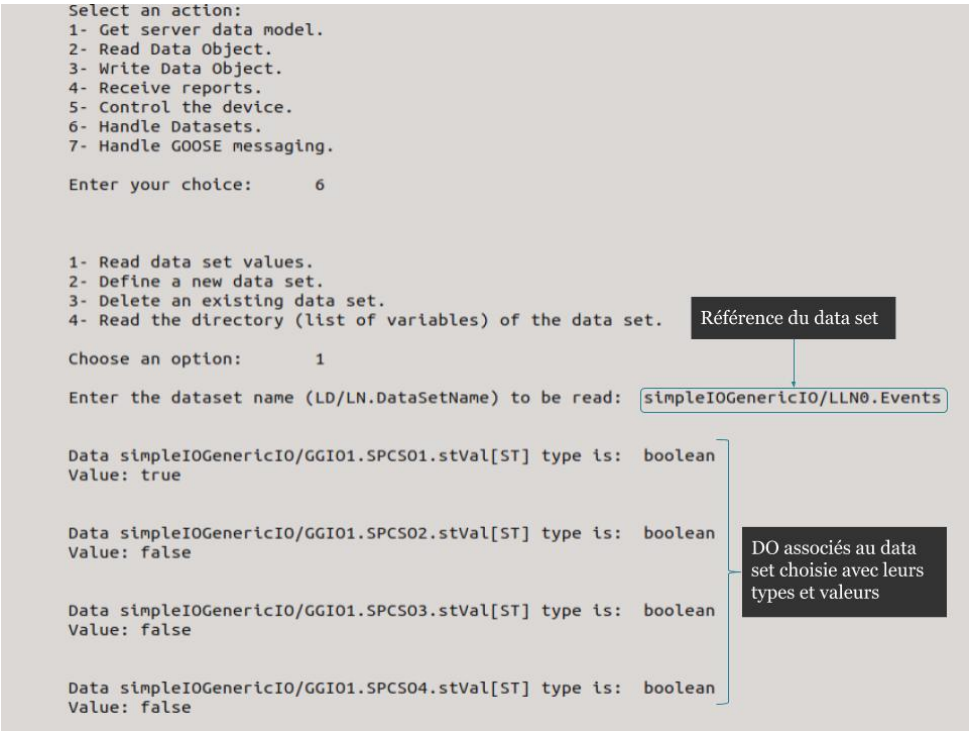

FIGURE 4.9 – Illustration de la lecture d'un data set

4.2.1.6 Le service : Handel GOOSE messages

Les messages GOOSE sont générés automatiquement entre les IED une fois activés. Il existe deux types de configuration GOOSE, le premier provient du fichier ".icd" où toutes les spécifications sont prédéfinies. Le deuxième type provient de l'application cliente, où l'utilisateur crée un nouveau GoCB « effaçable ». Afin de démarrer une communication GOOSE, le client doit l'activer sur le serveur. Le client peut également être informé des valeurs de configuration GoCB existantes ou même en créer une nouvelle.

Différentes fonctions liées à la gestion de GOOSE sont disponibles. La fonction `IedConnection_getGoCBValues` apporte les valeurs de configuration d'un GoCB qui informent l'utilisateur des champs du bloc. Le groupe de fonctions `ClientGooseControlBlock_set` permet de définir des valeurs de configuration (telles que l'activation de messages GOOSE, la définition d'un identifiant GoCB, la définition d'un data set sur un GoCB, etc.).

La figure 4.10 montre les différents champs qui existent dans un GoCB, l'utilisateur peut activer la messagerie GOOSE et modifier l'identifiant APPID et le fichier associé à un bloc de contrôle GOOSE existant. Autrement, tous les autres champs sont corrigés lors de la configuration du fichier SCL.

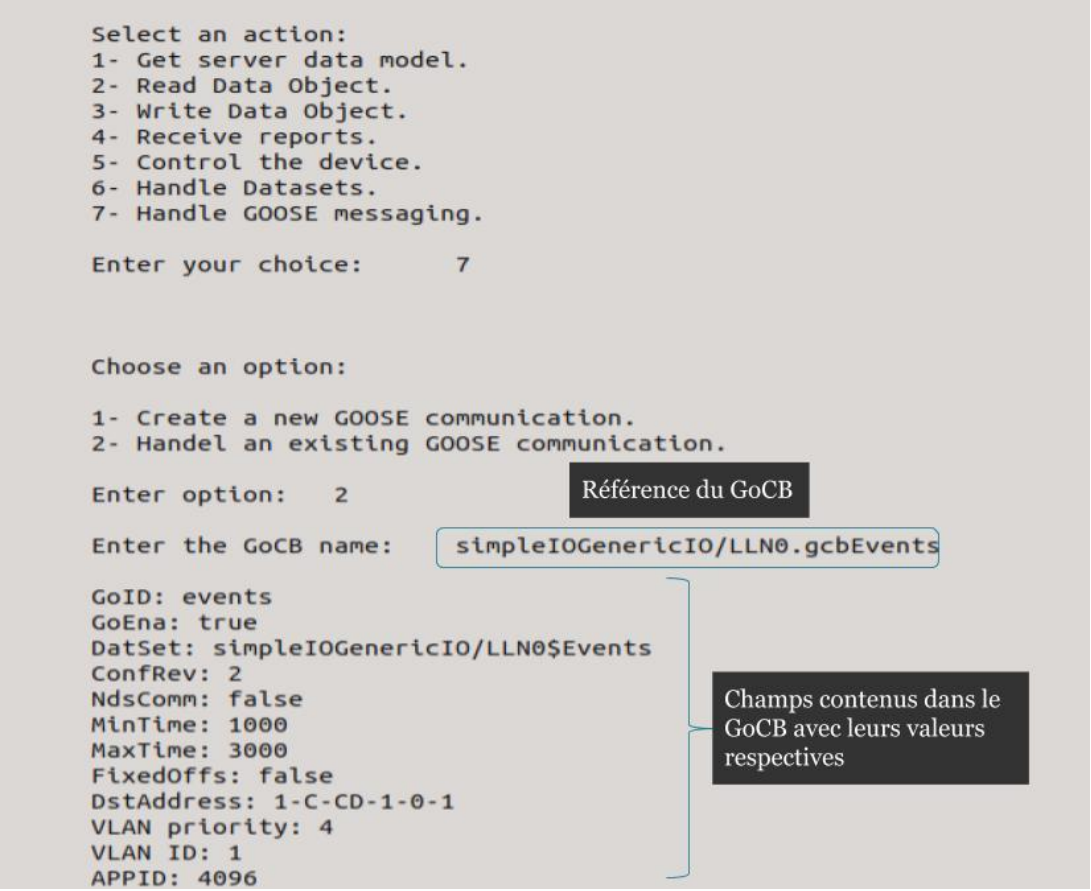

FIGURE 4.10 – Champs d'un GoCB existant dans le serveur

4.2.2 Interface graphique pour le client IEC 61850

Pour avoir une interface graphique conviviale, nous avions l'intention de développer une application Web et cela en utilisant la bibliothèque libIEC61850 mentionnée au chapitre 3. Toutefois, la bibliothèque est écrite en langage C, qui n'est pas destiné pour le développement Web. En guise de solution, nous avons utilisé l'outil de développement logiciel SWIG qui relie les programmes écrits en C et C ++ à divers langages de programmation de haut niveau, notamment Python. Notre application Web utilise les mêmes principes que les fonctions écrites en C.

La figure 4.11 montre la page d'accueil de l'application web ainsi que les différents services équivalents aux celle de l'application CLI.

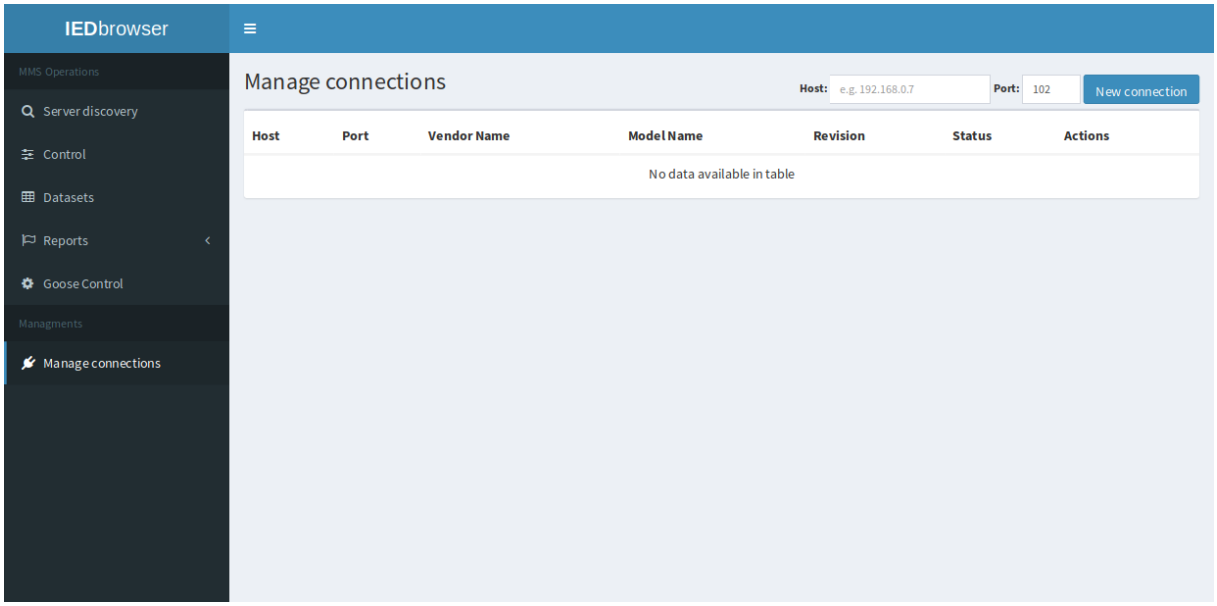

FIGURE 4.11 – Page d'accueil de l'application Web

4.3 L'implémentation de la partie communication de l'IED

Un IED a trois perspectives différentes selon la norme IEC 61850. Il peut être considéré comme un serveur dans le cas d'une communication avec un client, ou un éditeur (respectivement un abonné) dans le cas d'un échange d'informations au niveau de la baie (communications entre IED). Dans ce qui suit, nous allons découvrir ces perspectives et leurs implémentations.

4.3.1 IED en tant que serveur IEC 61850

Afin de fournir des services aux clients, le code développé pour l'IED (en tant que serveur) commence par analyser (en anglais : parse) le fichier de configuration et créer le modèle de données à l'aide de l'application java fournie par la bibliothèque libIEC61850. Le modèle de données créé servira à constituer une instance de serveur qui, d'une part, traitera les demandes des clients et d'autre part, permettra à l'utilisateur de l'IED de définir les différentes configurations utilisées pour gérer le dispositif. Dans ce qui suit, nous présenterons chaque partie des configurations développées appartenant au côté communication du serveur IED implémenté.

4.3.1.1 Option : Start server

Cette option permet au serveur d'écouter les demandes des clients depuis un port pré-sélectionné (généralement le port TCP : 102). Une fois que le serveur est en cours d'exécution (ou également appelé : état d'écoute), aucune configuration ne peut être définie tant que l'utilisateur n'arrête pas le serveur.

La fonction qui gère le processus de démarrage du serveur est `IedServer_start`. Elle prend comme argument une instance `IedServer` créée par rapport au modèle défini par le fichier ".icd" pour fonctionner correctement. La figure 4.12 montre le serveur en état d'écoute.

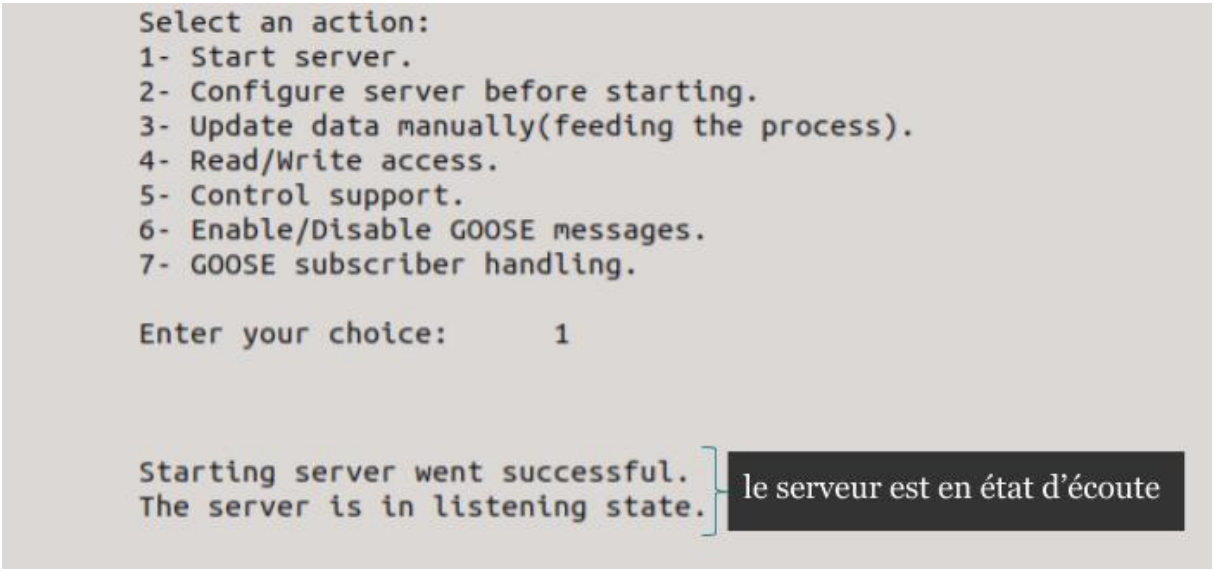

FIGURE 4.12 – Serveur en état d'écoute

4.3.1.2 Option : Configure server before starting

Ce choix permet à l'utilisateur de définir une configuration avant de démarrer le serveur. Cette configuration inclut : la définition de la taille de la mémoire tampon pour les blocs de contrôle des rapports à tampon, la définition du nombre maximal de clients que le serveur peut écouter simultanément et la configuration du data set qui permet d'activer / désactiver le service de data set dynamique pour MMS, ainsi que de de nombreux data sets (non permanents) qu'un client peut créer.

Une configuration est créée à l'aide de l'instance `IedServerConfig` qui dispose des méthodes pour définir les champs de configuration. Une fois la configuration est prête, la fonction `IedServer_createWithConfig` est invoquée en prenant cette configuration comme paramètre, ainsi que l'instance de modèle, et en générant une

instance de serveur correspondant à cette dernière configuration. Ce serveur créé peut ensuite être démarré en sélectionnant l'option de démarrage du serveur.

La figure 4.13 montre une configuration du serveur, ici l'utilisateur a fixé un nombre maximal de connections client à deux connections. Par suite, deux clients seulement peuvent communiquer avec ce serveur une fois démarré.

```
Enter your choice:      2


Configuration options:

1- Set buffer size for buffered report control blocks.
2- Enter the max number of client connections.
3- Dataset configuration.

Enter your choice:      2

Enter the max number of client connections:     2

Keep configurating? 1-Yes 2-No

Enter your choice:      2

The server configuration is done.
```

FIGURE 4.13 – Illustration d'une configuration serveur

4.3.1.3 Option : Update data manually

Ici, l'utilisateur à la possibilité de mettre à jour les valeurs de données. Le choix de cette option demandera une référence de DA pour en agir. Une fois que l'utilisateur a fourni la référence DA, le serveur demande à saisir une nouvelle valeur de données en fonction du type de la DA d'origine (booléen, entier, flottant, chaîne, etc.), puis la valeur est mise à jour dans le modèle de serveur.

Il existe des fonctions permettant de mettre à jour une valeur DA sur le serveur en fonction de son type. Le groupe de fonctions `IedServer_update` se charge de la mise à jour de ces valeurs DA.

Dans la figure 4.14, l'utilisateur est informé à propos de la valeur courante de la DA, et il a la possibilité de la changée en entrant une nouvelle valeur ayant le type correspondant.

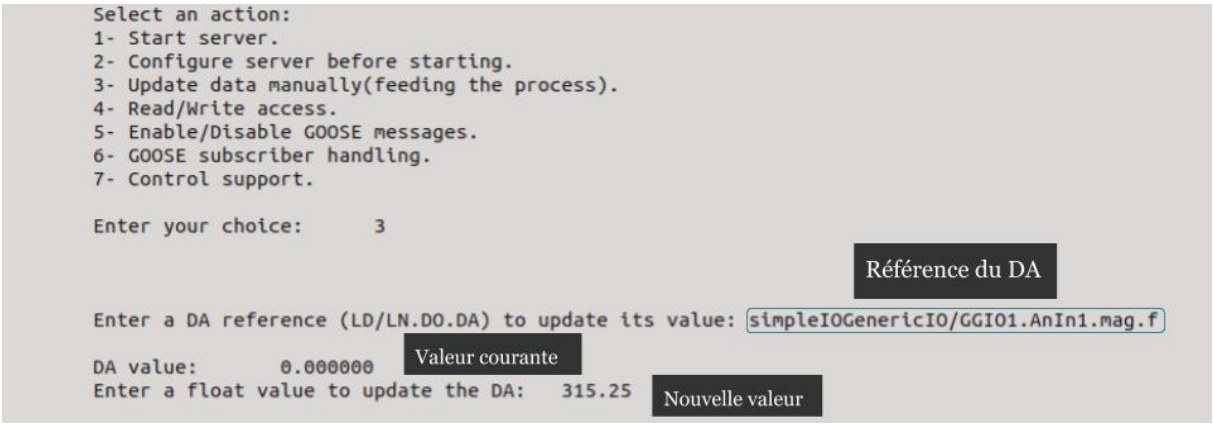

FIGURE 4.14 – Illustration d'une mise à jour de DA au niveau de serveur

4.3.1.4 Option : Write access

Cette option définira l'accessibilité des données en cas de mises à jour côté client (accessibilité en écriture). L'utilisateur doit choisir un type de FC pour lequel la stratégie d'accès sera définie (autorisée ou refusée). Une fois le choix effectué et tant que le serveur est en marche, aucun client ne peut modifier les valeurs de données dont le type de FC est déjà sélectionné. Ceci est très utile dans certains cas où le FC est "CO" (contrôle) pour éviter les accidents.

Pour modifier la stratégie d'accès en écriture, la fonction `IedServer_setWriteAccessPolicy` est appelée, elle prend comme argument le type FC et une valeur macro `ALLOW` pour autoriser l'accès ou `DENY` pour le refuser.

La figure 4.15 montre une liste des types FC à sélectionner. Une fois la sélection est terminée, l'utilisateur doit définir la stratégie d'accès pour les données ayant le FC choisie.

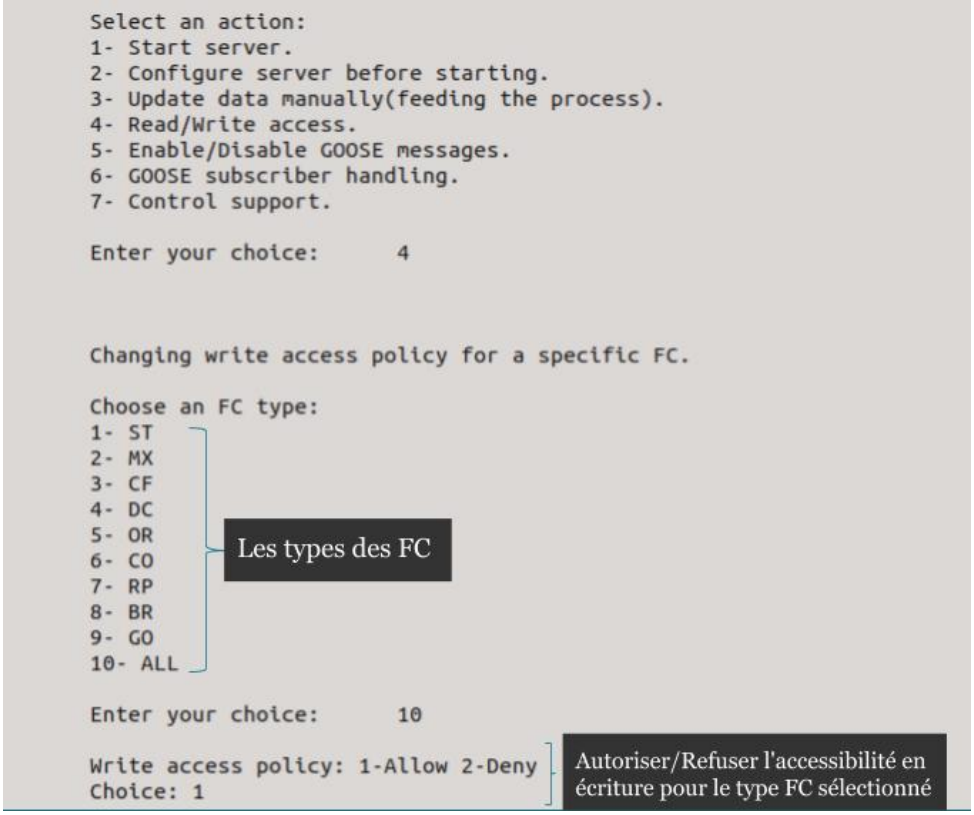

FIGURE 4.15 – Illustration de l'accessibilité en écriture

4.3.1.5 Option : Control support

Cette option est utilisée à des fins de contrôle (côté serveur), son objectif principal est d'installer un gestionnaire qui sera appelé chaque fois qu'un client effectue une action de contrôle. Il mettra à jour les données relatives au service de contrôle. Même s'il semble être similaire à l'option « Update data manually » décrite précédemment, les fonctions de contrôle font l'objet d'une gestion spécifique en raison de sa lourdeur.

L'option de contrôle utilise les mêmes fonctions de mise à jour que dans l'option « Update data manually ». Cependant, ils doivent être appelés dans un gestionnaire pour fonctionner correctement.

Dans la figure 4.16, une fois l'utilisateur entre la référence du DO contrôlable, le gestionnaire de contrôle est installé et les valeurs du DO choisie sont en mise à jour après chaque action de contrôle effectuée par un client.

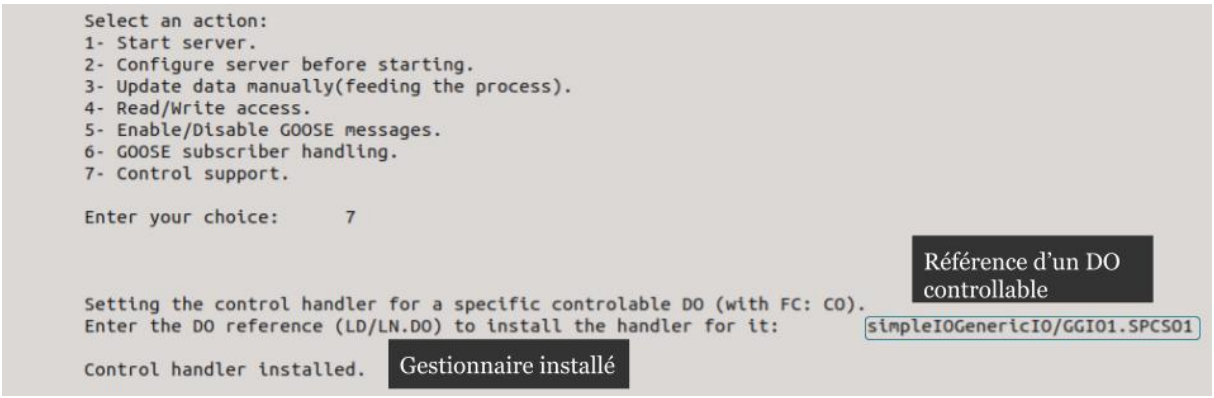

FIGURE 4.16 – Installation du gestionnaire de control pour un DO contrôlable

4.3.1.6 Option : Stop server

Cette option permet à l'utilisateur d'arrêter le serveur afin de pouvoir définir de nouvelles configurations. Elle est disponible uniquement lorsque le serveur est en cours d'exécution, sinon l'option est masquée.

Les fonctions qui gèrent le processus d'arrêt du serveur sont `IedServer_stop`. Elle prend comme argument une instance `IedServer` créée qui correspond au serveur en marche. La figure 4.17 montre le serveur en état de repos.

FIGURE 4.17 – Serveur en état d'arrêt

4.3.2 IED en tant qu'éditeur IEC 61850

Les communications au niveau de la baie sont basées sur le protocole GOOSE. Les IED peuvent démarrer / arrêter la publication des messages GOOSE pré-configurés en appelant la fonction d'activation / dés-activation correspondante appliquée sur l'instance du serveur. De plus, un nouvel éditeur GOOSE peut être créé par l'utilisateur, cet éditeur est associé à un GoCB. Il doit être configuré avec les champs correspondants (ID, paramètres de communication, data set associé, etc.)

L'activation / dés-activation de la messagerie GOOSE est défini à l'aide de la fonction `IedServer_enablegoose` avec le paramètre correspondant (`true` pour l'activation et `false` pour la dés-activation).

La création d'un nouvel éditeur est faite en établissant l'instance `GoosePublisher` et en initialisant ses champs (ID, APPID, data set, etc.) à l'aide des fonctions de définition (Setter Functions) `GoosePublisher_set`.

Dans la figure 4.18, l'utilisateur a choisie l'interface de communication qui va être utilisée lors de la publication des messages GOOSE, ensuite il a activé la messagerie GOOSE.

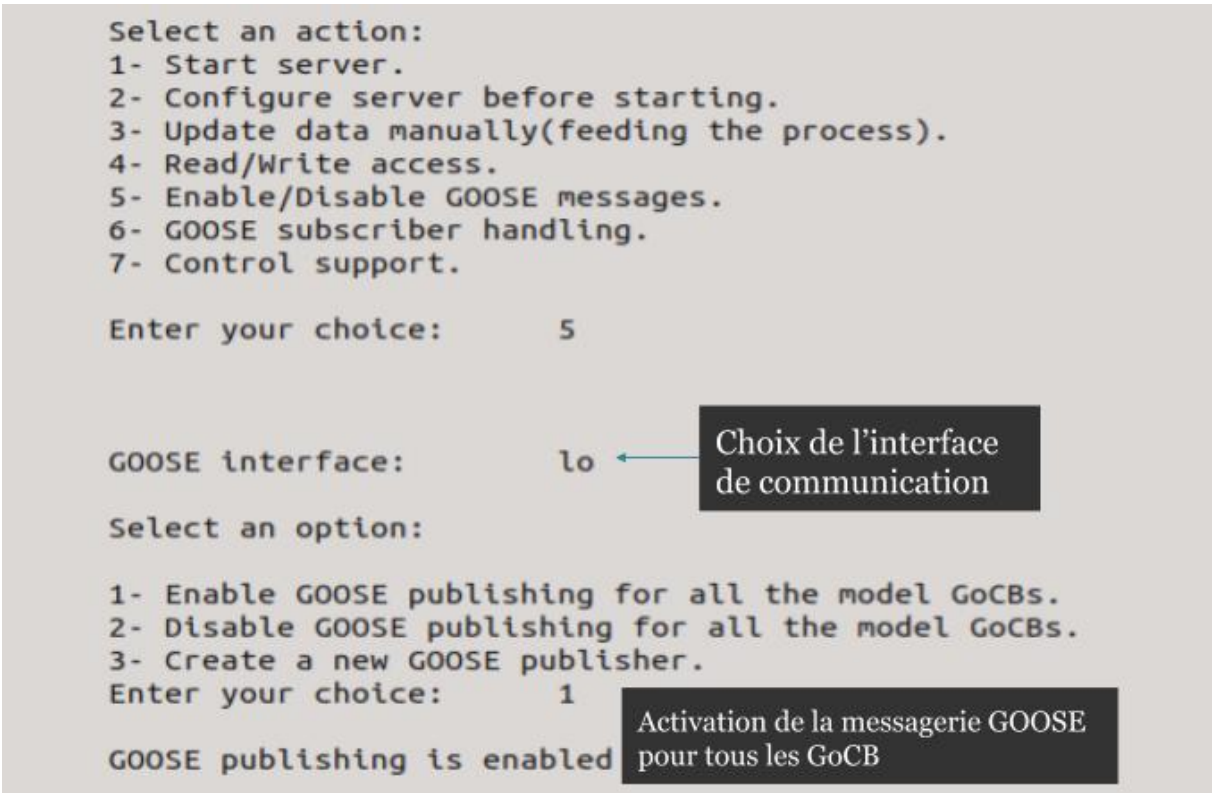

FIGURE 4.18 – Activation de la messagerie GOOSE au niveau du serveur

4.3.3 IED en tant qu'abonné IEC 61850

Utiliser le IED en tant qu'abonné GOOSE nécessite en premier lieu de créer une instance de récepteur GOOSE et de définir l'ID de l'interface (interface de communication Ethernet) à partir duquel il sera à l'écoute. Une instance d'abonné est également créée., son travail consiste à stocker les données reçues par le destinataire afin d'installer un gestionnaire qui les traite (les affiche ou prend une décision en fonction des valeurs reçues).

Avant de démarrer l'instance du récepteur à l'état d'écoute, l'instance d'abonné devrait y être ajoutée. L'abonné a un gestionnaire installé, qui sera appelé une fois le message GOOSE reçu. De plus, le "APPID" pour l'abonné doit être défini de manière identique à celui existant dans le GoCB afin de relier l'abonné à cet éditeur spécifique, sinon les messages GOOSE ne seront pas reçus.

La fonction `GooseSubscriber_setListener` est responsable d'installer le gestionnaire de traitement qui dans notre cas il affiche la séquence des messages GOOSE reçue. La figure 4.19 montre les messages GOOSE reçues.

```
GOOSE event:
  {stNum: 2 sqNum: 0}
   timeToLive: 1500
   timestamp: 1560614798.155
{{{50.750000},0000000000000,20190615160638.155Z},{{0.000000},0000000000
000,19700101000000.000Z},{{0.000000},0000000000000,19700101000000.000Z}
,{{0.000000},0000000000000,19700101000000.000Z}}
GOOSE event:
  {stNum: 3 sqNum: 0}
   timeToLive: 1500
   timestamp: 1560614798.255
{{{101.500000},0000000000000,20190615160638.255Z},{{0.000000},000000000
0000,19700101000000.000Z},{{0.000000},0000000000000,19700101000000.000Z
},{{0.000000},0000000000000,19700101000000.000Z}}
GOOSE event:
  {stNum: 4 sqNum: 0}
   timeToLive: 1500
   timestamp: 1560614798.355
{{{152.250000},0000000000000,20190615160638.355Z},{{0.000000},000000000
0000,19700101000000.000Z},{{0.000000},0000000000000,19700101000000.000Z
},{{0.000000},0000000000000,19700101000000.000Z}}
```

FIGURE 4.19 – Capture de la réponse du gestionnaire au niveau du l'abonné

4.4 Conclusion

Dans ce chapitre, nous avons présenté la mise en œuvre du nœud de communication IEC 61850, en présentant les procédés de développement suivi ainsi que les différentes approches de l'implémentation.

Dans le chapitre suivant, nous allons exposer les résultats des tests fait en utilisant le nœud implémenté et présenté notre application web pour un client IEC 61850.

Chapitre 5

Test et résultats

5.1 Introduction

Dans ce dernier chapitre, nous allons présenter les résultats du test de l'application Web client IEC 61850 avec différents serveurs (principalement deux) pour la communication MMS. Le premier est celui implémenté dans le BBB et le second est l'IED de la société SEL. De plus, nous allons tester la communication GOOSE en créant un scénario émulé pour vérifier les performances. Les résultats des tests seront interprétés conformément aux spécifications de la norme IEC 61850.

5.2 Tests de l'application Web client IEC 61850 avec différents serveurs

Pour que l'application cliente se connecte à un serveur, ce dernier doit être à l'écoute. Lors des tests que nous avons effectués, les deux serveurs SEL IED et le serveur implémenté BBB ont été activés. Dans la figure 5.1, l'application cliente ayant été lancée, l'utilisateur doit entrer l'adresse IP des serveurs avec lesquels il souhaite interagir (dans ce cas, il ne s'agit que de trois serveurs à la fois), ainsi que le port de communication. Une fois la communication établie, l'utilisateur peut commencer à demander des services aux différents serveurs. Dans ce qui suit, nous présenterons les résultats des différentes actions du client avec les serveurs.

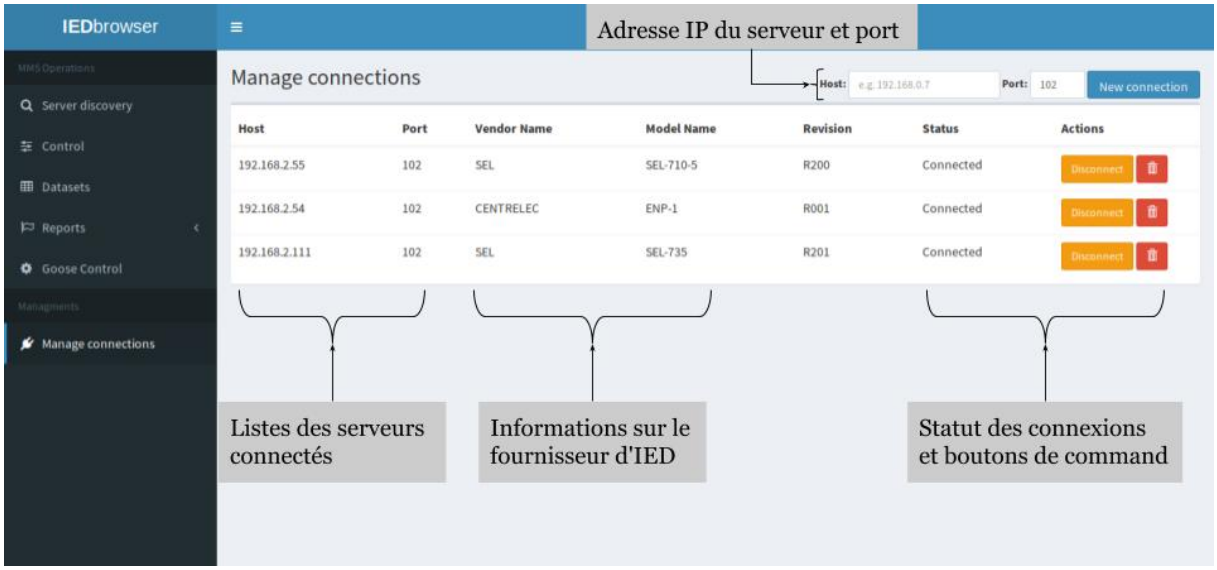

FIGURE 5.1 – La gestion des connexions aux serveurs

Comme le montre la figure 5.1 , l'application cliente peut se connecter aux divers serveurs de différents fournisseurs (deux IED de la société SEL et le serveur développé pour CENTRELEC) ce qui approuve l'interopérabilité avec différents périphériques.

Dans la figure 5.2, le client a demandé le modèle de données pour l'IED de la société SEL. Nous pouvons voir les différentes données présentes sur le serveur (LD, LN, DO, DA, RCB, etc.) avec leurs valeurs respectives. Ici, le client n'a pas obtenu uniquement le modèle de données mais peut effectuer d'autres opérations de lecture et d'écriture.

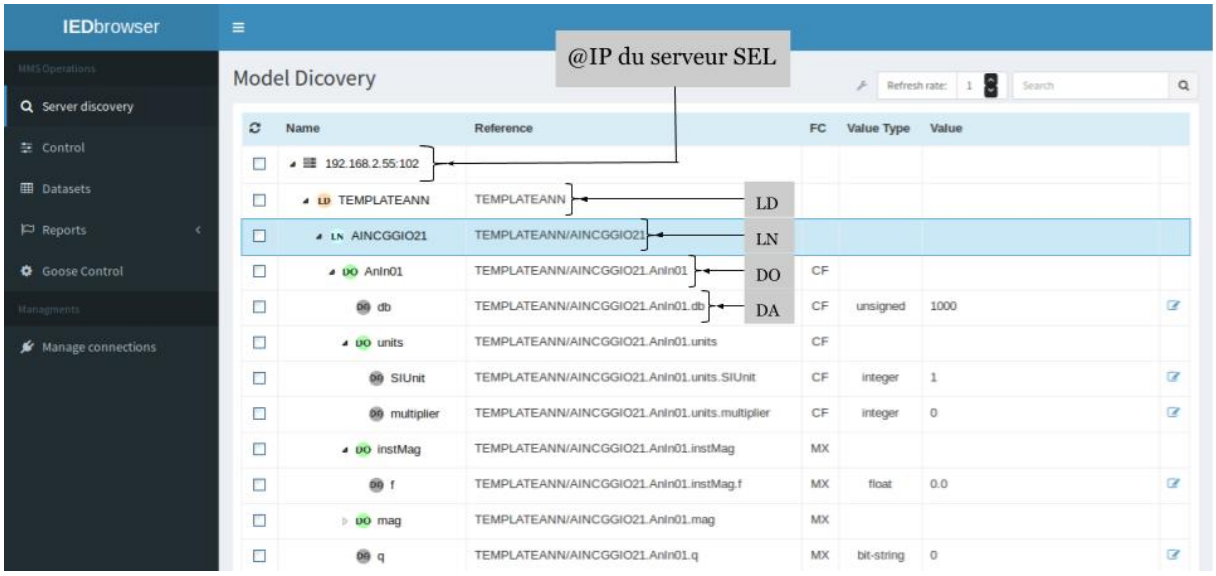

FIGURE 5.2 – Modèle de données du serveur SEL

La figure 5.2 ne montre qu'une partie du modèle. L'utilisateur est informé du type de FC et des valeurs des données (à lire lorsque coché dans les cases). Le processus d'écriture peut être lancé en cliquant sur la valeur souhaitée et en entrer une nouvelle. L'utilisateur doit introduire une valeur avec le type approprié, sinon un message d'erreur s'affichera et les données ne seront pas mises à jour. Les données qui ne sont pas accessibles en écriture affichent un message d'erreur `IED_ACCESS_DENIED`.

La figure 5.3 montre les configurations des rapports. Il présente les champs de bloc de contrôle de rapport, nous définissons ici les rapports comme activés avec une option de déclencheur : intégrité. Dans la figure 5.4, nous pouvons confirmer que les rapports sont reçus toutes les 3 secondes (la période d'intégrité), contenant les valeurs des données collectées dans l'ensemble de données associé au rapport configuré.

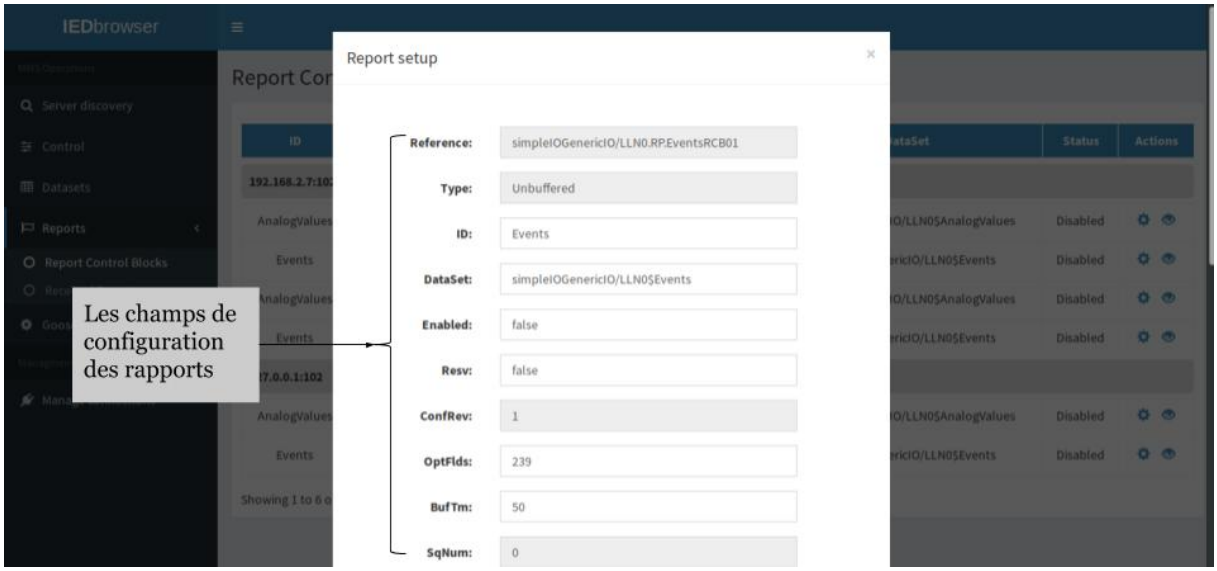

FIGURE 5.3 – Champs de configuration du bloc de contrôle de rapport

FIGURE 5.4 – Réception des rapports périodiques

Dans la figure 5.5, nous présentons les data set. L'utilisateur peut lire des data set existants ou en créer de nouveaux. Les nouvelles data set créé sont ajoutés à la liste et comportent un bouton de suppression en contraste avec ceux créés à partir du fichier ".icd".

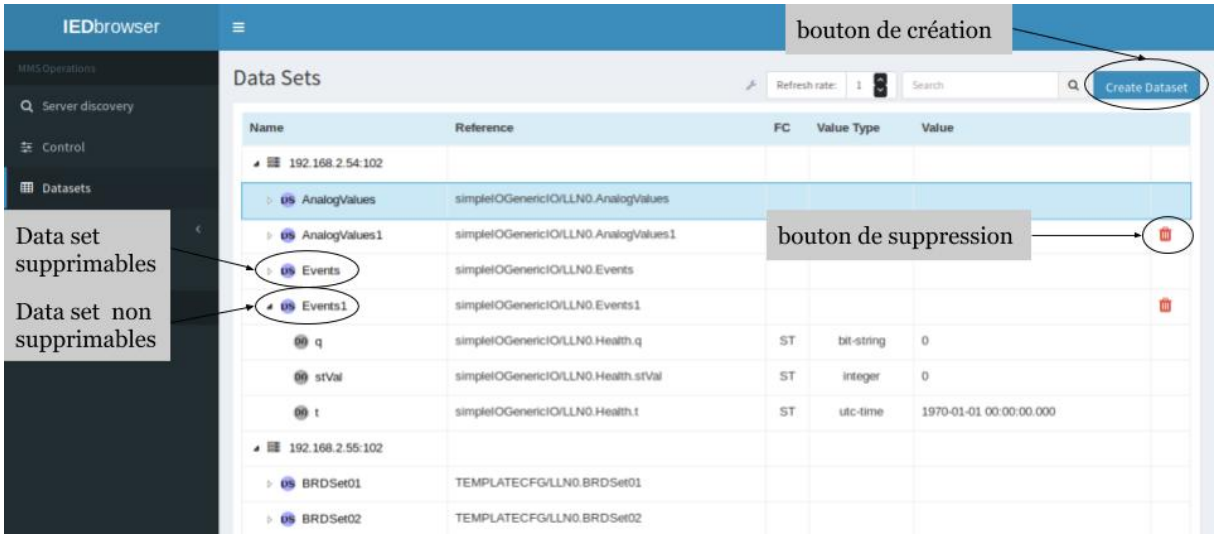

FIGURE 5.5 – La gestion des data sets

La figure 5.6 montre une capture des paquets transférés avec Wireshark lors de la réception des rapports. Nous pouvons reconnaître les adresses IP du client (IP = 192.168.2.10) et du serveur BBB (IP = 192.168.2.7), ainsi que différents protocoles utilisés pour la communication (principalement les protocoles TCP et MMS).

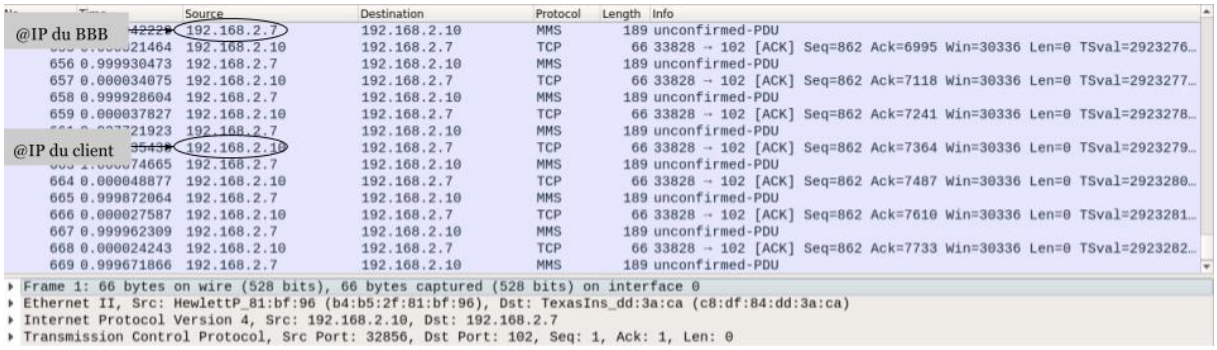

FIGURE 5.6 – Trames capturés avec Wireshark lors de la communication MMS

5.3 Test de messagerie GOOSE

Afin de tester la communication GOOSE dans un cas d'utilisation, nous avons créé un scénario dans lequel l'éditeur GOOSE (démarré dans l'ordinateur) publie des messages où le DA `simpleIOGenericIO/GGIO.AnIn1.mag.f` (valeur de type « Float » supposée contenir une mesure de tension) change dans le temps (de 0 à 569,8836). De l'autre côté, l'abonné GOOSE (travaillant dans le BBB) est configuré pour recevoir les messages de l'éditeur GOOSE. Le gestionnaire utilisé pour cet

abonné a été développé pour écrire les informations GOOSE dans un fichier nommé `GOOSE_Messages.txt` ainsi que pour effectuer une action de contrôle interne (action d'ouverture : valeur = 0) sur le DO contrôlable `simpleIOGenericIO/GGIO.SPCSO1` qui est initialement dans un état fermé (valeur = 1) lorsque la valeur du DA mentionnée ci-dessus dépasse 450.

Dans la figure 5.7, l'utilisateur active la messagerie GOOSE pour l'éditeur souhaité GoCB `simpleIOGenericIO/LLN0.gcbAnalogValues`. Le DO contrôlable `simpleIOGenericIO/GGIO.SPCSO1` est à l'état fermé, comme dans la figure 5.8.

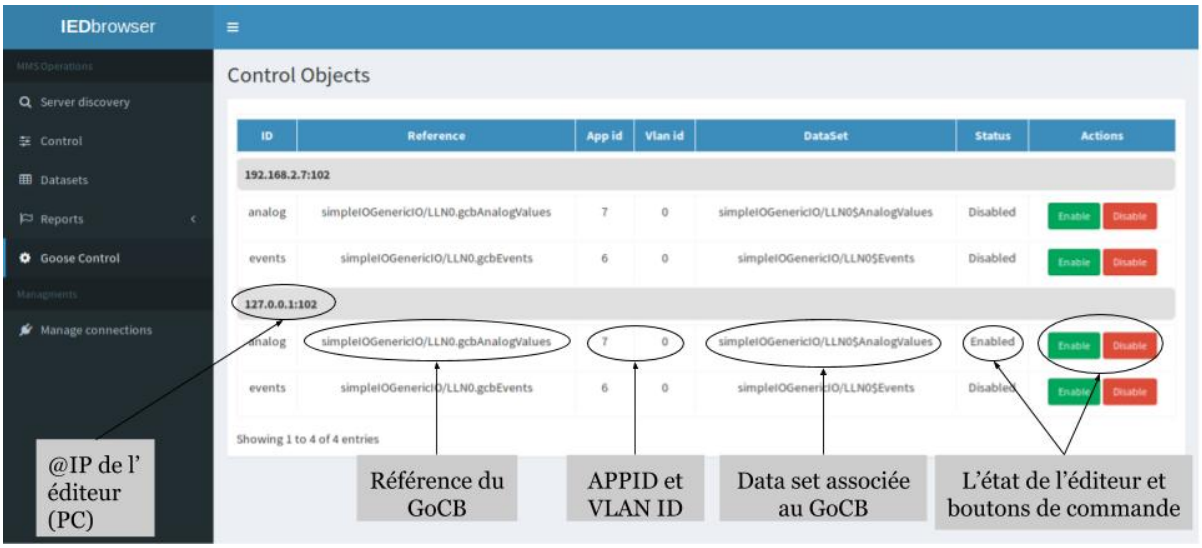

FIGURE 5.7 – Gestion des blocks de control GOOSE

FIGURE 5.8 – Gestion des objets de contrôle

Sur le serveur BBB (figure 5.9), l'abonné est configuré avec la référence GoCB correspondante, APPID et le gestionnaire installé.

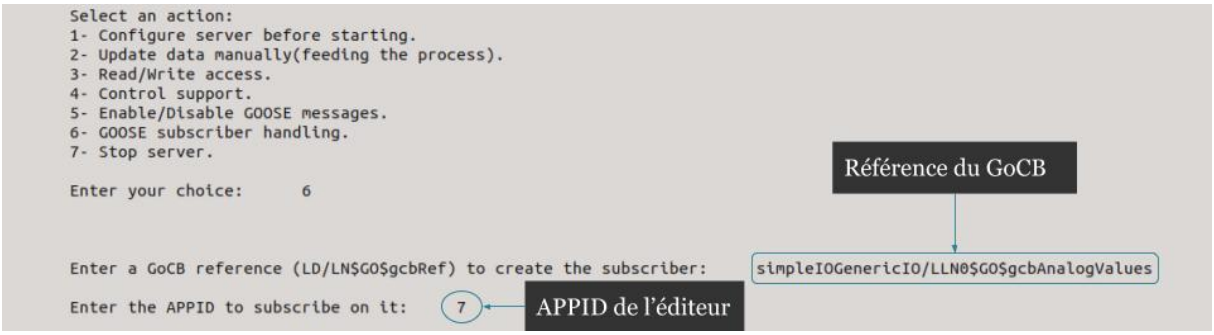

FIGURE 5.9 – Configuration de l'abonné

Après un certain temps du lancement du processus, le fichier `GOOSE_Messages.txt` contient les informations indiquées dans la figure 5.10. Le changement de données indiqué dans la figure 5.10 représente le DA `simpleIOGenericIO/GGIO.AnIn1.mag.f`. Nous pouvons observer que le numéro d'état augmente tandis que le numéro de séquence est égal à zéro, ce qui est conforme à la réception de données mises à jour.

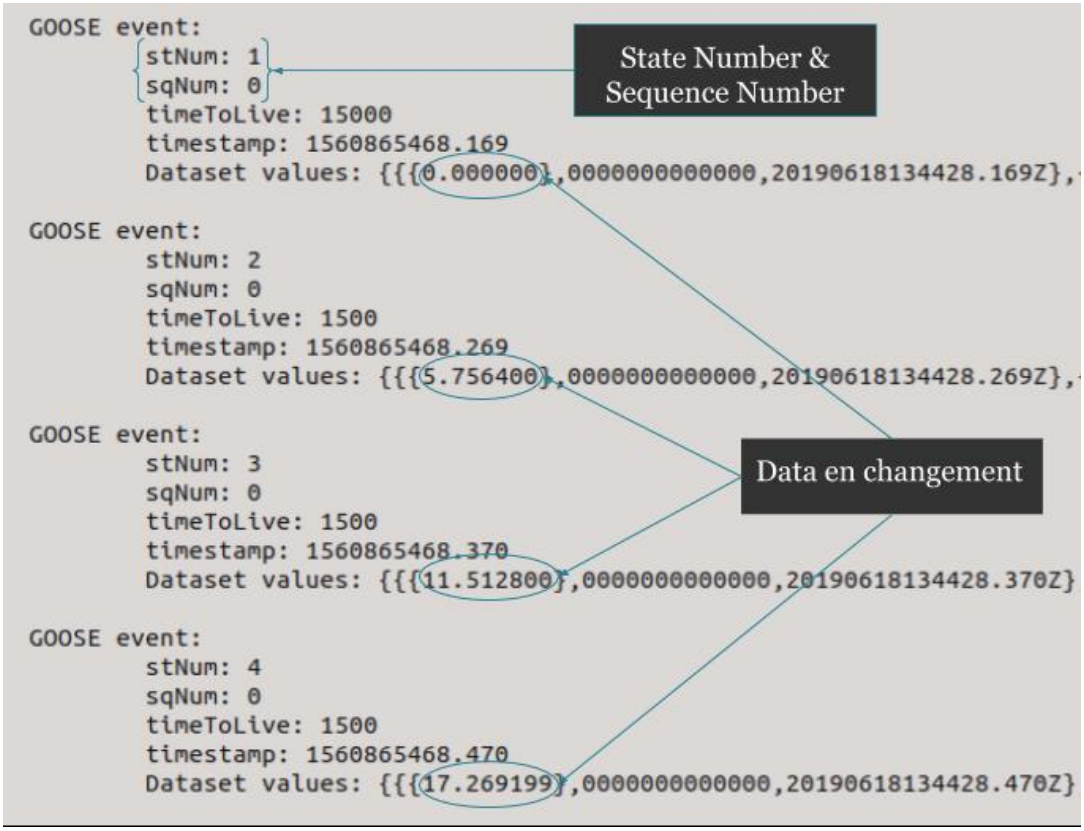

FIGURE 5.10 – Une capture des messages GOOSE reçus

Dans la figure 5.11, nous vérifions qu'effectivement la DO contrôlable a été mise à l'état bas (valeur = `false`) en consultant le champ de contrôle.

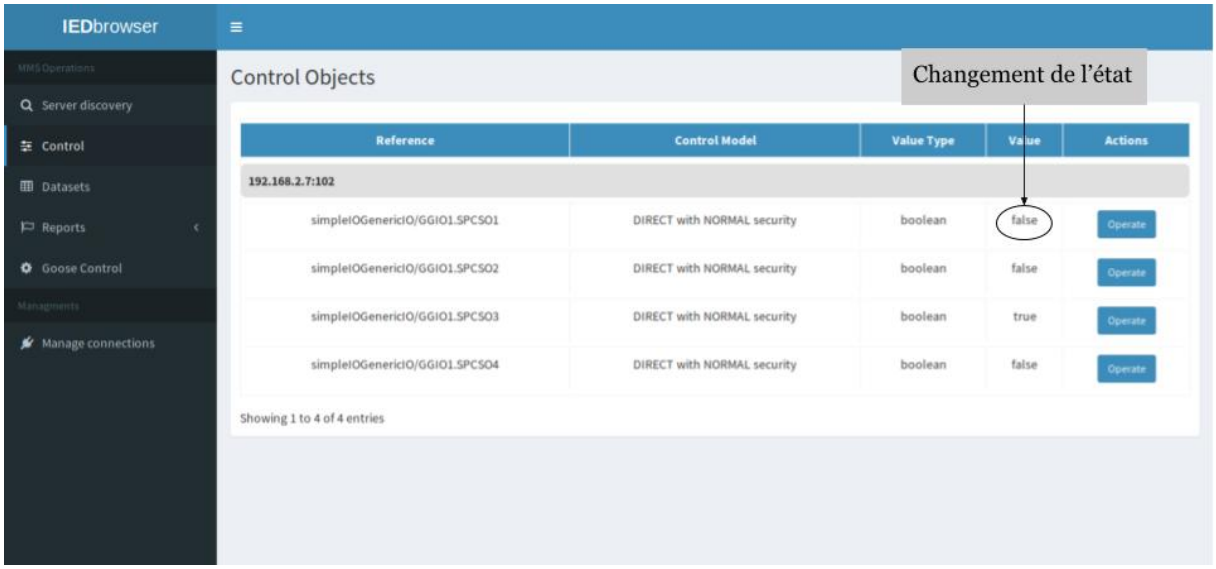

FIGURE 5.11 – Changement d'état de la DO contrôlable

La figure 5.12 montre les paquets capturés par Wireshark lors d'une communication GOOSE. L'adresse source est l'adresse MAC de l'éditeur (PC HP) et l'adresse de destination est l'adresse MAC de l'abonné (BBB : abonné IEC 61850 GOOSE). Nous pouvons distinguer la partie où les données étaient en cours de modification (indiquée par le symbole 1 sur la figure 5.12). Ici, l'intervalle de temps entre deux messages GOOSE était de l'ordre des millisecondes, tandis que lorsque les données atteignent une valeur constante, l'intervalle peut atteindre plus qu'une seconde (indiqué par le symbole 2 sur la figure 5.12). En conséquence, le mécanisme de transmission GOOSE mentionné dans le chapitre 2 est validé.

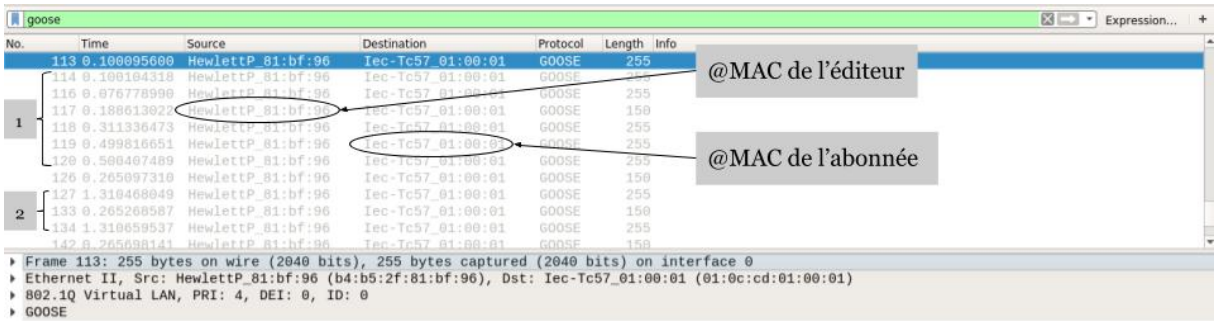

FIGURE 5.12 – Trames capturés avec Wireshark lors de la communication

5.4 Conclusion

Dans ce chapitre, nous avons présenté les résultats des tests menés sur le nœud de communication IEC 61850 implémenté à l'aide de l'application Web en illustrant certaines de ces fonctionnalités possibles. Nous croyons que cette application Web peut être améliorée à l'avenir et que de nombreux autres projets peuvent en être tirés. Nous avons terminé le chapitre en présentant un scénario dans lequel la communication GOOSE était impliquée.

Conclusion générale

Comme indiqué dans l'introduction, la communication joue un rôle essentiel dans le développement industriel et les opérations en temps réel du système électrique. Ce travail contribue à la mise en place d'un nœud de communication conforme aux protocoles de niveau industriel IEC 61850 en utilisant des solutions Open Source.

Dans les premiers chapitres de ce travail, nous nous sommes concentrés sur les concepts de base de la norme IEC 61850 : modèle de données, configuration et protocoles de communication en particulier.

Après, nous avons défendu nos choix techniques pour la conception et la mise en œuvre de ce projet. Un matériel dédié (carte BeagleBone) et un logiciel Open Source (libiec61850) ont été utilisés.

Ensuite, nous avons présenté les étapes suivies pour la mise en œuvre de chaque côté de communication (client IEC 61850 et partie communication d'un IED comme étant serveur et éditeur / abonné).

Enfin, nous avons validé le bon fonctionnement des application implémentées en appliquant divers tests et en exposant leurs résultats.

Dans le cadre de la poursuite de ce projet, les recommandations et les travaux futurs pourraient porter sur :

- Ajout d'un graphique de processus dynamique au client implémenté
- Implémentation et test de la partie communication SV (à ajouter à l'IED)
- Joindre la partie de communication liée à l'IED à un circuit d'alimentation électronique dédié prenant en charge les différents rôles qu'un IED complet peut offrir (partie électrotechnique)
- Ajout d'une HMI sophistiquée à l'IED (pour une meilleure interaction).

Bibliographie

[1] Y. Liang and R. H. Campbell, "Understanding and simulating the iec 61850 standard [en ligne]. [consulté le 23 mars 2019]. disponible sur : <https://www.researchgate.net/publication/32964907_Understanding_and_Simulating_the_IEC_61850_Standard>."

[2] T. Arun, L. Lathesh, and A. Suhas, "Substation automation system [en ligne]," *International Journal of Scientific & Engineering Research*, vol. 7, no. 5, pp. 215–218, 2016. [consulté le 7 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/319503832_Substation_Automation_system>.

[3] I. Automation, "Inductive automation [en ligne]. [consulté le 7 avril 2019]. disponible sur : <https://inductiveautomation.com/resources/article/what-is-hmi>."

[4] R. Gore, H. Satheesh, M. Varier, and S. Valsan, "Analysis of an iec 61850 based electric substation communication architecture. [en ligne]," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 388–393, IEEE. [consulté le 10 avril 2019]. Disponible sur : <https://www.semanticscholar.org/paper/Analysis-of-an-IEC-61850-based-Electric-Substation-Gore-Satheesh/2778e498d53b07729331e97ac4a95194e2bb105b>, 2016.

[5] M. Salman, J. Mousavi Mirrasoul, S. James, *et al.*, "Modeling distribution automation system components using iec 61850 [en ligne]," in *2009 IEEE Power and Energy Society General Meeting, PES09*, 2009. [consulté le 10 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/224598619_Modeling_Distribution_Automation_System_Components_Using_IEC_61850>.

[6] D. Baigent, M. Adamiak, R. Mackiewicz, and G. M. G. M. SISCO, "Iec 61850 communication networks and systems in substations : An overview for users [en ligne]," *SISCO Systems*, 2004. [consulté le 10 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/292238350_IEC_61850_communication_networks_and_systems_in_substations_An_overview_for_users>.

[7] D. Nowak, "*Bridge between industrial wireless sensor network and wired Ethernet network* [en ligne]. thèse : électronique. agh université des sciences et de la technologie de cracovie pologne, 2012. [consulté le 13 mars 2019]. [consulté le 23 mars 2019]. disponible sur : <http://www.smartgrid.agh.edu.pl/pdf/Master_Thesis_DominikNowak.pdf>."

[8] J. Park, E. In, S. Ahn, C. Jang, and J. Chong, "Iec 61850 standard based mms communication stack design using oop [en ligne]," in *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pp. 329–332, IEEE., 2012. [consulté le 14 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/254028540_IEC_61850_standard_based_MMS_communication_stack_design_using_OOP>.

[9] P. Matoušek, "Description of iec 61850 communication [en ligne]," 2018. [consulté le 28 avril 2019]. Disponible sur : <http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11832>.

[10] P. de Araújo, J. Rodrigues, J. Oliveira, S. Braga, *et al.*, "Infrastructure for integration of legacy electrical equipment into a smart-grid using wireless sensor networks [en ligne]," *Sensors*, vol. 18, no. 5, p. 1312, 2018. [consulté le 28 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/324752017_Infrastructure_for_Integration_of_Legacy_Electrical_Equipment_into_a_Smart-Grid_Using_Wireless_Sensor_Networks>.

[11] M. El Hariri, T. Youssef, and O. Mohammed, "On the implementation of the iec 61850 standard : Will different manufacturer devices behave similarly under identical conditions? [en ligne]," *Electronics*, vol. 5, no. 4, p. 85, 2016. [consulté le 10 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/311448099_On_the_Implementation_of_the_IEC_61850_Standard_Will_Different_Manufacturer_Devices_Behave_Similarly_under_Identical_Conditions>.

[12] F. Xia, Z. Xia, and X. Huang, "Summary of goose substation communication [en ligne]," in *MATEC Web of Conferences*, vol. 25, p. 01007, EDP Sciences, 2015. [consulté le 28 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/282771945_Summary_of_GOOSE_Substation_Communication>.

[13] K. K. M. S. Kariyawasam, "Implementation of an iec 61850 sampled values based line protection ied with a new transients-based hybrid protection algorithm [en ligne]. thèse : Génie électrique et informatique. université du manitoba winnipeg, mb, canada, 2016," 2016. [consulté le 28 avril 2019]. Disponible sur : <http://hdl.handle.net/1993/31306>.

[14] E. Solomin, D. Topolsky, and N. Topolsky, "Arrangement of data exchange between adaptive digital current and voltage transformer and scada-system under iec 61850 standard [en ligne]," *Procedia engineering*, vol. 129, pp. 207–212, 2015. [consulté le 28 avril 2019]. Disponible sur : <https://www.researchgate.net/publication/287157760_Arrangement_of_Data_Exchange_between_Adaptive_Digital_Current_and_Voltage_Transformer_and_SCADA-system_under_IEC_61850_Standard>.

[15] BeagleBone, "beagleboard.org [en ligne]. [consulté le 13 mai 2019]. disponible sur : <https://beagleboard.org/black>."

[16] T. Instruments, "Texas instruments [en ligne].[consulté le 13 mai 2019]. disponible sur : <https://www.ti.com/product/AM3358>."

[17] M. Zillgith, "libiec61850 / lib60870-5 [en ligne]. [consulté le 9 mai 2019]. disponible sur : <https://libiec61850.com/libiec61850/about>."

[18] M. Zillgith, "libiec61850 / lib60870-5 [en ligne]. [consulté le 9 mai 2019]. disponible sur : <https://libiec61850.com/libiec61850/configuration-file-format/>."

Annexes

.1 Annexe A : ASN.1 and BER Encoding

La notation de syntaxe abstraite 1 (ASN.1, définie par l'UIT-T X.680) spécifie les catégories suivantes de types de données :

1. Types de données primitifs :

- BOOLEAN
- INTEGER
- BIT STRINT
- OCTET STRING
- NULL
- IDENTIFIANT D'OBJET
- ObjectDescriptor
- EXTERNAL
- REAL
- ENUMERATED
- UTF8String
- NumericString
- PrintableString
- IA5String
- UTCTime
- GeneralizedTime
- GraphicString
- VisibleString
- GeneralString
- UniversalString
- CHARACTER STRING

2. Types de données à l'échelle de l'application : Non normalisé mais défini par chaque application (MMS, GOOSE and SV).

3. Types de données constructeur :

- SEQUENCE, SEQUENCE OF - étiquette universelle de classe 16
- SET, SET OF - étiquette de classe universelle 17
- CHOICE
- SELECTION
- ANY

Les règles de codage de base (BER, norme ITU-T X.690) définissent la syntaxe de transfert des structures de données ASN.1 transmises entre applications. BER décrit une méthode permettant de coder les valeurs de données ANS.1 sous la forme d'une chaîne d'octets. Il code pour une valeur ASN.1 comme un (-longueur-valeur type) triplet TLV qui comprend un identificateur du type de données, la longueur de la valeur, et la valeur elle-même.

• Type (ou identifiant) : est une valeur sur un octet qui indique le type ASN.1, voir ci-dessous.

• Length : indique la longueur de la représentation de la valeur réelle.

• Value : représente la valeur du type ASN.1 sous la forme d'une chaîne d'octets. Pour les types construits, la valeur peut être un triplet de TLV incorporé.

L'identificateur spécifie le type de données ASN.1, la classe du type et la méthode de codage comme suit :

1. Les deux premiers bits déterminent la classe du type de données ASN.1 :

• Universel (00) - les types de données universels (primitifs ou constructifs) sont donnés par la norme

• Application (01)

• Spécifique au contexte (10)

• Privé (11)

2. Le troisième bit décrit la méthode de codage :

Forme primitive (0) ou construite (1). S'il est défini sur 1, les types de données construits tels que SEQUENCE, SET, CHOICE, etc. sont utilisés. Cela signifie également qu'un autre triplet de TLV est incorporé en tant que valeur.

3. Les cinq derniers bits identifient le type de données :

Ceci s'appelle une étiquette. Les balises de type de données universelles sont énumérés ci-dessus. Les balises d'application, spécifiques au contexte et privées sont définies par l'application.

La longueur peut être encodée sous trois formes :

• Longueur définie courte (forme primitive) : valeur de 1 octet si le bit de poids fort est égal à 0, c'est-à-dire pour une longueur comprise entre 0 et 127 B

• Longueur définie longue (forme primitive) : le premier octet (sans le premier 1) représente la longueur du champ de longueur, c'est-à-dire le nombre d'octets nécessaires au codage de la longueur.

• Longueur indéfinie (forme construite) : le premier octet 1000 0000 indique cette forme, les octets suivants représentent la valeur de la longueur et deux octets nuls (0x 00 00) sont ajoutés après l'encodage de la valeur.

Encodage de la valeur BIT STRING :

La forme de codage de la valeur BIT STRING peut être primitive ou construite. Dans la forme primitive, la chaîne est découpée en octets et un octet avant est ajouté de sorte que le nombre de bits non utilisés à la fin puisse être identifié par un entier compris entre 0 et 7. Si cet octet est égal à 0, cela signifie que tous les bits sont utilisés.

.2 Annexe B : Outil de parsage de fichier de configuration SCL

Il y a deux façons pour manipuler les fichiers ICD :

1. **Générer le code statique source modèle de fichier ICD (SCL) :**

   Cela nécessite un environnement d'exécution Java sur le système de développement pour générer le fichier de configuration. Dans le fichier make et sous le répertoire tools / model_generator :

   ```
   java -jar genmodel.jar mon_modèle.icd
   ```

   Cela générera les deux fichiers static_model.c et static_model.h. Copiez ces fichiers dans le répertoire de votre projet où le système de génération peut les trouver. Le fichier static_model.c contient la définition des structures de données constituant le modèle de données IED et contient également des valeurs préconfigurées fournies par le fichier SCL. Le code static_model.h est destiné à être inclus par votre code et définit les descripteurs que vous pouvez utiliser pour accéder efficacement au modèle de données.

2. **Générez un fichier de configuration du serveur à partir du fichier ICD (SCL) :**

   Cela nécessite également un environnement d'exécution Java sur le système de développement pour générer le fichier de configuration. L'utilisation de l'outil fourni ne diffère pas beaucoup de l'outil de génération de modèle statique. Dans le fichier make et sous le répertoire tools / model_generator :

   ```
   java -jar genconfig.jar mon_modèle.icd mon_modèle.cfg
   ```

   Cela générera le fichier my_model.cfg. Le fichier est un format de fichier texte simple contenant la description complète du modèle de données, y compris les valeurs prédéfinies et les adresses courtes facultatives. L'accès aux attributs de données se fait également par des « Handlers » au moment de l'exécution. La différence est que ces descripteurs ne sont pas connus au moment de la compilation par l'application. Au lieu de cela, les descripteurs doivent être demandés par des appels d'API, en utilisant la référence de l'objet ou l'adresse courte de l'attribut de données. Le format du fichier de configuration est conçu pour être très simple à analyser afin de conserver un code de serveur IEC 61850 réduit. Il décrit

le modèle de données, les blocs de contrôle, les ensembles de données et certains paramètres de communication tels que les adresses MAC et les paramètres de VLAN pour GOOSE.

La description du modèle est définie de manière hiérarchique. Chaque élément de modèle contenant des sous-éléments commence par une abréviation du type d'élément de modèle (balise d'élément) suivie d'une accolade. Chaque accolade ouvrante doit être complétée plus tard par une accolade fermante. Si un élément n'a pas de sous-éléments, il est terminé par un point-virgule.

Exemple : le modèle de données complet commence par le mot «MODEL» en majuscules, suivi d'une accolade (il est important que l'accolade soit le dernier caractère de la ligne) :

```
MODEL (simpleIO) {
    LD (GenericIO) {
        LN (LLN0) {
            DO (Mod 0) {
                DA (q 0 23 0 2 0);
                DA (t 0 22 0 0 0);
                DA (ctlModel 0 12 4 0 0) = 0;
            }
        }
    }
}
```

Chaque ligne ouvrant un nouvel élément de modèle commence par le spécificateur d'élément. Il peut s'agir de MODEL, LD, LN, DO ou DA pour le modèle de données, de DS, DE pour les définitions de jeu de données et de RC, LC, GC pour les définitions de bloc de contrôle. Les descriptions d'éléments de modèle contiennent des paramètres pour les éléments de modèle sous forme de liste entre accolades séparée par des espaces. Le premier paramètre est généralement le nom de l'élément.

• MODEL :

Le mot clé MODEL spécifie un nouveau modèle de données. Il ne peut y avoir qu'un modèle de données dans le fichier de configuration.

Syntaxe : MODEL (<nom du modèle>) {...}

L'élément MODEL ne peut contenir que des éléments LD en tant que sous-éléments.

• LD :

Le mot clé LD spécifie un nouveau périphérique logique. Syntaxe : LD (<nom de périphérique logique>) {...}

L'élément LD ne peut contenir que des éléments LN en tant que sous-éléments.

• LN :

Le mot clé LN spécifie un nouveau nœud logique.

Syntaxe : LN (<nom du noeud logique>) {...}

L'élément LN peut contenir des éléments DO (objets de données), DS (ensembles de données), RC (contrôle de rapport) et GC (contrôle GOOSE).

• DO :

Le mot clé DO spécifie un nouvel objet de données ou un nouvel objet de données.

Syntaxe : DO (<nom de l'objet de données> <nombre d'éléments de tableau>) {...}

Le deuxième paramètre spécifie le nombre d'éléments du tableau si l'objet de données est un tableau. Une valeur « 0 » indique que le DO n'est pas un tableau. L'élément DO peut contenir d'autres éléments DO (objets de sous-données) et DA (attributs de données).

• DA :

Le mot clé DA spécifie un nouvel attribut de données ou un nouvel attribut de données. Syntaxe pour les attributs de données de base :

DA (<nom d'attribut de données> <nombre d'éléments de tableau> <type> <FC> <options de déclenchement> <sAddr>) [= valeur]; Syntaxe pour les attributs de données construits :

DA (<nom d'attribut de données> <nombre d'éléments de tableau> 27 <FC> <options de déclencheurs> <sAddr>) {...}

L'élément DA ne peut contenir que d'autres éléments DA s'il spécifie un attribut de données construit.

Les codes de type sont les valeurs de l'énumération `DataAttributeType` qui peut être trouvée dans le fichier `iec61850_model.h`. Les codes FC sont les valeurs du type d'énumération `FunctionalConstraint` qui peut être trouvé dans le fichier `iec61850_common.h`. Les options de déclenchement sont 1 (données modifiées), 2 (qualité modifiée) ou 4 (mise à jour des données). "sAddr" est une valeur entière différente de 0. Les autres valeurs de "sAddr" ne sont pas acceptées par libiec61850.[18]

.3 Annexe C : Description du fichier de configuration SCL

Le fichier de configuration de l'IED est écrit dans un langage SCL basé sur XML et possède une extension ".icd". Ce fichier permet la configuration d'un IED, il comporte différentes sections qui définissent la communication, les priorités, les modèles de type de données, etc. Dans ce qui suit, nous allons décrire les parties d'un des fichiers ".icd" que nous avons utilisés dans ce projet.

Le fichier commence par la version XML et le codage du type de caractères informatiques utilisé (voir la première ligne du fichier). La deuxième ligne présente la source du fichier SCL, ici le site Web officiel de la IEC. Ensuite, la partie en-tête qui contient un identifiant et le nom de l'IED défini par l'utilisateur. Après, nous avons la section de communication, elle commence de la ligne 5 à la ligne 36. Cette section contient les informations de sous-réseau (ou simplement réseau) pour le IED, son adresse IP, le réseau local virtuel auquel il appartient et une configuration GSE pour les besoins de la communication GOOSE. La configuration GSE présente l'adresse MAC (puisque GOOSE utilise la couche liaison de données) et le nom du bloc de contrôle associé (ligne 18 et 26).

La section IED (de la ligne 37 à 128) décrit la configuration complète d'un IED. Il contient différents services (de la ligne 38 à 52) fournis par l'IED (en tant que serveur) et les points d'accès de l'IED spécifiques (de la ligne 53 à 127). Ce dernier contient toutes les spécifications du serveur (définitions des data sets statiques, blocs de contrôle de rapport, blocs de contrôle GOOSE et configurations de contrôle de données).

• Définitions des datsets statiques (de la ligne 58 à la ligne 77) : Cette sous-section est définie sous un LD spécifique nommé LLN0. Tout dataset doit se trouver sous le même LD qu'un RCB ou un GoCB pour pouvoir être envoyé sous forme de rapport ou de message GOOSE, respectivement. Dans ce fichier, il existe trois data sets (Events, Events2 et AnalogValues).

• Blocs de contrôle des rapports (de la ligne 79 à la ligne 88) Contient les configurations RCB (nom, data set associé, mise en mémoire tampon / non mise en mémoire tampon (i.e. buffered/unbuffered), période de déclenchement, etc.).
Dans ce cas, il y a deux RCB, le premier est "EventsRCB", il s'agit d'un RCB sans tampon (unbuffered) avec une période de déclenchement activée (appelée aussi période d'intégrité = 1000 ms) et un ensemble de données associé "Events". Le second est "Ana-

logValuesRCB", il a les mêmes configurations que le premier sauf le data set, qui est dans son cas "AnalogValues".

• Blocs de contrôle GOOSE (de la ligne 89 à la ligne 90) Il y a 2 blocs de contrôle GOOSE portant les noms "gcbEvents" et "gcbAnalogValues" et ayant respectivement les data sets "Events" et "AnalogValues".

• Configurations de contrôle des données (de 91 à 124) Cette sous-section englobe les données contrôlables et définit leur type de modèle de contrôle en associant à la DA "ctlModel" de chaque DO à la valeur correspondante (c'est-à-dire du type). Les types "ctlModel" sont expliqués dans la section suivante.

La section DataTypeTemplates (lignes 129 à 260) présente les données (DO et / ou DA) contenues dans chaque LN. Selon le type de données, d'autres informations supplémentaires peuvent exister telles que FC : contrainte fonctionnelle, q : qualité, stVal : valeur d'état, etc. À la fin de cette section (lignes 242 à 260), deux types d'énumération "EnumType" sont définis, le premier "CtlModels" est associé aux configurations de contrôle de données décrites précédemment. Il présente cinq types pour le DA "ctlModel", ces types informent le client des manières par lesquelles il peut contrôler ces données. Le deuxième type de "EnumType" est la catégorie d'origine "OrCat", il est utilisé pour indiquer l'origine d'un contrôle (par exemple, automatique ou statique, depuis la baie ou la station, etc.).

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xmlns="http://www.iec.ch/61850/2003/SCL">
  <Header id="" nameStructure="IEDName">
  </Header>
  <Communication>
    <SubNetwork name="subnetwork1" type="8-MMS">
      <Text>Station bus</Text>
      <ConnectedAP iedName="simpleIO" apName="accessPoint1">
        <GSE ldInst="GenericIO" cbName="gcbEvents">
          <Address>
              <P type="VLAN-ID">0</P>
              <P type="VLAN-PRIORITY">4</P>
              <P type="MAC-Address">01-0c-cd-01-00-01</P>
              <P type="APPID">6</P>
          </Address>
      </GSE>
        <GSE ldInst="GenericIO" cbName="gcbAnalogValues">
          <Address>
              <P type="VLAN-ID">0</P>
              <P type="VLAN-PRIORITY">4</P>
              <P type="MAC-Address">01-0c-cd-01-00-01</P>
              <P type="APPID">7</P>
          </Address>
      </GSE>
      </ConnectedAP>
    </SubNetwork>
  </Communication>
  <IED name="simpleIO">
    <Services>
      <DynAssociation />
      <GetDirectory />
      <GetDataObjectDefinition />
      <GetDataSetValue />
      <DataSetDirectory />
      <ReadWrite />
      <GetCBValues />
      <ConfLNs fixPrefix="true" fixLnInst="true" />
      <GOOSE max="5" />
      <GSSE max="5" />
      <FileHandling />
      <GSEDir />
```

```
42      <TimerActivatedControl />
43    </Services>
44    <AccessPoint name="accessPoint1">
45      <Server>
46        <Authentication />
47        <LDevice inst="GenericIO">
48          <LN0 lnClass="LLN0" lnType="LLN01" inst="">
49            <DataSet name="Events" desc="Events">
50              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="ST" lnInst="1"
    doName="SPCSO1" daName="stVal" />
51              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="ST" lnInst="1"
    doName="SPCSO2" daName="stVal" />
52              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="ST" lnInst="1"
    doName="SPCSO3" daName="stVal" />
53              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="ST" lnInst="1"
    doName="SPCSO4" daName="stVal" />
54            </DataSet>
55            <DataSet name="AnalogValues" desc="analog values">
56              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="MX" lnInst="1"
    doName="AnIn1" />
57              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="MX" lnInst="1"
    doName="AnIn2" />
58              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="MX" lnInst="1"
    doName="AnIn3" />
59              <FCDA ldInst="GenericIO" lnClass="GGIO" fc="MX" lnInst="1"
    doName="AnIn4" />
60            </DataSet>
61            <ReportControl name="EventsRCB" confRev="1" datSet="Events"
    rptID="Events" buffered="false" intgPd="1000" bufTime="50">
62              <TrgOps period="true" />
63              <OptFields seqNum="true" timeStamp="true" dataSet="true"
    reasonCode="true" entryID="true" configRef="true" />
64              <RptEnabled max="1" />
65            </ReportControl>
66            <ReportControl name="AnalogValuesRCB" confRev="1" datSet="
    AnalogValues" rptID="AnalogValues" buffered="false" intgPd="1000"
    bufTime="50">
67              <TrgOps period="true" />
68              <OptFields seqNum="true" timeStamp="true" dataSet="true"
    reasonCode="true" entryID="true" configRef="true" />
69              <RptEnabled max="1" />
```

```
70          </ReportControl>
71
72      <LogControl name="EventLog" datSet="Events" logName="EventLog" logEna
    ="false">
73        <TrgOps dchg="true" qchg="true" />
74      </LogControl>
75
76      <LogControl name="GeneralLog" datSet="" logName="" logEna="false">
77        <TrgOps dchg="true" qchg="true" />
78      </LogControl>
79
80      <Log />
81      <Log name="EventLog" />
82
83          <GSEControl appID="events" name="gcbEvents" type="GOOSE" datSet
    ="Events" confRev="2"/>
84          <GSEControl appID="analog" name="gcbAnalogValues" type="GOOSE"
    datSet="AnalogValues" confRev="2"/>
85          <DOI name="Mod">
86            <DAI name="ctlModel">
87              <Val>status-only</Val>
88            </DAI>
89          </DOI>
90        </LN0>
91        <LN lnClass="LPHD" lnType="LPHD1" inst="1" prefix="" />
92        <LN lnClass="GGIO" lnType="GGIO1" inst="1" prefix="">
93          <DOI name="Mod">
94            <DAI name="ctlModel">
95              <Val>status-only</Val>
96            </DAI>
97          </DOI>
98          <DOI name="AnIn2">
99            <DAI name="mag" sAddr="101"/>
100           <DAI name="t" sAddr="102" />
101         </DOI>
102         <DOI name="SPCSO1">
103           <DAI name="ctlModel">
104             <Val>direct-with-normal-security</Val>
105           </DAI>
106         </DOI>
107         <DOI name="SPCSO2">
```

```
108                 <DAI name="ctlModel">
109                   <Val>direct-with-normal-security</Val>
110                 </DAI>
111               </DOI>
112               <DOI name="SPCSO3">
113                 <DAI name="ctlModel">
114                   <Val>direct-with-normal-security</Val>
115                 </DAI>
116               </DOI>
117               <DOI name="SPCSO4">
118                 <DAI name="ctlModel">
119                   <Val>direct-with-normal-security</Val>
120                 </DAI>
121               </DOI>
122             </LN>
123           </LDevice>
124         </Server>
125       </AccessPoint>
126     </IED>
127     <DataTypeTemplates>
128       <LNodeType id="LLN01" lnClass="LLN0">
129         <DO name="Mod" type="INC_1_Mod" />
130         <DO name="Beh" type="INS_1_Beh" />
131         <DO name="Health" type="INS_1_Beh" />
132         <DO name="NamPlt" type="LPL_1_NamPlt" />
133       </LNodeType>
134       <LNodeType id="LPHD1" lnClass="LPHD">
135         <DO name="PhyNam" type="DPL_1_PhyNam" />
136         <DO name="PhyHealth" type="INS_1_Beh" />
137         <DO name="Proxy" type="SPS_1_Proxy" />
138       </LNodeType>
139       <LNodeType id="GGIO1" lnClass="GGIO">
140         <DO name="Mod" type="INC_2_Mod" />
141         <DO name="Beh" type="INS_1_Beh" />
142         <DO name="Health" type="INS_1_Beh" />
143         <DO name="NamPlt" type="LPL_2_NamPlt" />
144         <DO name="AnIn1" type="MV_1_AnIn1" />
145         <DO name="AnIn2" type="MV_1_AnIn1" />
146         <DO name="AnIn3" type="MV_1_AnIn1" />
147         <DO name="AnIn4" type="MV_1_AnIn1" />
148         <DO name="SPCSO1" type="SPC_2_SPCSO1" />
```

```
149       <DO name="SPCSO2" type="SPC_1_SPCSO2" />
150       <DO name="SPCSO3" type="SPC_1_SPCSO3" />
151       <DO name="SPCSO4" type="SPC_1_SPCSO1" />
152       <DO name="Ind1" type="SPS_1_Proxy" />
153       <DO name="Ind2" type="SPS_1_Proxy" />
154       <DO name="Ind3" type="SPS_1_Proxy" />
155       <DO name="Ind4" type="SPS_1_Proxy" />
156     </LNodeType>
157     <DOType id="INC_1_Mod" cdc="INC">
158       <DA name="q" bType="Quality" fc="ST" qchg="true" />
159       <DA name="t" bType="Timestamp" fc="ST" />
160       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
161     </DOType>
162     <DOType id="INS_1_Beh" cdc="INS">
163       <DA name="stVal" bType="INT32" fc="ST" dchg="true" />
164       <DA name="q" bType="Quality" fc="ST" qchg="true" />
165       <DA name="t" bType="Timestamp" fc="ST" />
166     </DOType>
167     <DOType id="LPL_1_NamPlt" cdc="LPL">
168       <DA name="vendor" bType="VisString255" fc="DC" />
169       <DA name="swRev" bType="VisString255" fc="DC" />
170       <DA name="d" bType="VisString255" fc="DC" />
171       <DA name="configRev" bType="VisString255" fc="DC" />
172       <DA name="ldNs" bType="VisString255" fc="EX" />
173     </DOType>
174     <DOType id="DPL_1_PhyNam" cdc="DPL">
175       <DA name="vendor" bType="VisString255" fc="DC" />
176     </DOType>
177     <DOType id="SPS_1_Proxy" cdc="SPS">
178       <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true" />
179       <DA name="q" bType="Quality" fc="ST" qchg="true" />
180       <DA name="t" bType="Timestamp" fc="ST" />
181     </DOType>
182     <DOType id="LPL_2_NamPlt" cdc="LPL">
183       <DA name="vendor" bType="VisString255" fc="DC" />
184       <DA name="swRev" bType="VisString255" fc="DC" />
185       <DA name="d" bType="VisString255" fc="DC" />
186     </DOType>
187     <DOType id="MV_1_AnIn1" cdc="MV">
188       <DA name="mag" type="AnalogueValue_1" bType="Struct" fc="MX" dchg="
    true" />
```

```
189       <DA name="q" bType="Quality" fc="MX" qchg="true" />
190       <DA name="t" bType="Timestamp" fc="MX" />
191     </DOType>
192     <DOType id="SPC_1_SPCSO1" cdc="SPC">
193       <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true" />
194       <DA name="q" bType="Quality" fc="ST" qchg="true" />
195       <DA name="Oper" type="SPCOperate_1" bType="Struct" fc="CO" />
196       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
197       <DA name="t" bType="Timestamp" fc="ST" />
198     </DOType>
199     <DOType id="INC_2_Mod" cdc="INC">
200       <DA name="q" bType="Quality" fc="ST" qchg="true" />
201       <DA name="t" bType="Timestamp" fc="ST" />
202       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
203     </DOType>
204     <DOType id="SPC_2_SPCSO1" cdc="SPC">
205       <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true" />
206       <DA name="q" bType="Quality" fc="ST" qchg="true" />
207       <DA name="Oper" type="SPCOperate_1" bType="Struct" fc="CO" />
208       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
209       <DA name="t" bType="Timestamp" fc="ST" />
210     </DOType>
211     <DOType id="SPC_1_SPCSO2" cdc="SPC">
212       <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true" />
213       <DA name="q" bType="Quality" fc="ST" qchg="true" />
214       <DA name="Oper" type="SPCOperate_1" bType="Struct" fc="CO" />
215       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
216       <DA name="t" bType="Timestamp" fc="ST" />
217     </DOType>
218     <DOType id="SPC_1_SPCSO3" cdc="SPC">
219       <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true" />
220       <DA name="q" bType="Quality" fc="ST" qchg="true" />
221       <DA name="Oper" type="SPCOperate_1" bType="Struct" fc="CO" />
222       <DA name="ctlModel" type="CtlModels" bType="Enum" fc="CF" />
223       <DA name="t" bType="Timestamp" fc="ST" />
224     </DOType>
225     <DAType id="AnalogueValue_1">
226       <BDA name="f" bType="FLOAT32" />
227     </DAType>
228     <DAType id="Originator_1">
229       <BDA name="orCat" type="OrCat" bType="Enum" />
```

```
230       <BDA name="orIdent" bType="Octet64" />
231     </DAType>
232     <DAType id="SPCOperate_1">
233       <BDA name="ctlVal" bType="BOOLEAN" />
234       <BDA name="origin" type="Originator_1" bType="Struct" />
235       <BDA name="ctlNum" bType="INT8U" />
236       <BDA name="T" bType="Timestamp" />
237       <BDA name="Test" bType="BOOLEAN" />
238       <BDA name="Check" bType="Check" />
239     </DAType>
240     <EnumType id="CtlModels">
241       <EnumVal ord="0">status-only</EnumVal>
242       <EnumVal ord="1">direct-with-normal-security</EnumVal>
243       <EnumVal ord="2">sbo-with-normal-security</EnumVal>
244       <EnumVal ord="3">direct-with-enhanced-security</EnumVal>
245       <EnumVal ord="4">sbo-with-enhanced-security</EnumVal>
246     </EnumType>
247     <EnumType id="OrCat">
248       <EnumVal ord="0">not-supported</EnumVal>
249       <EnumVal ord="1">bay-control</EnumVal>
250       <EnumVal ord="2">station-control</EnumVal>
251       <EnumVal ord="3">remote-control</EnumVal>
252       <EnumVal ord="4">automatic-bay</EnumVal>
253       <EnumVal ord="5">automatic-station</EnumVal>
254       <EnumVal ord="6">automatic-remote</EnumVal>
255       <EnumVal ord="7">maintenance</EnumVal>
256       <EnumVal ord="8">process</EnumVal>
257     </EnumType>
258   </DataTypeTemplates>
259 </SCL>
```

.4 Annexe D : L'outil SWIG

SWIG (Simplified Wrapper and Interface Generator) est un outil logiciel open source, permettant de connecter des logiciels ou bibliothèques logicielles écrites en C/C++ avec des langages de scripts tels que : Tcl, Perl, Python, Ruby, PHP, Lua ou d'autres langages de programmation comme Java, C#, Scheme et OCaml.

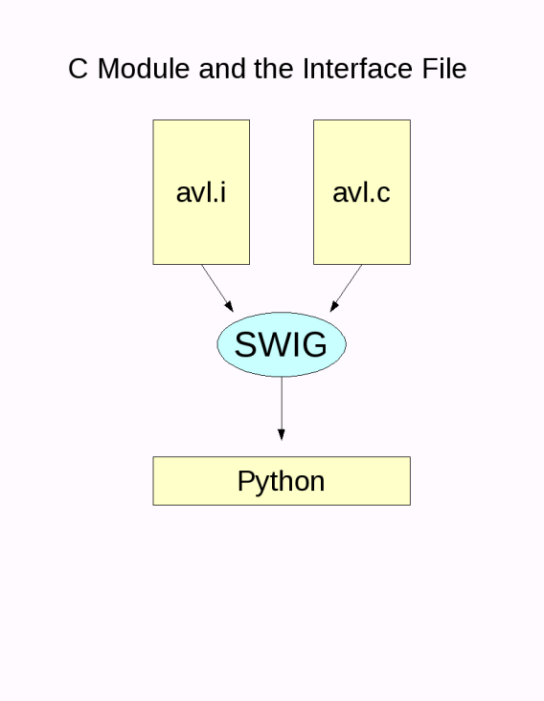

FIGURE 13 – Schéma montrant le principe de l'outil SWIG (cas du Python)

Source :`https://sunilkumarn.wordpress.com/`

Le but est de pouvoir appeler les fonctions natives (écrites en C ou C++) par d'autres langages de programmation (Python dans notre cas), passer des données types complexes à ces fonctions, protéger la mémoire contre les libérations inappropriées, hériter d'objet de classes entre langages, etc. Un fichier d'interfaçage contenant une liste de fonctions en C/C++ étant visible par un interpréteur. SWIG va compiler le fichier d'interfaçage et générer du code en C/C++ et dans le langage de programmation cible. L'outil SWIG fournit le lien entre C/C++ et l'autre langage de programmation cible. En fonction du langage, le lien se présente sous deux formes :

• Une bibliothèque partagée qu'un interpréteur peut lier à une sorte de module d'extension.

• Une bibliothèque partagée qui peut être liée à d'autres programmes compilés dans le langage cible.

SWIG n'est pas utilisé pour appeler des fonctions interprétées par le code natif, ceci est implémenté manuellement.

Voici une partie du fichier de configuration de l'outils SWIG utilisé pour l'implémentation du nœud IEC 61850 (coté client) avec des commentaires explicatifs :

```
%module iec61850 //the name of the python module that SWIG will create
%{
    #include <iec61850_client.h> //code inside %{...%} gets inserted into
    the wrapper file
%}

%include "iec61850_client.h" //include the header for functions that would
    be availbe from the created module
```

.5 Annexe E : Développement Web avec Python

Python est un langage de programmation de haut niveau interprété, orienté objet, avec une sémantique dynamique intégrée principalement pour le développement d'applications Web .

Dans l'application Web et pour le Back-end, comme libIEC61850 fournit une version pour le développement Python, nous avons utilisé le framework "Flask", qui est un micro-framework pour Python basé sur "Werkzeug" et le langage de modélisation "Jinja2". pour le Front-end "HTML" et "CSS" languges ont été impliqués en utilisant le modèle AdminLTE.

AdminLTE est un panneau de configuration gratuit, utilisé pour créer la partie administration d'un site ou d'applications Web. Grâce à la structure du modèle, nous pouvons le modifier en fonction des besoins. Ensuite, les résultats Back-end (résultats de traitement logique) sont transmis au modèle afin d'être affichés dans la page Web.

Dans ce qui suit, nous présentons à titre d'exemple une "view function" à partir du code Back-end :

```
@app.route('/discovery')
def index():
    tree = ServerTreeNode('IEDs', is_root=True)
    for server_name in ieds:
        iedconnection = ieds[server_name]['connection']
        if not iedconnection.is_connected:
            continue
        if ieds[server_name]['node'] is None:
            try:
                ieds[server_name]['node'] = make_ied_node(server_name,
    iedconnection)
            except IedClientException:
                continue


        tree.add_child(ieds[server_name]['node'])

    tree_json = json.dumps(tree)
    return render_template('discovery.html', json=tree_json)
```

Dans la ligne 1 on définit la route (le lien) pour laquel le code qui suit va s'exécuter (code de la logique). De la ligne 2 jusqu'à la ligne 17, il y'a le code Python qui définit le raisonnement du fonctionnement du service "Get data model" ou aussi nommé "Server discovery". Dans la dernière ligne on appelle à la fonction `render_template` qui prend les arguments nécessaires (le nom du modèle et les variables à transmettre au moteur de modèle en tant qu'arguments de mots clés). Flask recherchera le nom du modèle dans le dossier des modèles et mettra à jour les valeurs en fonction du résultat du traitement effectué au niveau du code Python.