Conception d’un outil pour la gestion de recharge des véhicules électriques en milieu multilogement

par

Amina CHAOUCH

MÉMOIRE PRÉSENTÉ À L’ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLEÀ L’OBTENTION DE
LA MAÎTRISE EN GÉNIE DES TECHNOLOGIES DE L’INFORMATION
M. Sc. A

MONTRÉAL, LE 23 MAI 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE RAPPORT DE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Tony Wong, directeur de mémoire
Département de génie des systèmes à l'École de technologie supérieure

M.Julio Cesar Montecinos, codirecteur de mémoire
Département de génie des systèmes à l'École de technologie supérieure

M. Georges Ghazi, président du jury
Département de génie des systèmes à l'École de technologie supérieure

M. Mohammadhadi Shateri, membre du jury
Département de génie des systèmes à l'École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 16 MAI 2023

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J’aimerais bien exprimer ma gratitude à mon directeur de recherche Tony Wong et ma reconnaissance profonde pour sa disponibilité et ses conseils pertinents qu’il m’a prodigués au cours de ce projet de recherche. Il n’était pas un superviseur ordinaire, mais aussi un vrai être humain qui m’a aussi soutenue moralement. Je suis ravie et fière à la fois d’avoir l’opportunité de rencontrer un tel homme. Je lui souhaite toujours le bonheur.

Je suis aussi reconnaissante à monsieur Julio Montecinos qui a été le responsable de la coordination de tout le projet et qui a pris du temps à lire mon rapport et à l’évaluer. Mes sincères remerciements.

Je tiens également à remercier l’organisme MITACS et l’entreprise RVE de m'avoir donnée l'opportunité de poursuivre mes études au Canada dans une institution d'aussi grande qualité que l'ETS et de réaliser un stage qui m’a donnée envie de persévérer dans le domaine de l’IoT.

Un grand merci aux membres du jury pour l’intérêt qu’ils portent à mon mémoire ainsi que pour avoir accepté à évaluer mon travail.

Je dédie ce travail à la lumière de mes jours, celle qui a toujours sacrifié pour me voir réussir, ma raison de vivre et mon bonheur, ma mère et à mon premier amour, mon soutien moral, celui qui n’a jamais cessé de me combler d’amour et d’affection, mon père.

À mon frère et mes sœurs, en les souhaitant la réussite et le bonheur.

À toute ma famille, que nous soyons toujours unis.

À tous ceux qui comptent énormément dans ma vie, je vous dédie le fruit de mes efforts durant mes études.

Conception d'un outil pour la gestion de recharge des véhicules électriques en milieu multilogement

Amina CHAOUCH

RÉSUMÉ

Le présent projet, réalisé en partenariat avec l'entreprise RVE, entre dans le cadre de la gestion de recharge des véhicules électriques. En effet, les bâtiments multilogements ont des exigences en termes de capacité d'énergie électrique. Les installations électriques de ces bâtiments sont dimensionnées de manière à répondre aux besoins des bâtiments au moment de leur construction. C'est pourquoi les installations électriques ne prévoient pas nécessairement une grande marge de manœuvre pour des besoins futurs. L'ajout de bornes de recharge pose alors un défi dans le maintien de cette marge de manœuvre. D'autant plus que la capacité disponible pour chaque logement d'un multilogement est généralement plus faible que celle d'une maison individuelle. Il faut donc mesurer et contrôler le transfert de puissance entre les tableaux de distribution et les véhicules électriques pour ne pas dépasser la capacité maximale permise.

L'approche IIoT (Internet Industriel des Objets) est adoptée pour réaliser la gestion et la surveillance des bornes de recharge. Notre travail propose un outil de gestion basé sur une passerelle IIoT implantant les protocoles MQTT (Message Queuing Telemetry Transport) et LoRa (Long Range) pour la lecture des puissances transférées en temps réel. L'interaction entre l'outil de gestion et la passerelle est réalisée au moyen du protocole MQTT. Un nombre de tests ont été réalisés afin valider le fonctionnement et la performance du système de l'outil. Les résultats obtenus ont démontré que l'outil conçu est capable d'effectuer la communication à distance et possède une bonne gestion des évènements asynchrones du système de mesure déployé dans des multilogements.

Mots-clés : IIoT, passerelle, coordonnateur, MQTT, LoRa, ThingSpeak, outil de gestion, GUI

Design of a monitoring tool for electrical charging stations in a multi- dwelling environment

Amina CHAOUCH

ABSTRACT

This work is part of a research project on electric vehicle recharge management initiated by our industrial partner RVE enterprise. Multi-unit buildings or multi-dwellings have specific requirements in terms of electrical energy capacity. The electrical installations of these buildings are sized to meet the needs of the buildings at the time of their construction. This is why electrical installations do not necessarily provide a large safety margin for future needs. Therefore, the addition of charging stations poses a challenge in maintaining this leeway. Especially since the capacity available for each unit of a multi-dwelling is generally lower than that of a single-family house. It is thus necessary to measure and control the transfer of power between the main switchboards and the electric vehicles so as not to exceed the allowed maximum capacity.

The IIoT (Industrial Internet of Things) approach is adopted to manage and monitor charging stations. Our work proposes a management tool based on an IIoT gateway implementing the MQTT (Message Queuing Telemetry Transport) and LoRa (Long Range) protocols for sampling the powers transferred in real time. The interaction between the management tool and the gateway is carried out using the MQTT protocol. A number of tests were performed to validate the operation and performance of the tool. The results obtained demonstrated that the designed management tool is capable of performing remote communication and can be used with asynchronous events generated by the measurement system in a multi-dwelling setting.

Keywords: IIoT, gateway, coordinator, MQTT, LoRa, ThingSpeak, management tool, GUI

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACK	Acknowledgement
AI	Artificial Intelligence
AP	Access Point
CLI	Command Line Interface
CoAP	Constrained Application Protocol
DCC	Charge Controller
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IIoT	Industrial Internet of Things
IoT	Internet of Things
IOT	In-Orbit Testing
IP	Internet Protocol
ISO	International Organization for Standardization
LBT	Listened Before Talk
LoRa	Long Range Radio Wide Area
MQTT	Message Queuing Telemetry Transport
NOACK	No Acknowledgement
QoS	Quality of Service
REST	Representational State Transfer
RL	Reinforcement Learning

RSSI	Received Signal Strength Indication
UART	Universal Asynchronous Receiver/Transmitter
UCD	User-Centered Design
UCSD	User-Centered System Design
UI	User interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UX	User Experience
Wi-Fi	Wireless Fidelity
WOR	Wake Over Air
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

INTRODUCTION

0.1 Contexte et problématique

Le nombre de véhicules électriques au Québec est en constante croissance avec plus de 179 000 unités en circulation en date du 31 décembre 2022 (AVÉQ, 2023) .Un nombre de ces véhicules électriques est rechargé dans des installations électriques appartenant à des multilogements. Le terme multilogement désigne, dans ce mémoire, les immeubles en copropriété et des bâtiments résidentiels à logements multiples. Dans les tableaux électriques de ces multilogements, la somme des capacités des circuits est normalement supérieure à la capacité de l'entrée électrique principale. Cette différence est justifiée par le fait que les circuits électriques des unités de logement ne sont pas tous utilisés au maximum de leur capacité en même temps. Il s'agit là d'une justification acceptée dans la pratique puisque le Code de construction du Québec utilise cette justification dans le calcul de la capacité minimale des bâtiments (Hydro-Québec, 2022).

Les installations électriques sont dimensionnées de manière à répondre aux besoins des bâtiments au moment de leur construction. Afin de restreindre les coûts de construction, les installations électriques ne prévoient pas nécessairement une grande marge de manœuvre pour des besoins futurs. De façon générale, la consommation électrique et résidentielle a tendance à diminuer à mesure que la technologie progresse. Or, l'ajout des bornes de recharge dans les tableaux de distribution électrique pose un défi de taille dans le maintien de cette marge de manœuvre. D'autant plus que la capacité disponible pour chaque logement d'un multilogement est plus faible que celle d'une maison individuelle. En effet, le tableau de distribution de certains logements peut n'avoir qu'une capacité de 100 A alors qu'une borne de recharge est habituellement cotée à 30 A (Hydro-Québec, 2022). La marge de manœuvre est donc moins grande que dans une maison individuelle dotée d'un tableau de 200 A. Donc, l'enjeu de la capacité électrique devient acéré dans un multilogement et chaque borne de recharge ajoutée peut devenir problématique.

Pour pallier ce problème, on emploie souvent des dispositifs de contrôle de charge (DCC) afin de maintenir une marge de manœuvre acceptable entre la somme des capacités des circuits et la capacité de l'entrée principale. Chacune des bornes de recharge est alors gérée par un DCC. Ces dispositifs de contrôle sont conçus pour interrompre la recharge et éviter l'excès de la capacité allouée. Le cas d'utilisation type consiste à maximiser la recharge des véhicules électriques en dehors des pointes de consommation qui sont le matin et les heures de souper. En principe, ces DCC sont capables de tirer profit de la capacité résiduelle inutilisée pour recharger les véhicules électriques. Cependant, pour pouvoir affirmer une telle capacité des dispositifs de contrôle, il nous faut des preuves tangibles et concrètes. C'est dans ce contexte qu'un système d'acquisition de données a été envisagé pour mesurer la puissance électrique transférée aux bornes de recharge dans des multilogements. La Figure 0.1 est une illustration de ce système.

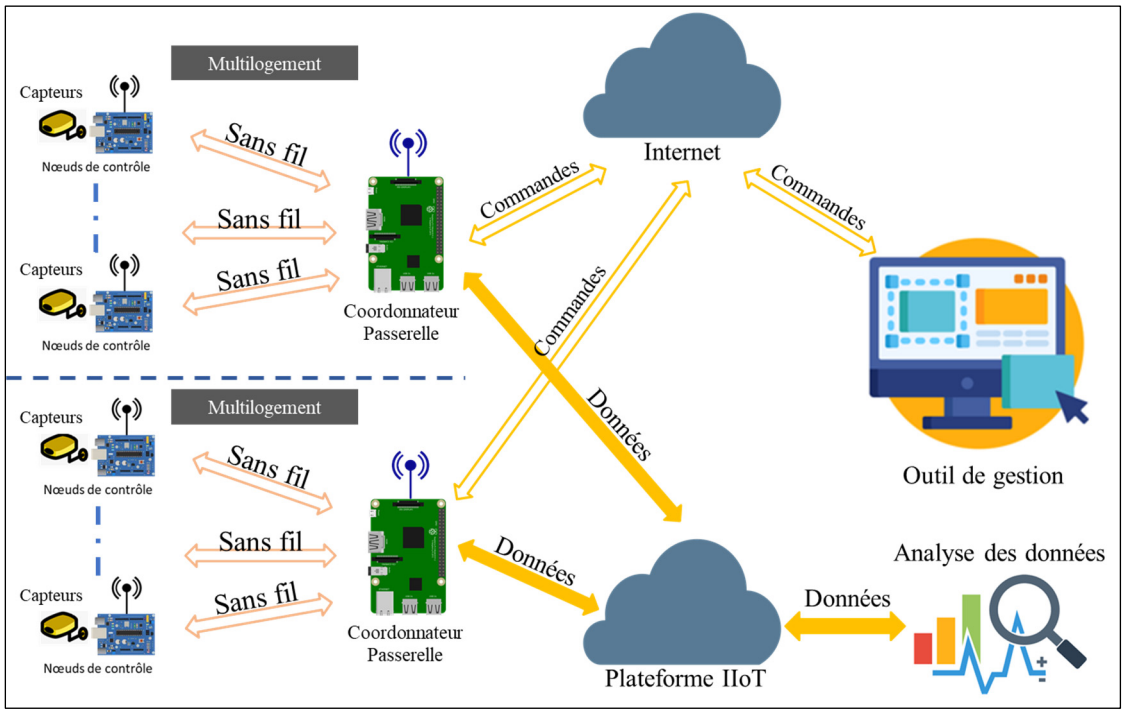

Figure 0.1 Système d'acquisition de données envisagé

Ce système d'acquisition de données permet la mesure et surveillance de la puissance transférée dans les bornes de recharge en temps réel. Ceci est réalisé par la lecture des puissances à chaque *N* minutes. La période d'échantillonnage du système d'acquisition est réglable en utilisant l'outil de gestion. L'utilisateur de l'outil de gestion peut également obtenir des rapports d'évènements et voir s'il y a des dépassements de la capacité permise des puissances transférées. Il sera ainsi en mesure de régler la gestion de la puissance de la recharge des véhicules électriques. En formant cette base de données collectées, on permet à l'utilisateur de garder trace des problèmes qui ont été engendrés.

Pour mesurer la puissance électrique transférée aux véhicules électriques d'un multilogement, on doit installer des capteurs dans les DCC alimentant les bornes de recharge. On voit dans la Figure 0.1 que les capteurs sont connectés à des nœuds de contrôle exécutant sur des cartes Arduino. Les puissances lues par les capteurs sont transférées par les nœuds de contrôle à un coordonnateur exécutant sur une carte Raspberry Pi. C'est ce dernier qui est responsable de transmettre les données reçues sur une plateforme infonuagique Internet industrielle des objets (IIoT). D'où la désignation « passerelle » accordée aux coordonnateurs. Ainsi, chaque multilogement aura un coordonnateur et des nœuds de contrôle sous sa responsabilité. De ce fait, tous les coordonnateurs déversent les données sur la même plateforme IIoT afin de permettre le traitement et l'analyse des données en provenance des capteurs. Dans ce système d'acquisition, la surveillance et le contrôle des coordonnateurs sont déléguées à un outil de gestion qui est délocalisé à l'extérieur des multilogements. Grâce à la communication par l'Internet, l'outil de gestion peut connaître l'état des coordonnateurs, des nœuds de contrôle et leur transmettre des commandes d'opération.

La raison principale de cet arrangement du système d'acquisition réside dans le fait que les DCC sont installés dans des tableaux de distribution à l'intérieur des chambres électriques. Le raccordement filaire de chacun des capteurs au coordonnateur engendra des coûts beaucoup trop élevés. La communication sans fil permet alors à un seul coordonnateur de communiquer avec les nœuds de contrôle situés dans différentes chambres électriques du même

multilogement. Les puissances mesurées des DCC sont envoyées sur une plateforme IIoT car le prélèvement sur place des données est tout simplement impraticable. Les multilogements sont situés dans des régions différentes et le déplacement sur ces lieux pour recueillir les données exigerait trop de main-d'œuvre.

Le système d'acquisition de données de la Figure 0.1 possède quatre (4) couches clés qui sont :

- La couche physique: qui comporte des capteurs pour la détection et la collecte des informations de l'environnement.

- La couche réseau: qui est responsable de la connexion des objets industriels pour la transmission des données.
- La couche de traitement de données : elle traite les données qui proviennent de la couche réseau.

- La couche application : elle présente l'outil logiciel utilisé pour contrôler et surveiller les appareils connectés. Elle peut être présentée sous forme d'une application mobile/web disposant d'une interface utilisateur.

0.2 Contraintes

Ce projet de recherche est une collaboration entreprise – université. Le système d'acquisition de données envisagé a été développé en synergie avec le milieu industriel. Le choix des capteurs de puissance, des cartes électroniques Arduino et Raspberry Pi a été décidé par les parties prenantes de ce projet. Il en est de même pour le choix des technologies de communication. Ainsi, le mandat de réalisation comprend l'application de la technologie sans fil *Long Range* (LoRa) dans le contrôle des nœuds par les coordonnateurs et l'intégration du protocole MQTT dans le contrôle des coordonnateurs par l'outil de gestion. Il est nécessaire de concevoir des solutions utilisant ces deux technologies sur des plateformes matérielles de

type ordinateur monocarte ou système embarqué. Donc, la maîtrise des technologies LoRa et MQTT est essentielle et elles représentent les contraintes majeures de ce projet de recherche.

0.3 Contributions

Ce projet de recherche concerne la conception et l'implantation des couches réseau et l'application du système d'acquisition des données. Sans l'apport de ces couches, le système d'acquisition ne pourra être concrétisé. Enfin, le choix d'une technologie sans fil pour la communication entre les coordonnateurs et les nœuds de contrôle est le choix d'un protocole de communication entre l'outil de gestion et les coordonnateurs font aussi partie des tâches de conception. La réalisation de ces objets technologiques contribue donc à la cueillette automatisée des données de la recharge des véhicules électriques. À l'heure actuelle, on connaît peu sur les habitudes de recharge des véhicules électriques en multilogement au Québec. L'obtention de ces données donnera un portrait plus juste des fluctuations de charge dans ces immeubles. De plus, les données recueillies serviront à valider d'une façon concrète la capacité des DCC à contrôler la recharge des véhicules électriques.

Notre contribution consiste principalement à la conception d'un outil de gestion qui est capable d'interagir avec des dispositifs IIoT. Cette interaction est basée sur l'utilisation du protocole de communication MQTT et permet à l'utilisateur de contrôler ces dispositifs à distance et de recevoir des rapports d'évènements en temps réel. Ainsi, notre outil est capable de commander les nœuds de contrôle par l'intermédiaire des coordonnateurs et de lire leurs réponses. L'utilisation de la communication MQTT montre son efficacité dans le maintien de la communication entre l'outil de gestion, les coordonnateurs et les nœuds de contrôle du système d'acquisition des données.

0.4 Objectifs de recherche

L'objectif général de notre projet est de concevoir les couches réseau et application qui permettront le bon fonctionnement du système d'acquisition des données. La communication sans fil doit pouvoir transporter des données dans un milieu bruité électriquement. L'outil de gestion doit être capable de communiquer à distance avec les coordonnateurs installés dans divers multilogements. Ainsi, à partir de cet objectif principal, nous avons identifié les sous-objectifs suivants:

- Concevoir un outil de gestion qui permettra à des opérateurs de contrôler et surveiller les coordonnateurs et les nœuds de contrôle du système d'acquisition de données.

- Concevoir un le système de communication sans fil pour le transfert des données entre les nœuds de contrôle et le coordonnateur assigné.

- Concevoir le système de communication permettant aux coordonnateurs situés dans différents multilogements de recevoir des commandes en provenance de l'outil de gestion.

0.5 Plan du mémoire

Le premier chapitre présente la revue littéraire effectuée sur les concepts d'un système IIoT et de ses principaux composants. Ce chapitre se concentre également l'approche « expérience utilisateur » (UX) dans la conception de l'outil de gestion et les protocoles utilisés dans la communication coordonnateur – nœuds de contrôle et la communication outil de gestion – coordonnateur.
Le deuxième chapitre détaille la conception de l'outil de gestion et de son développement. Le choix des composants de l'interface utilisateur ainsi que l'implantation technique du système de communication MQTT sont également expliqués. Le troisième chapitre est consacré à la conception du système de communication des coordonnateurs. Ce chapitre présente le choix des composants matériels pour la communication LoRa entre les nœuds de contrôle et le

coordonnateur assigné. L'intégration de la communication LoRa et de la communication MQTT au sein des coordonnateurs représente un défi pour ce projet de recherche.

Enfin, le chapitre 4 présente les résultats de tests de chacun des composants du système de communication. Ces tests ont pour but de valider le bon fonctionnement du système dans son ensemble.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'Internet industriel des objets (IIoT) est considéré comme une technologie des systèmes qui permet un meilleur contrôle des procédés en utilisant l'infonuagique (*cloud computing*). Cette technologie s'inscrit à l'ère de l'industrie 4.0, qui est définie comme une évolution vers une technologie intelligente dans la fabrication en mettant l'accent sur la connectivité (A. Karmakar et al., 2019).

L'IIoT consiste à obtenir les informations dont chaque consommateur a besoin quand il le souhaite. Il est également utilisé pour la transformation numérique qui consiste à numériser une entreprise dans la façon dont nous devons passer à un espace de données organisé unifié. Cette numérisation est réalisée en utilisant la société internationale de norme d'automatisation (ISO) pour développer une interface automatisée entre entreprise et systèmes de contrôle. L'IIoT consiste à prendre des décisions à partir des informations reçues en temps réel et non à partir d'un rapport créé à partir des données d'hier. Habituellement, cette technologie concerne des dispositifs interconnectés, des capteurs et de nombreux autres dispositifs qui sont mis en réseau et qui communiquent avec des applications industrielles pilotées par ordinateur via des gestionnaires logiciels.

Dans ce chapitre, nous allons traiter deux aspects de l'IIoT. Le premier concerne le pilotage d'un système IIoT par le biais d'un outil logiciel de gestion. Cet outil de gestion sert à coordonner les activités et le contrôle du système IIoT et est indispensable à son bon fonctionnement. Ici, l'étude portera sur la conception de l'interface utilisateur (UI) de cet outil de gestion. On s'intéresse donc à l'utilisation opérationnelle de l'outil de gestion et à sa présentation graphique.

Le deuxième aspect abordé est la communication entre les dispositifs connectés et la façon dont on reçoit et traite les données à partir d'une passerelle de communication. Cette dernière est un élément crucial d'une infrastructure IIoT. C'est à travers cette passerelle de communication que l'outil de gestion entre en interaction avec les dispositifs connectés. Ainsi, la coordination des activités et le contrôle d'un système IIoT reposent directement sur les capacités de cette passerelle de communication.

1.2 UI de l'outil de gestion

L'utilisation des UI dans l'IIoT est devenue de plus en plus un phénomène répandu en raison des avantages que cela peut offrir en termes d'expérience utilisateur (UX). Le terme UX a été inventé par Don Norman en 1993 (Jakob Nielsen, 2017). Il décrit d'une façon générale les différents aspects de l'interaction des personnes avec un service ou un produit. Un aspect spécifique de ce concept est l'UI qui cherche à faciliter l'interaction entre l'humain et la machine (HMI) en considérant la capacité de l'utilisateur et de son environnement. Parmi les exemples nombreux de l'HMI, on peut considérer l'interaction d'un utilisateur avec son téléphone intelligent, qui avec quelques gestes sur son écran tactile, il peut visualiser des vidéos ou envoyer des messages, et les domotiques que le propriétaire peut contrôler avec une simple commande vocale. Ainsi, le lien entre UX et UI est devenu fort : afin de concevoir une bonne UI, il est essentiel qu'une UX soit positive.

1.2.1 Évolution de la conception des UI pour l'IIoT

La conception des UI est passée par deux phases principales définies par (Wigdor & Wixon, 2011) qui sont : la phase de saisie des commandes (CLI) puis la phase de développement de l'interface utilisateur graphique (GUI). Dans les premiers ordinateurs, il n'y avait pratiquement aucune interface utilisateur, à part quelques boutons sur la console de l'opérateur. L'interface utilisateur a évolué avec la mise en place de l'interface de ligne de commande. Les utilisateurs

utilisaient un clavier et un ensemble de commandes pour contrôler l'échange d'informations avec l'ordinateur. L'inconvénient du CLI est qu'il est compliqué à utiliser pour les utilisateurs. Ils doivent apprendre la syntaxe et la liste des commandes qui peuvent être exécutées. Cependant, une interface GUI permet à l'utilisateur d'interagir avec un ordinateur grâce à des éléments visuels tels les boutons, les listes déroulantes, etc., ce qui rend l'interaction plus intuitive. Ainsi, l'opérateur nécessite moins de compétences techniques que les CLI.

Avec l'utilisation quotidienne croissante des ordinateurs accompagnés du progrès technologique, la conception des UI a été évoluée. Les premières conceptions d'interfaces graphiques étaient simples et ont fourni des fonctionnalités de base, telle que l'apparition du système NLS (système oN-Line) développé par Douglas C. Engelbart (Engelbart & English, 1968) . C'était le premier système qui a introduit les principes de l'interface graphique moderne en ajoutant la souris, le clavier et l'écran (Aceituno & Roussel, 2014) . Il convient également de mentionner la visite de Steve Jobs avec son équipe au PARC (Palo Alto Research Center) Xerox de 1979, qui lui a permis de découvrir la souris, les fenêtres, les icônes et d'autres technologies qui avaient été développées au PARC. Apple, sous la gouverne de Steve Jobs, a ensuite commencé son projet Lisa en 1979 (Birrs, 1984) qui a été parmi les premières machines possédant une GUI. Il s'agit d'une station de travail destinée à améliorer la productivité des employés de bureau (Birrs, 1984).

Au cours des années, la conception des GUI est devenue de plus en plus sophistiquée. L'intégration de la conception de l'interface graphique dans les appareils IIoT est devenue importante, car ces appareils sont de plus en plus répandus dans les entreprises et les industries. Les appareils IIoT ont souvent un espace d'écran limité et nécessitent une interface conviviale pour interagir avec eux. Cela a conduit au développement de nouveaux principes de conception de l'interface graphique tels que le minimalisme et l'utilisation de gestes et de commandes vocales. De plus, de nombreux systèmes IIoT utilisent désormais des applications mobiles ou des sites Web comme UI principale pour le contrôle ou la surveillance des dispositifs.

Des études de recherche ont été menées au fil des années pour améliorer la conception des UI et produire de l'UX plus transparente et agréable pour l'opérateur. L'un des premiers travaux sur la conception d'interfaces utilisateur IIoT a été réalisé par (W. D. Kelley et al., 2000). Il a utilisé pour la mesure de la tension d'une antenne de satellite une fenêtre d'interface utilisateur graphique (GUI) sur la console de commande de l'IOT (In-Orbit Testing). Il est à noter que l'IOT réfère aux tests faits en orbite. Cette interface permet à l'opérateur de saisir les paramètres de mesure. La configuration faite par l'utilisateur avec l'interface peut ensuite être enregistrée dans un fichier permanent sous un nom unique qui peut être exécuté ultérieurement pour effectuer la mesure telle qu'elle a été configurée. Avec un simple clic sur le bouton OK de la fenêtre, l'opérateur peut alternativement lancer les calculs. Les interfaces GUI ont été aussi développées dans les domotiques pour faciliter la surveillance et le contrôle des appareils, tel est le cas dans le travail de Kumar et Pati (P. Kumar & U. C. Pati, 2016). Ils constatent que l'interface GUI est adaptée aux personnes peu compétentes comme les personnes âgées. Dehingia a présenté le développement d'un système d'interface GUI pour aider les personnes âgées lors de l'exécution d'activités quotidiennes dans la cuisine comme la préparation des boissons, le nettoyage, la commande des aliments dans la cuisine (S. Dehingia et al., 2015). L'interface proposée génère des indices sous forme de textes et de graphiques qui constituent un retour visuel de l'état d'interaction de l'utilisateur et facilite la sélection des éléments à utiliser pour l'activité donnée.

Des recherches ont aussi porté leur attention sur l'intégration des interfaces utilisateur pour contrôler la consommation énergétique. (Xu et al., 2018) par exemple, a développé une interface GUI générique adaptée de manière flexible à divers types de bâtiments pour surveiller la consommation énergétique. (J. B. Ibarra et al., 2019) a conçu une interface utilisateur graphique (GUI) qui enregistre la consommation d'énergie d'une prise en utilisant la technologie RFID.

Dans l'ensemble, la conception d'interfaces utilisateur pour les dispositifs IIoT est une tâche complexe qui nécessite de considérer le contexte de l'application, l'environnement physique et

les capacités du dispositif et de l'utilisateur. Toutefois, les conceptions minimalistes et les interfaces utilisateur simples sont également de plus en plus populaires grâce à leur faible coût et leur précision dans l'exécution des tâches, en plus de leur capacité à rendre les appareils IIoT plus accessibles aux utilisateurs non techniques.

1.2.2 Tendances et défis

Actuellement, plusieurs tendances techniques sont introduites dans la conception des UI dans le domaine de l'IIoT. Parmi lesquels, on distingue l'exploit de l'apprentissage automatique et de l'intelligence artificielle (AI). On peut ainsi remarquer, par exemple, l'intégration de la technologie vocale et/ou motrice pour permettre à l'utilisateur de contrôler les dispositifs IIoT avec sa voix et/ou ces gestes corporels. D'où l'UX est devenue plus naturelle. Ces interfaces sont appelées interfaces utilisateur naturelles (NUI), comme son nom l'indique, il rend l'interaction entre l'homme et la machine plus naturelle et conversationnelle. En revanche, ce type d'interface présente des inconvénients. (Gonzalez Calleros et al., 2018) a montré que l'interaction naturelle avec l'utilisateur n'est pas vraiment naturelle en se basant sur l'analyse de l'interaction appuyée sur les gestes dans les environnements virtuels, ainsi de l'interaction basée sur la souris et le clavier. Les expérimentations ont montré que l'interaction avec la souris et le clavier a été plus précise. Le temps d'exécution de la tâche avec l'interaction gestuelle a été plus long et il y avait des tâches non terminées. De plus, l'utilisateur a fait plus d'erreurs quand il a utilisé le système sans les équipements de l'ordinateur. Dans ce cas-là, on peut conclure que pour l'exécution des tâches importantes telles que les tâches dans l'industrie, l'interface graphique typique est mieux pour éviter les erreurs qui peuvent emmener à des failles catastrophiques.

Parmi les tendances qui se basent sur l'intelligence artificielle et/ou le traitement du langage naturel, il y a les interfaces adaptatives qui sont devenues de plus en plus communes dans l'IIoT comme elles permettent à l'utilisateur d'interagir avec les appareils intelligents d'une façon plus personnalisée. (Rothrock et al., 2002) les a définies comme des systèmes qui

adaptent leurs affichages et les actions disponibles aux objectifs et aux capacités actuels de l'utilisateur en surveillant le statut de l'utilisateur, l'état du système et la situation actuelle. Cependant, la conception de ce type d'interface est un processus plus long et plus méthodique que d'autres interfaces utilisateur non adaptatives, ce qui explique l'augmentation de leur coût par rapport aux autres interfaces (admin, 2018). La conception minimaliste des interfaces reste toujours un bon choix si le budget des entreprises est limité.

Autres défis courants de la conception d'interfaces utilisateur pour l'IIoT se présentent avec les tendances qui apparaissent, comme la garantie de la sécurité et la confidentialité des données (Amer, 2016), la gestion de la complexité croissante des dispositifs IIoT et la compatibilité avec différents systèmes d'exploitation.

1.2.3 Méthodologies utilisées dans la conception des UI

Il existe plusieurs méthodologies et approches couramment utilisées pour la conception des UI pour l'IIoT, notamment la conception centrée utilisateurs (UCD) qui est basée sur l'UX. La conception de l'UX selon Hartson et Pyla est un processus composé de quatre étapes (Hartson & Pyla, 2012). La première étape est « l'analyse » qui consiste à effectuer des recherches sur les utilisateurs et des recherches contextuelles. Ensuite, la « conception » qui présente l'ensemble spécifique d'activités visant à créer un modèle conceptuel. Le « prototypage » est le modèle conceptuel concret possédant les fonctionnalités souhaitées. Finalement, « l'évaluation » consiste à évaluer le modèle qui a été prototypé dans chaque itération du cycle de développement.

Le terme UCD a été introduit par D. Norman en 1986 (Norman & Draper, 1986). Dans cette approche, la conception centrée sur l'utilisateur signifie qu'il faut commencer par bien comprendre les utilisateurs et les besoins auxquels la conception est censée répondre. Cette méthode implique la participation active des utilisateurs dès la première étape pour éviter des

itérations inutiles. Elle se base sur leurs définition, objectifs, besoins et exigences de la tâche. La majorité des décisions de conception sont prises en fonction de la façon dont elles fonctionnent pour les utilisateurs, d'où l'orientation de la conception pour des techniques d'interaction spécifiques. Par contre, selon Gulliksen et al., il n'existe pas de définition unique ni de méthodes de mise en œuvre unique de la UCD (Gulliksen et al., 2003).

1.2.3.1 Principes

Les chercheurs ont exploré de différents principes dans la conception des UI. L'utilisabilité est l'un des aspects importants de l'UX et est souvent considérée comme un critère d'évaluation des UI ou GUI centrée sur l'UCD. En effet, l'utilisabilité est définie dans la norme ISO 9241-11 comme étant *«* la mesure dans laquelle un produit peut être utilisé par des utilisateurs spécifiés pour atteindre des objectifs spécifiés avec efficacité, efficience et satisfaction dans un contexte d'utilisation spécifié. *»* (Marc, 2022). D'ailleurs, Molina et al. ont considéré que la réussite d'une application logicielle dépend largement de sa convivialité et de sa courbe d'apprentissage (Molina et al., 2012).

D'autres aspects clés sont introduits dans l'ouvrage « The psychology of everyday things » (l'ancienne version de « Design Of Everyday Things ») considéré comme une classique dans le domaine de l'HMI et l'UCD. Cet ouvrage explique comment la conception des objets modifie nos interactions avec eux (Norman, 1988). L'auteur souligne l'importance de comprendre le modèle mental de l'utilisateur et comment la conception d'un objet peut soit le soutenir ou le contredire. On y retrouve les principes fondamentaux régissant l'interaction avec les objets qui sont souvent considérés dans la conception des UI :

- Les affordances: fait références aux propriétés de l'objet qui suggèrent comment ce dernier devrait être utilisé.

- Les signifiants: qui sont les fonctionnalités qui indiquent comment l'objet pourra être utilisé. Ce sont des signes qui communiquent un sens à l'utilisateur comme les boutons dans les interfaces sur lesquelles est écrit START/STOP.
- Le mappage/ la cartographie: qui représente la relation entre l'interface utilisateur et les actions qu'elle contrôle pour construire une disposition efficace et compréhensible pour l'affichage tel est le cas d'une interface graphique ou les boutons sont organisés physiquement d'une manière qui correspond aux fonctions qu'elle contrôle comme le réglage de puissance, luminosité, etc.
- Le retour d'information : est le moment où le résultat d'une action est communiqué à l'utilisateur et doit être immédiat. Si par exemple un utilisateur envoie une commande en utilisant une interface GUI, un message de retour affiché sur cette interface indiquant l'échec ou la réussite de l'envoi de ce message aide l'utilisateur à savoir l'état de l'opération.

- Les modèles conceptuels : est le modèle mental qui explique le fonctionnement d'un système. Les croquis et les prototypes sont des exemples de la définition d'un modèle conceptuel, car ils servent d'explorer et de communiquer les différentes parties d'une interface, les liaisons entre elles et de faire de petits tests des idées avec les parties prenantes avant d'investir dans le développement du produit final.

La façon dont on applique les directives de conception dépend également des contextes d'utilisation, de la plateforme, de la conception et du type d'interaction que les utilisateurs auront avec elle, en tenant compte de leurs attentes et capacités. Ces directives permettent également de tester la fonctionnalité et l'utilisabilité du produit, pour enfin obtenir des UI qui offrent une expérience transparente et agréable à ses clients.

1.3 Éléments de base d'une passerelle IIoT

Une passerelle de communication IIoT joue le rôle d'un pont qui assure la communication entre les dispositifs connectés et une plateforme infonuagique comme il est montré dans la Figure 1.1. En effet, elle reçoit les données des dispositifs connectés puis les transfère au service infonuagique pour l'analyse et traitement. La passerelle peut également remplir d'autres fonctions telles la sécurité et la transformation des données. L'architecture d'une passerelle IIoT peut varier en fonction des exigences spécifiques de l'application IIoT.

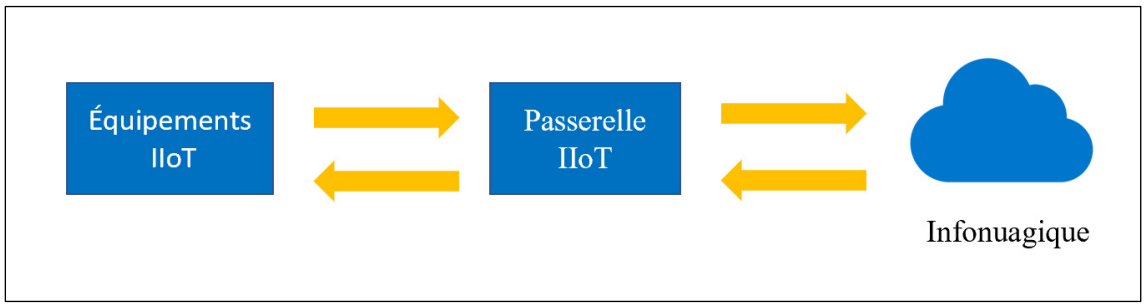

Figure 1.1 Relation entre les dispositifs IIoT, une plateforme infonuagique et une passerelle de communication

Ce processus utilise généralement des protocoles de communication différents pour connecter les dispositifs IIoT. Parmi ces protocoles, on distingue le « *Message Queuing Telemetry Transport* » (MQTT) et le « *Representational State Transfer* » (REST). Soni & Makwana ont remarqué que le protocole MQTT est davantage présent dans des applications à faible consommation d'énergie (Soni & Makwana, 2017).

1.3.1 Protocole de communication MQTT

Le MQTT est un protocole de messagerie basé sur le modèle de communication publication et d'abonnement. Il a été développé par IBM à la fin des années 1990 pour pouvoir envoyer des données entre des appareils dans un environnement à réseau faible et à forte latence (Soni & Makwana, 2017). L'une des caractéristiques de MQTT est son modèle de communication,

qui permet aux différents intervenants d'échanger des messages sans le besoin de se référer à une interface de programmation (Botta et al., 2020).

Le mécanisme de publication et d'abonnement repose sur un courtier de messages spécial, appelé « *broker* », qui agit comme un intermédiaire entre les publieurs et les abonnés. Ce mécanisme est illustré dans la Figure 1.2. Un appareil industriel peut publier des messages pour les autres dispositifs. Il peut aussi s'abonner à un sujet particulier pour la réception des messages associés. Par exemple un dispositif A1 publie sur un sujet et le message est hébergé par le broker. Le dispositif A2 reçoit les messages en s'abonnant au sujet sur lequel le dispositif A1 fait ses publications. De cette façon, les dispositifs A1 et A2 ne sont pas en communication directe et ils n'ont pas à connaître les adresses de l'expéditeur ou du destinataire. La contrainte à respecter entre le publicateur et l'abonné est de s'entendre sur le sujet (*topic*) dans la transmission des messages. On peut considérer un sujet MQTT (*topic* MQTT) comme un canal de transmission. Un client MQTT doit s'abonner à un sujet pour recevoir les messages qui y sont publiés. De même, le publicateur doit transmettre des messages sur le sujet approprié pour que l'abonné puisse les recevoir. Ce modèle de communication peut être appliqué sur un très large réseau et assurer la communication par l'intermédiaire d'un ou plusieurs brokers.

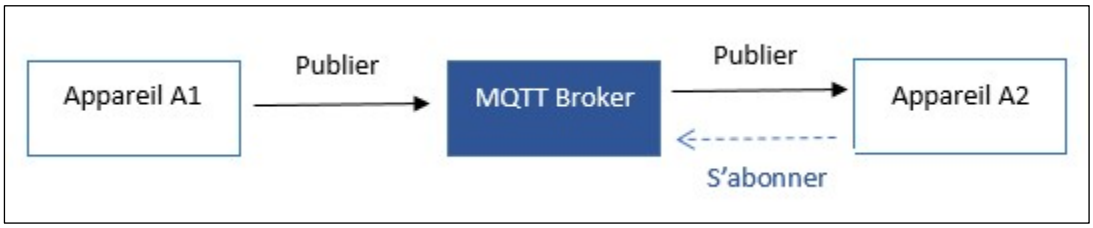

Figure 1.2 Modèle publication et abonnement du protocole MQTT

Les résultats de la recherche de Botta et al. montrent d'excellentes performances de l'architecture de la passerelle IoT en utilisant le protocole MQTT avec un broker de type

Mosquitto (Botta et al., 2020). Le MQTT présente également de bonnes fonctionnalités en termes de sécurité. En effet, le MQTT utilise le « *Transport Layer Security* » (TLS), qui est la couche cryptographique du protocole TCP/IP (The HiveMQ Team, 2015). Le MQTT fournit un mécanisme permettant à un client de s'authentifier à l'aide d'un nom d'utilisateur et d'un mot de passe. Lorsque le client MQTT tente de se connecter à un broker, il doit fournir un nom d'utilisateur et un mot de passe pour que la connexion soit acceptée.

L'utilisation de MQTT dans une application IoT présente d'autres avantages. Le protocole est « léger », car l'entête de ce protocole n'est que 2 octets comparé à 8000 octets d'une communication HTTP (Kim et al., 2015). Ce protocole est conçu pour être utilisé dans des environnements contraignants tels que les dispositifs embarqués qui ont une capacité de traitement limitée. Les dispositifs qui sont connectés à un réseau instable peuvent bénéficier du modèle publication/abonnement de MQTT qui conserve les messages sur le broker (Lampkin et al., 2012). Les messages pourront être lus durant la période de stabilité du réseau en se connectant au broker.

En revanche, le MQTT présente certains inconvénients. Il n'est pas conçu pour transférer de grandes quantités de données, et peut donc ne pas convenir aux applications qui nécessitent des flux de données à large bande passante. L'expérience menée par l'équipe de Calabretta et al. a montré qu'il est possible de surcharger le MQTT en imposant l'authentification « nom d'utilisateur / mot de passe » pour la lecture de différents messages publiés (Calabretta et al., 2018). Un autre inconvénient majeur de MQTT est le manque d'ordonnancement dans les messages. À cause du routage des paquets de données à travers l'Internet, les messages n'arrivent pas toujours dans l'ordre chez le broker. Conséquemment, les messages lus par un client MQTT ne sont pas nécessairement ordonnés.

Néanmoins, le MQTT est un protocole de messagerie fiable et couramment utilisé. Sa simplicité, son efficacité et sa prise en charge des réseaux non fiables le rendent bien adapté à une utilisation dans un large éventail d'applications IIoT.

1.3.2 Plateforme IIoT-ThingSpeak

Les données générées par des dispositifs connectés de l'IIoT doivent être entreposées afin de permettre leur exploitation et leur valorisation. Dans cette partie de la revue de la littérature, nous accorderons une attention particulière à la plateforme infonuagique ThingSpeak (Maureira et al., 2011). Cette dernière offre des services infonuagiques pour agréger, visualiser et analyser des flux de données.

D'un point de vue conceptuel, la plateforme ThingSpeak est organisée sous forme d'un ensemble de canaux de données. Chaque canal possède une configuration qui décrit le contenu des champs formant le canal et les paramètres associés. Cette organisation conceptuelle est montrée dans la Figure 1.2. Les données enregistrées dans les champs du canal sont affichées sous forme de « vues » qui sont en quelque sorte des tableaux de bord simplistes. Un point fort de ces vues est sa capacité de rendu en temps réel. Chacune des données reçues par ThingSpeak est automatiquement affichée dans une vue donnant ainsi une l'allure générale du flux de données des champs.

On note que les données de chacun des champs sont disponibles pour le traitement par le logiciel de calcul MATLAB qui est imbriqué directement dans cette plateforme infonuagique. Ainsi, on peut y effectuer des analyses statistiques et graphiques sans la nécessité de déplacer les données dans un autre environnement infonuagique. Enfin, l'accès aux champs d'un canal est contrôlé par des clés : une pour l'écriture et une autre pour la lecture. Ainsi, il est possible de confier les tâches d'écriture et de lecteur à différents dispositifs du système IIoT.

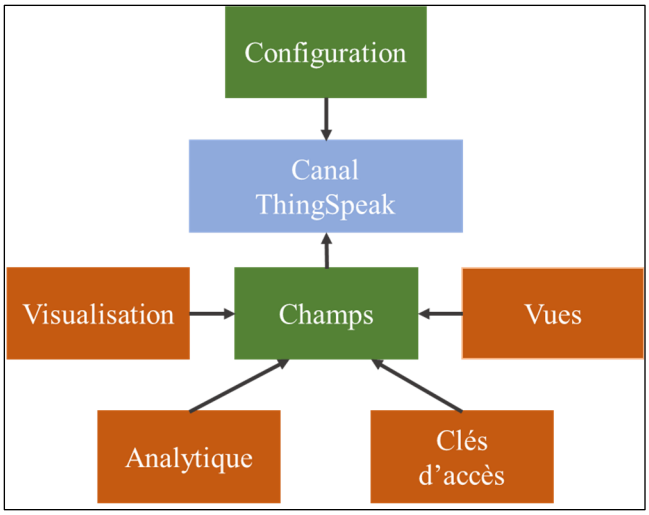

Figure 1.3 Organisation conceptuelle d'un canal de ThingSpeak

ThingSpeak offre le protocole MQTT pour le transfert et la lecture des données d'un canal. C'est ainsi que l'équipe de Nettikadan a pu concevoir leur système de surveillance de conditions environnementales et de sécurité des habitations (Nettikadan & Raj, 2018). Dans ce projet, des données environnementales sont envoyées dans un canal de ThingSpeak au moyen du protocole MQTT. Un script réactif du type « If This Then That » analyse en temps réel le flux de données du canal et réalise trois ensembles d'actions possibles : 1) démarrer/arrêter le ventilateur, 2) allumer/éteindre une lumière, 3) activer/désactiver une alarme. Ce projet IIoT est représentatif des applications modernes du protocole MQTT et des plateformes infonuagiques dédiées à des dispositifs connectés.

1.4 Conclusion

Dans ce chapitre, nous avons présenté une revue de la littérature portant sur les méthodologies de conception d'une l'interface utilisateur. Cette étude est nécessaire pour comprendre les concepts et les tendances modernes de cette discipline du design. On constate en effet que l'interface utilisateur est beaucoup plus qu'un simple programme informatique. Elle est la porte d'entrée vers les fonctionnalités d'une application logicielle et est l'élément principal de

l'expérience utilisateur. C'est aussi à travers l'interface utilisateur que l'on porte un jugement sur l'utilisabilité de l'application logicielle en développement. Bien entendu, une mauvaise conception de l'application logicielle ne peut être camouflée par une belle interface utilisateur. En revanche, une mauvaise conception de l'interface utilisateur peut très bien nullifier l'utilisabilité d'un bon logiciel.

La revue de la littérature s'est poursuivie en examinant le protocole de communication MQTT et la plateforme infonuagique ThingSpeak. Le protocole MQTT est basé sur le modèle publication et abonnement. Il est très utilisé dans des applications où la consommation d'énergie est une contrainte majeure. Cette tendance est justifiée par le fait que le protocole : 1) utilise moins de bits par paquet de données à transférer; 2) utilise un seul paquet de données pour maintenir l'état d'une connexion au broker. En comparaison, le HTTP doit ouvrir et fermer la connexion à chaque transfert de données. Ce qui engendre un taux d'utilisation du processeur plus élevé. Le ThingSpeak est une plateforme infonuagique dédiée aux objets connectés et possède une fonctionnalité similaire à la plupart des plateformes IIoT. L'avantage principal de ThingSpeak est son intégration directe avec les outils d'analyse très étendus de MATLAB.

CHAPITRE 2

CONCEPTION DE L'OUTIL DE GESTION

2.1 Introduction

Le système d'acquisition de données conçu pour ce projet est illustré dans la Figure 2.1. Il est composé de $M > 1$ coordonnateurs Raspberry Pi reliés à $N > 1$ nœuds de contrôle Arduino par la communication sans fil *Long Range* (LoRa). Ce sont ces nœuds de contrôle qui effectuent la lecture des puissances transférées aux véhicules électriques par le biais de wattmètres connectés aux contrôleurs à courant continu (DCC) des bornes de recharge. Dans ce système, le coordonnateur joue aussi le rôle d'une passerelle de communication puisque les nœuds de contrôle sont isolés de l'Internet. C'est le coordonnateur qui est responsable d'entreposer les données reçues des nœuds de contrôle sur la plateforme infonuagique ThingSpeak.

Le design du système d'acquisition de données permet l'échantillonnage périodique des puissances transférées, et ce d'une façon entièrement autonome. Pour connaître et contrôler l'état de fonctionnement du système, il a été décidé d'introduire un outil de gestion capable de communiquer avec le coordonnateur à distance. La partie pointillée de la Figure 2.1 montre l'emplacement logique de cet outil à concevoir.

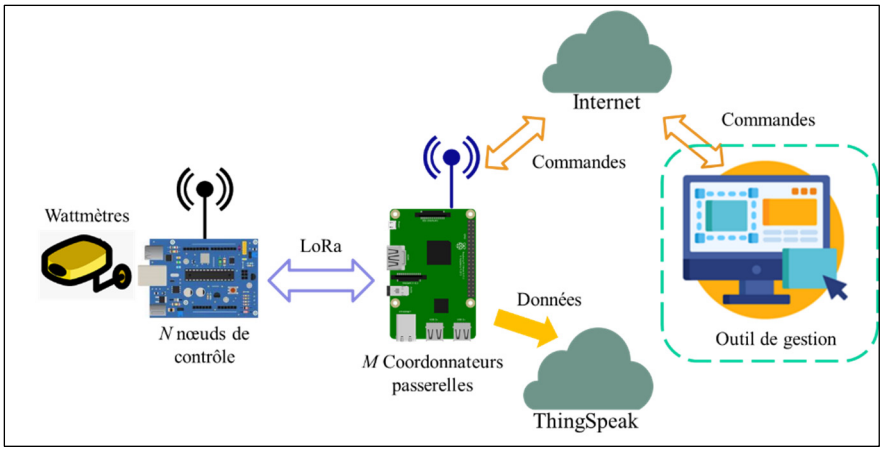

Figure 2.1 Schéma descriptif du système d'acquisition de données

L'état de fonctionnement du système d'acquisition comprend deux volets : 1) l'état du coordonnateur, 2) l'état des nœuds de contrôle. Or, les nœuds de contrôle ne sont pas accessibles par l'Internet (voir les justifications présentées dans la section 0.x). C'est la raison pour laquelle l'outil de gestion doit interroger le coordonnateur pour connaître son état et l'état des nœuds de contrôle. Pour accomplir cette tâche de gestion, il est convenu d'utiliser le schème « commande / réponse » dans la communication entre l'outil de gestion et le coordonnateur. Son principe de fonctionne est simple : lorsque l'outil de gestion envoie une commande au coordonnateur, celui-ci décode la commande et s'il y a lieu transmet la demande au nœud destinataire. Pour une commande destinée au coordonnateur, ce dernier l'exécute puis retourne la réponse à l'outil de gestion. Pour une commande destinée à un nœud de contrôle, le coordonnateur la redirige vers le nœud destinataire. Le nœud reçoit la commande, l'exécute et retourne le résultat au coordonnateur qui le redirige vers l'outil de gestion. Ce schème de traitement des commandes constitue l'une des capacités majeures de l'outil de gestion à concevoir. Une autre capacité majeure à prévoir est de l'application de ce schème de communication à $M > 1$ coordonnateurs. En effet, on doit pouvoir transmettre des commandes à différents coordonnateurs et de recevoir leurs réponses. Ces coordonnateurs peuvent être installés dans des bâtiments distincts et possiblement géographiquement éloignés les uns des autres. Dans ce cas, la gestion des coordonnateurs doit se faire au sein du même outil.

2.2 Conception de l'outil de gestion

Nous nous sommes basés sur la méthodologie UCD, présentée dans la section 1.2.3, pour concevoir l'outil de gestion des coordonnateurs. Dans un premier temps, nous avons analysé les besoins du système d'acquisition de données. Voici les constatations résultant de cette analyse :

a. Communication asynchrone

La communication entre l'outil de gestion et les coordonnateurs doivent se faire d'une façon asynchrone. Chaque coordonnateur du système d'acquisition fonctionne indépendamment et est capable de recevoir les puissances lues par les nœuds de contrôle à des intervalles réguliers. Les commandes envoyées aux coordonnateurs ne sont donc pas au pas avec la période d'échantillonnage des nœuds de contrôle.

b. Commandes distinctes destinées aux coordonnateurs et aux nœuds

Les commandes pour les coordonnateurs et pour les nœuds ne sont pas de même nature. Celles pour les coordonnateurs concernent leur état de fonctionnement et l'ensemble des nœuds. Les commandes destinées aux nœuds concernent le fonctionnement individuel des nœuds de contrôle. Le Tableau 2.1 donne la liste des commandes à traiter dans ce projet.

Tableau 2.1 Liste de commandes

Commande	Description
Pour les nœuds de contrôle	
PING	Commande venant du coordonnateur pour tester la présence d'un nœud.
Get Restart Count	Retourne la valeur du compteur de démarrage d'un nœud.
Set Restart Count	Régler la valeur du compteur de démarrage d'un nœud.
Sanity check	Faire une analyse de l'état de fonctionnement d'un nœud.
Pour les coordonnateurs	

Get Non Response Count	Retourner la valeur du compteur de « non-réponse » d'un nœud. Une commande qui n'a pas été exécutée par un nœud est considérée comme une « non-réponse ».
Get Sampling Time	Lire la période d'échantillonnage. Cette période de temps correspondant à la lecture des nœuds de contrôle par la commande READ.
Set Sampling Time	Régler la période d'échantillonnage. Cette période de temps correspondant à la lecture des nœuds de contrôle par la commande READ.
Pause Sampling	Suspendre l'échantillonnage des données.
Resume Sampling	Reprendre l'échantillonnage des données.
Get Sampling State	Lire l'état de l'échantillonnage.

On doit donc prévoir dans l'outil un traitement distinct de ces deux ensembles de commandes. Pour les commandes destinées aux nœuds de contrôle, il faut adjoindre aux commandes un identifiant ou une adresse permettant d'identifier le nœud destinataire. De même pour les commandes destinées aux coordonnateurs, il faut adjoindre aux commandes un identifiant ou une adresse permettant d'identifier le coordonnateur destinataire.

c. Commandes provoquant une réponse de différents types

L'analyse des besoins a révélé que les réponses à recevoir par l'outil varient selon les commandes envoyées. Le Tableau 2.2 donne la liste de réponses possibles. On peut observer, dans ce tableau, qu'il existe trois (3) types de réponse : 1) un code représentant un accusé de réception, 2) une valeur entière positive, 3) un indicateur de l'état de l'échantillonnage.

En pratique, ces trois types de réponses peuvent être encodés par des valeurs entières $v \geq 0$. Il suffit de choisir convenablement :

- Une valeur entière représentant l'accusé de réception ACK;
- Une valeur entière pour représenter « en marche » et une autre valeur pour représenter « en pause ».

Donc, une variable de type entier positif suffit pour entreposer la réponse reçue par l'outil de gestion.

Tableau 2.2 Réponse correspondante à chacune des commandes

Commande	Réponse
PING	Le code ACK (*acknowledge*).
Get Restart Count	Une valeur entière entre 0 à 100 000.
Set Restart Count	Le code ACK (*acknowledge*).
Sanity check	Un indicateur de l'état de l'échantillonnage : succès ou échec.
Get Non Response Count	Une valeur entière entre 0 à 100 000. Cette valeur est enregistrée dans la mémoire EEPROM des nœuds de contrôle. La valeur limite 100 000 est sélectionnée, car elle représente approximativement la durée de vie en écriture de l'EEPROM.
Get Sampling Time	Une valeur entière entre 60 à 3600. Cette valeur est en secondes. On veut limiter la période d'échantillonnage entre 1 minute à 1heure.
Set Sampling Time	Le code ACK (*acknowledge*).
Pause Sampling	Le code ACK (*acknowledge*).
Resume Sampling	Le code ACK (*acknowledge*).
Get Sampling State	Un indicateur de l'état de l'échantillonnage : en marche ou en pause.

d. Format de communication entre l'outil de gestion et les coordonnateurs

Il s'agit d'une communication machine à machine et l'utilisation d'un format de données standardisé est de mise. Dans le domaine de l'IIoT, les schémas XML (*Extensible Markup Language*), XHTML (*Extensible HyperText Markup Language*) et JSON (*Javascript Object Notation*) sont couramment utilisés pour la communication entre les objets en réseau. Ces schémas sont ouverts et standardisés : W3C recommandations pour XML et XHTML, ECMA-404 pour JSON. Le Tableau 2.3 est un résumé de type des paramètres des commandes à implanter dans l'outil de gestion de ce projet de recherche.

Tableau 2.3 Type des paramètres des commandes

Commande	Type d'entrée	Type de sortie

PING	Aucun	Code ACK
Get Restart Count	Aucun	Valeur entière
Set Restart Count	Valeur entière. Par défaut 0	Code ACK
Sanity check	Aucun	Valeur entière
Get Non Response Count	Aucun	Valeur entière
Get Sampling Time	Aucun	Valeur entière
Set Sampling Time	Valeur entière. Par défaut 60	Code ACK
Pause Sampling	Aucun	Code ACK
Resume Sampling	Aucun	Code ACK
Get Sampling State	Aucun	Valeur entière

Clairement, le format à adopter doit pouvoir structurer les commandes, les paramètres, les réponses et d'autres données nécessaires à une communication entre l'outil de gestion et les coordonnateurs du système d'acquisition.

2.2.1 Organisation proposée

L'organisation de l'outil de gestion retenue est celle basée sur le modèle principal / secondaire. Ce modèle est montré dans la Figure 2.2 où l'outil de gestion est identifié par la désignation GUI (*Graphical User Interface*) afin de simplifier l'écriture et la compréhension. Dorénavant, le terme GUI sera synonyme de notre outil de gestion à concevoir et à implanter.

Dans la Figure 2.2, le GUI joue le rôle du principal en communication avec trois coordonnateurs. Chaque coordonnateur joue également le rôle du principal vis-à-vis les nœuds. Le code numérique assigné aux coordonnateurs représente leur identifiant ou adresse. Il en de même pour les nœuds.

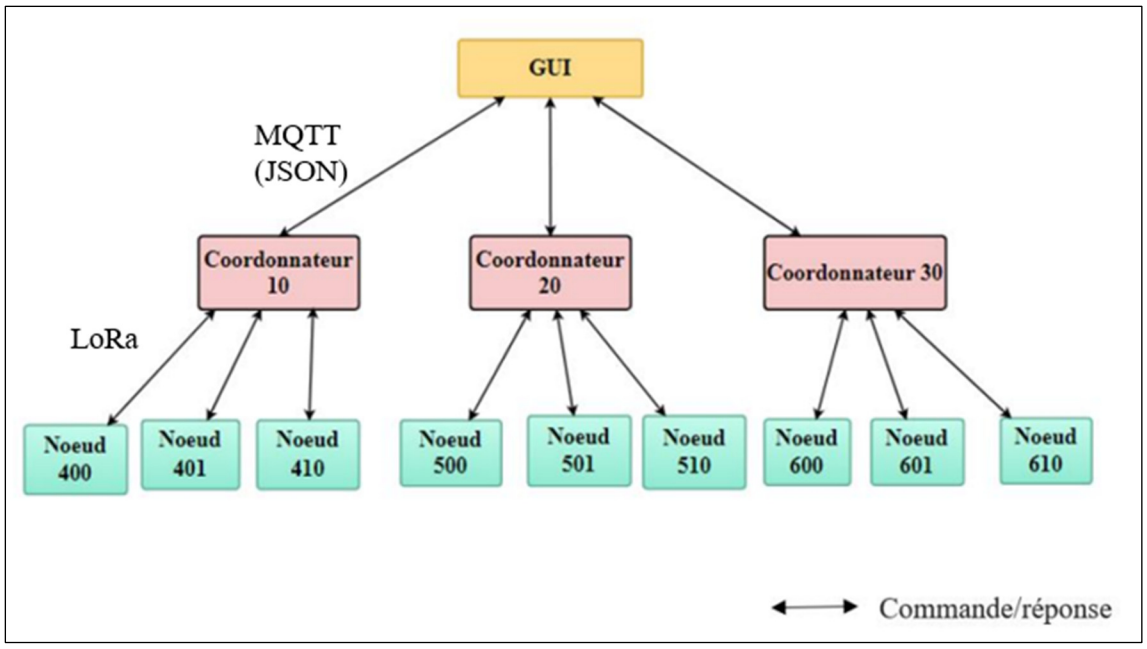

Figure 2.2 Organisation de l'outil de gestion

2.2.1.1 Justification

Le modèle principal / secondaire est convenable pour représenter l'aspect asymétrique de la communication entre le GUI et les coordonnateurs. En effet, le GUI est toujours l'initiateur de la communication puisqu'il est responsable de transmettre les commandes aux coordonnateurs. Les coordonnateurs, quant à eux, ne peuvent contacter directement le GUI. Ce dernier est activé par l'intervention directe de l'opérateur. Cette asymétrie est une mesure importante pour empêcher les erreurs d'opération en l'absence d'une personne opérant l'outil de gestion.

D'un point de vue général, ce modèle est le reflet du principe de fonctionnement du système d'acquisition des données. Les puissances transférées aux véhicules électriques sont lues par les nœuds de contrôle sous la demande des coordonnateurs. C'est pourquoi l'organisation proposée pour l'outil de gestion est similaire à celle du système d'acquisition des données.

2.2.2 Choix des composants du GUI

Dans la conception de l'outil de gestion, les croquis jouent un rôle crucial dans le processus de conception. Ils facilitent l'expérimentation rapide des idées distinguées, l'exploration des différents modèles et d'apporter des ajustements au design en éliminant le besoin d'investir en termes d'efforts et de temps. Ils permettent de rétrécir le nombre d'itérations dans le processus puisque les développeurs ont la possibilité de modifier et d'améliorer la conception avant qu'elle ne soit entièrement développée.

C'est ainsi que nous avons préparé des croquis qui présentent des modèles différents d'utilisation de l'outil de gestion. Une fois un croquis est créé, il est montré aux opérateurs pour avoir leurs avis et savoir l'outil convient aux besoins. La Figure 2.3 présente quelques exemples de croquis développés durant la phase de conception. Le croquis (a) a été le premier réalisé. La sélection du coordonnateur, de la commande et du nœud s'effectue par des listes déroulantes contenant les identifiants et les commandes possibles. Des champs de texte sont disponibles pour saisir les paramètres correspondants à la commande sélectionnée.

De même, nous avons choisi de placer les listes déroulantes et les champs de saisie à droite par rapport à leurs étiquettes (les titres), car les chiffres sont généralement alignés à droite puisque cela les rend plus faciles à lire. Il est plus confortable de lire les chiffres à droite, car on compte toujours les positions des chiffres de droite à gauche.

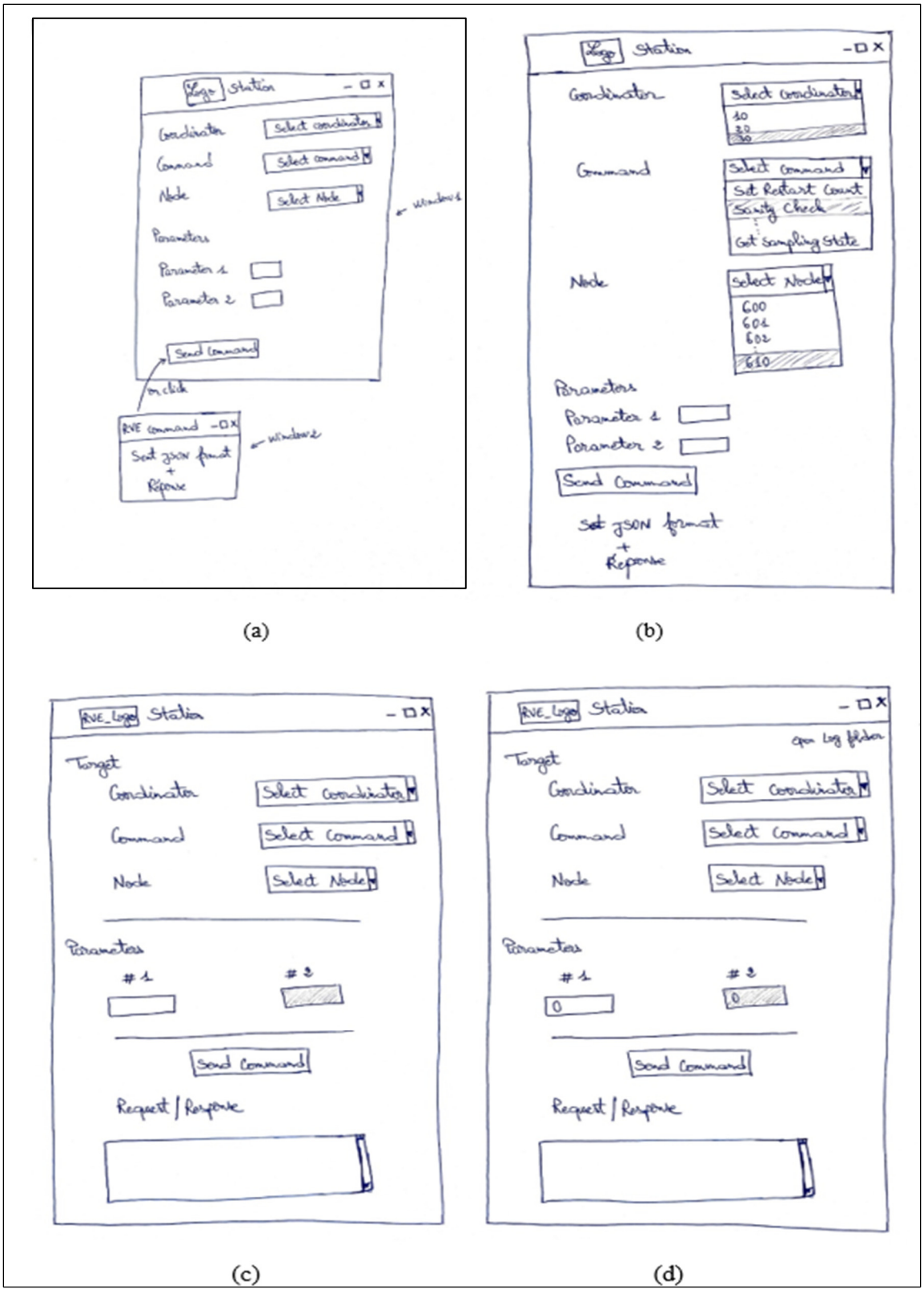
Logo Station
Coordinator
Command
Node
Select coordinator
Select command
Select Node
Parameters
Parameter 1
Parameter 2
Send Command
window 1
on click
RVE Command
Set json format
+
Reponse
window 2
Logo Station
Coordinator
Select Coordinator
10
20
30
Command
Select Command
Set Restart Count
Sanity Check
Get Sampling State
Node
Select Node
600
601
602
610
Parameters
Parameter 1
Parameter 2
Send Command
Set json format
+
Reponse
(a)
(b)
RVE_Logs Station
Target
Coordinator
Select Coordinator
Command
Select Command
Node
Select Node
Parameters
1
2
Send Command
Request / Response
RVE_Logs Station
open Log folder
Target
Coordinator
Select Coordinator
Command
Select Command
Node
Select Node
Parameters
1
2
0
0
Send Command
Request / Response
(c)
(d)

Figure 2.3 Exemples de croquis développés pendant la réalisation du projet

Dans cette première idéation, un clic sur le bouton « Send Command » déclenche l'envoi de la commande et l'ouverture d'une deuxième fenêtre. C'est dans cette seconde fenêtre qui affichera la commande envoyée et la réponse reçue du coordonnateur. L'inconvénient de cette première solution était la présentation de deux fenêtres. Le problème consiste à obliger l'utilisateur à fermer la deuxième fenêtre afin de lancer une nouvelle commande. Cette première idéation ne donne pas une bonne utilisabilité de l'application logicielle.

Nous avons donc remis la zone d'affichage dans l'application comme montré dans le croquis (b). Les croquis qui suivent (b) comportent des modifications qui concernent l'affichage des éléments de l'interface graphique et leurs dispositions. Dans le croquis (c), la zone d'affichage est plus grande pour qu'elle puisse afficher plus que 10 lignes et le champ `param2` est grisé pour faire comprendre à l'opérateur qu'il ne peut pas l'utiliser. De plus, nous avons placé le bouton et la zone d'affichage au centre pour une meilleure visibilité et pour donner une mise en page équilibrée avec un format symétrique.

2.2.2.1 Justification

Nous avons choisi de placer les listes déroulantes et les titres qui les correspondent verticalement au même niveau les uns par rapport aux autres pour un aspect général calme et équilibré. De plus, cette disposition physique évite la confusion dans la sélection des paramètres. Cependant, les champs `param1` et `param2` sont placés horizontalement pour gagner de l'espace dans la fenêtre et puisque, pour le moment, `param2` ne sera pas utilisé et il n'y aura pas de confusion qui pourra engendrer pour l'opérateur de l'outil.

Enfin, le croquis (d) représente la dernière version de l'interface graphique. Dans ce modèle, nous avons ajouté quelques détails comme l'initialisation des paramètres `param1` et `param2` à 0 et la possibilité d'ouvrir un dossier qui contient des fichiers horodatés (*log files*) contenant la trace des commandes envoyées et les réponses reçues des coordonnateurs.

2.2.3 Modèle d'utilisation

Afin de bien organiser les idées collectées dans les étapes précédentes, un diagramme de séquence illustrant la fonction de l'outil de gestion a été créé. La Figure 2.4 résume le fonctionnement du GUI et la façon dont elle interagit avec le coordonnateur.

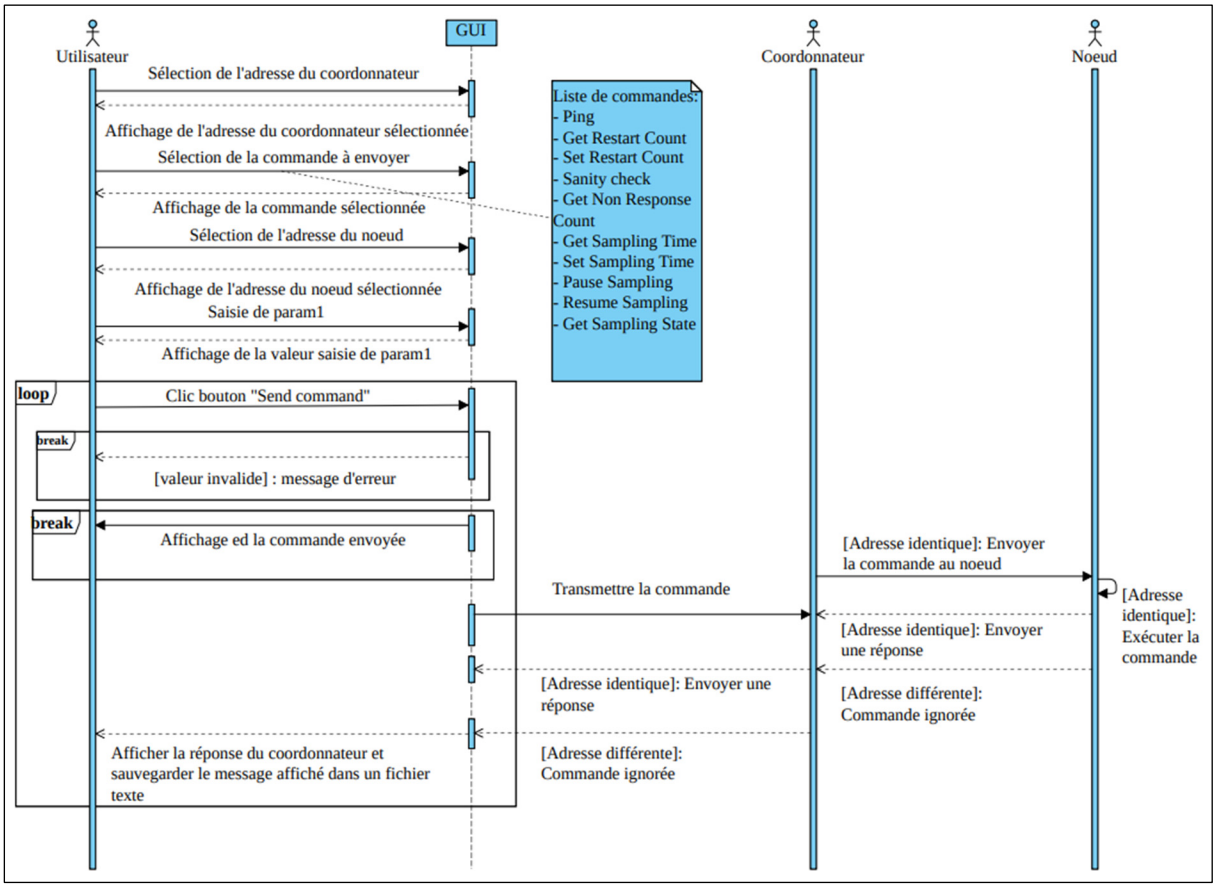

Figure 2.4 Diagramme de séquence sur le fonctionnement de l'interface graphique

L'opérateur devra choisir l'identifiant ou l'adresse du coordonnateur qu'il désire envoyer une commande. En suivant l'organisation proposée de la Figure 2.2, l'opérateur pourrait choisir de la liste déroulante les coordonnateurs à l'adresse 10 ou 20 ou 30. Par la suite, il sera en mesure de sélectionner la commande à envoyer d'une deuxième liste déroulante. Le même processus

sera fait pour la sélection de l'adresse d'un nœud destinataire et s'il y a lieu l'opérateur devra saisir la valeur du paramètre de la commande dans le champ `param1`. Finalement, l'opérateur doit sélectionner le bouton « Send command » pour transmettre la commande, le paramètre correspondant et l'identification du nœud de contrôle au coordonnateur destinataire.

2.2.3.1 Justification

Ce modèle d'utilisation a deux avantages. D'abord, l'opérateur ne peut accidentellement entrer une adresse invalide pour le coordonnateur et /ou le nœud de contrôle destinataire. Les listes déroulantes l'empêcheront de commettre cette erreur. Ensuite, ce modèle d'utilisation est conforme au principe de fonctionnement du système d'acquisition présenté dans la section 2.1 et à la constatation b) de la section 2.2. Selon le principe de fonctionnement énoncé, une adresse valide est nécessaire pour que l'exécution des étapes de la communication. Si la commande est destinée à un nœud de contrôle, le coordonnateur la redirigera au nœud dentinaire si et seulement si l'adresse du nœud récipient est valide. Sinon, la commande sera ignorée.

2.2.4 Séquence d'activation des composants GUI

Dans le modèle d'utilisation, il n'y a pas un ordre spécifique pour la sélection des paramètres du message. Cependant, il y a de la dépendance entre les champs de GUI pour éliminer la possibilité d'envoyer une commande invalide. Cette dépendance est présentée dans la Figure 2.5. La liste déroulante pour la sélection des nœuds est initialement activée, mais elle se désactive quand la commande choisie est « Get Sampling Time », « Set Sampling Time », « Pause » ou « Resume ». Le champ de saisie de `param1` est toujours désactivé sauf dans le cas où la commande sélectionnée est « Set Sampling Time » ou « Set Restart Count ».

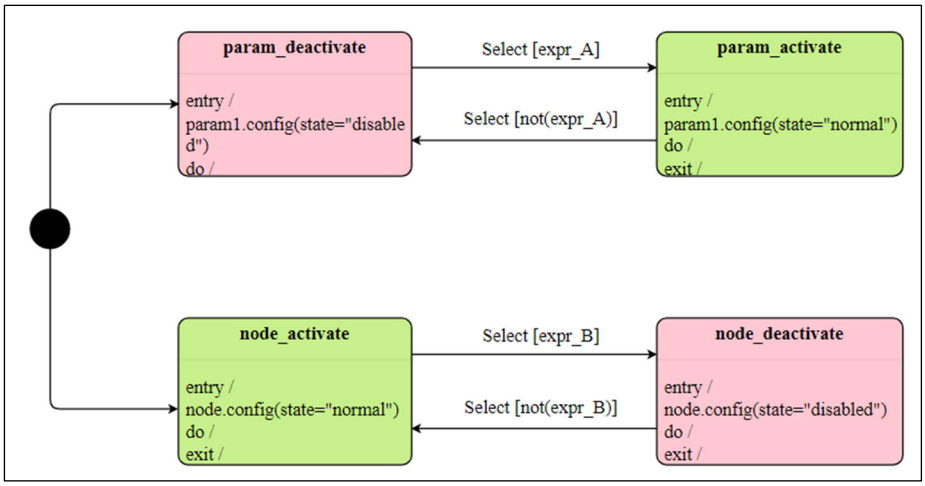

Figure 2.5 Diagramme d'état pour les champs responsables de la sélection des nœuds et la saisie de `param1`

Les expressions Expr_A et Expr_B de la Figure 2.5 sont respectivement :

$$\begin{aligned} \text{Expr}_A &= \text{Set Sampling Time} \vee \text{Set Restart Count} \\ \text{Expr}_B &= \text{Get Sampling Time} \vee \text{Set Sampling Time} \vee \text{Pause} \vee \text{Resume} \end{aligned} \tag{2.1}$$

Pour les valeurs entrées de `param1`, nous avons planifié un ajout de message d'erreur quand la valeur de `param1` n'est pas dans l'intervalle correcte. Ce message s'affiche après la sélection du bouton « Send commad ».

2.2.5 Implantation de l'outil de gestion

Pour la réalisation de l'outil de gestion, nous avons choisi le Python qui est un langage de programmation interprété de haut niveau. Python est puissant, facile à utiliser, et fournit les outils essentiels pour la conception d'une GUI. De plus, ce langage est devenu de plus en plus important et largement utilisé des grandes compagnies comme Google et NASA (Srinath, 2017). Le Tableau 2.4 répertorie brièvement les différentes bibliothèques nécessaires à l'implantation de l'outil de gestion.

D’abord, la bibliothèque tkinter offre plusieurs widgets pour le développement des GUI et est disponible par défaut dans les distributions de Python. Dans notre application, nous avons utilisé le widget appelé « combobox » qui est une combinaison d'une zone de sélection et d'un menu déroulant. Lorsque l'utilisateur clique sur la flèche, la liste déroulante apparaît. Si l'utilisateur clique sur l'un des éléments de la liste, ce choix remplace le contenu actuel de l'entrée. Les combobox sont utilisées pour la sélection du coordonnateur, de la commande et du nœud.

Tableau 2.4 Bibliothèques Python principalement utilisées

Nom de la bibliothèque	**Fonction**
tkinter	Tkinter est une interface standard orientée objet qui fournit les outils essentiels pour la conception d'interfaces graphiques (Srinath, 2017).
json	Constitue un format d'échange léger pour le stockage et le transport de données, inspiré de la syntaxe littérale des objets JavaScript. Les données JSON peuvent ainsi être utilisées par n'importe quel langage de programmation et facilement envoyées entre les machines.
paho-mqtt	C’est une bibliothèque qui permet d’implémenter le protocole de communication avec MQTT (Light, 2021).
time	Cette bibliothèque fournit de nombreuses façons de représenter le temps dans le code et fournit des fonctionnalités comme l'attente pendant l'exécution du code.
os	Ce module intégré possède un certain nombre de fonctions utiles en rapport avec le système d’exploitation qui peuvent être utilisées pour lister le contenu des répertoires, dater les fichiers, etc.
pyinstaller	Pyinstaller permet à l’utilisateur d’exécuter une application développée en Python sans avoir besoin d’installer l’interpréteur Python et ses bibliothèques en regroupant toutes les dépendances dans un seul paquet.

Nous avons également utilisé le widget `label` qui sert à afficher des textes ou des images. Dans cette application, le label a été utilisé pour ajouter les titres nécessaires pour chaque élément de l’interface pour que l’opérateur n’ait pas de confusion lors de l’utilisation de l’interface. Le widget `entry`, qui est le champ de saisie des valeurs, nous a été utile pour entrer

la valeur de `param1`. En revanche, cet élément ne nous permet pas d'empêcher la saisie des valeurs qui sont en dehors d'un intervalle spécifique. La validation des valeurs doit se faire dans le programme de l'outil de gestion. L'affichage du message envoyé et la réponse reçue par l'outil de gestion sont réalisés dans une zone de texte du widget `scrolledtext`. Ce dernier possède une barre de défilement vertical et on peut y faire afficher un bon nombre de caractères. Le résultat de l'implantation est illustré dans la Figure 2.6.

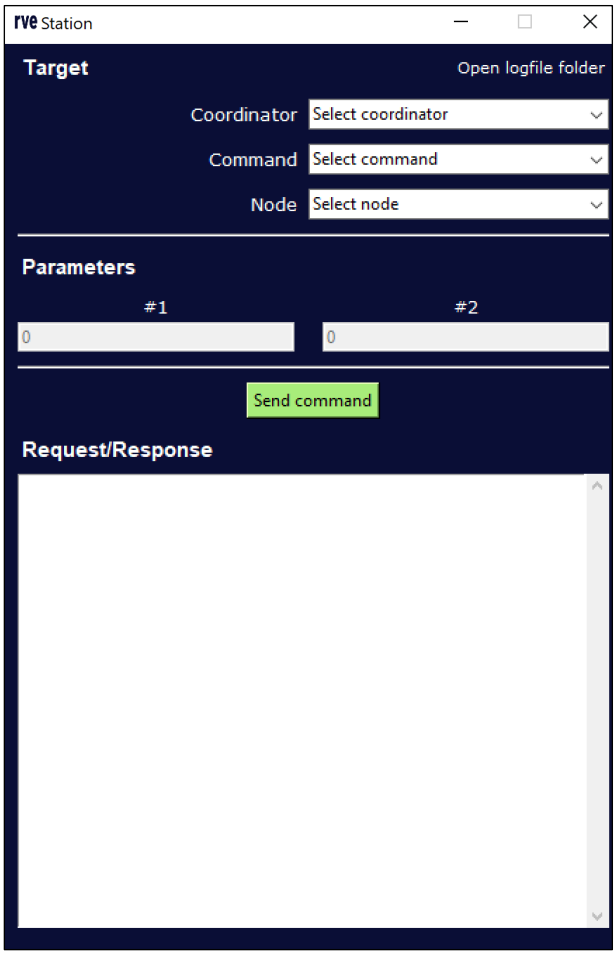

Figure 2.6 Interface utilisateur de l'outil de gestion

Afin de rendre l'installation de notre application facile le poste de l'utilisateur, nous avons utilisé la bibliothèque (Pyinstaller, 2023) qui nous a permis de convertir le programme en une

application bureau exécutable présentée avec une icône personnalisée. Pour ce faire, il suffit d'exécuter la ligne de commande suivante :

```
pyinstaller --onefile --icon=rve.ico rve_GUI.py
```

La Figure 2.7 montre l'application de bureau et le résultat de son exécution. On remarque que l'exécution de l'application de bureau produit une fenêtre de commande dans laquelle les messages de débogage sont affichés. On peut éliminer la console par l'option `--noconsole` de `pyinstaller`.

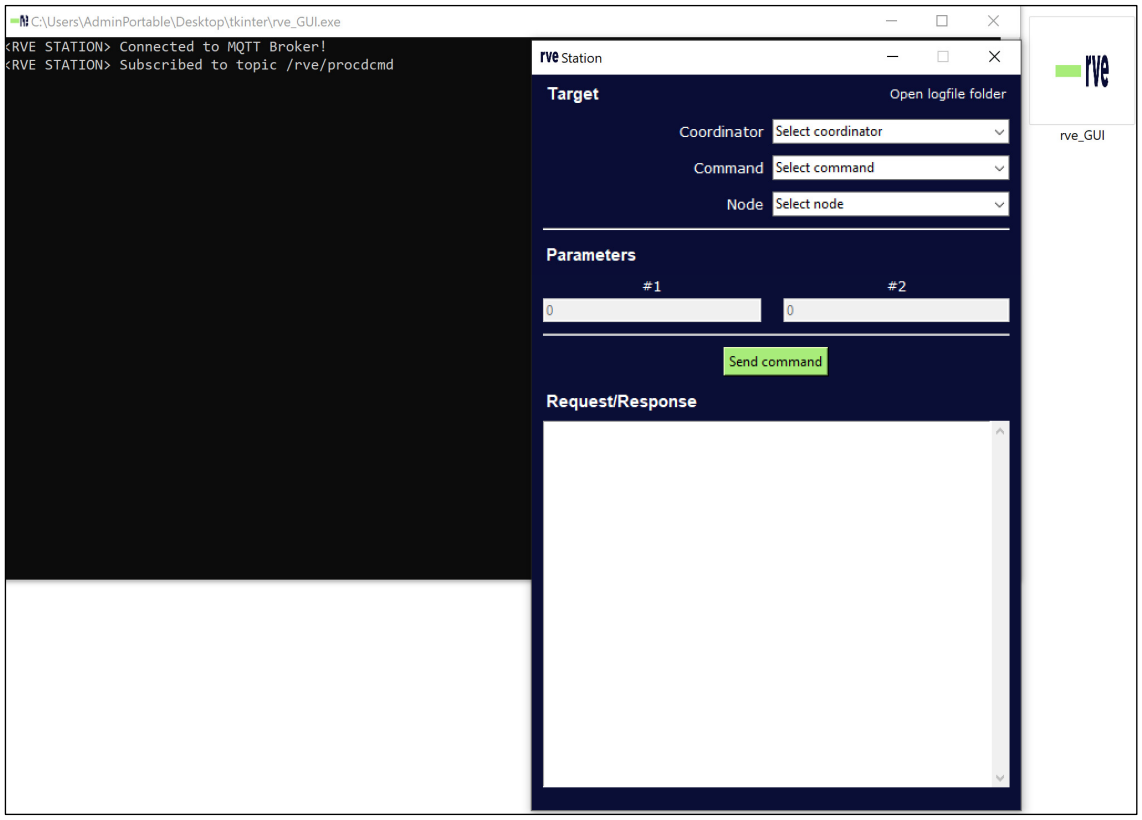

Figure 2.7 Application bureau pour l'exécution de l'outil de communication

2.3 Implantation de la communication MQTT

Le protocole MQTT est utilisé dans la communication entre l'outil de gestion et les coordonnateurs du système d'acquisition des données. L'implantation de ce protocole dans

l'outil de gestion est réalisée à l'aide de la bibliothèque `paho-mqtt` de la fondation Eclipse. Le choix de cette bibliothèque est basé sur la simplicité de son interface de programmation et sur l'engagement de la fonction Eclipse dans la promotion de la communauté open source. La classe client est la classe principale de ce module, car elle fournit toutes les fonctions nécessaires pour se connecter à un broker MQTT, publier des messages, s'abonner à des sujets et recevoir des messages. Un certain nombre de paramètres doivent être configurés. Ces paramètres, tels qu'ils sont nommés dans l'outil de gestion, sont énumérés dans le Tableau 2.5.

Tableau 2.5 Liste des paramètres à configurer pour réaliser la communication MQTT

Variable	Description
RVE_MQTT_CLIENT_NAME	Il s'agit du nom du client MQTT.
RVE_MQTT_BROKER	C'est l'adresse du broker utilisé qui est « broker.emqx.io ». Ce broker est gratuit avec plus de 100 millions de dispositifs connectés (EMQX, 2012/2023).
RVE_MQTT_PORT	Le port utilisé pour le transfert des données. Nous avons utilisé dans ce projet le port 1883.
RVE_MQTT_REVDCMD_TOPIC	C'est le sujet pour les commandes reçues « /rve/rvedcmd ». Il est utilisé lors de la publication.
RVE_MQTT_PROCDCMD_TOPIC	C'est le sujet pour les commandes exécutées « /rve/procdcmd ». Il est utilisé lors de l'abonnement pour pouvoir lire la réponse du coordonnateur.
RVE_MQTT_QOS	C'est la qualité de service. Il s'agit d'un accord entre les deux parties pour garantir la livraison du message. Nous avons choisi le niveau 2.
RVE_MQTT_CLIENT_ID	Ça sert à identifier le client (GUI) lors de la communication avec le broker. Pour ce faire, nous avons aléatoirement généré un entier qui varie entre 0 et 1000.
RVE_MQTT_PASSWORD	C'est le mot de passe associé au nom du client.

2.3.1 Connexion au broker

La tâche la plus importante pour établir la communication MQTT est la connexion au broker. Pour ce faire, la Figure 2.8 montre que cette connexion est réalisée. En premier lieu, on doit importer les modules externes tels que `paho-mqtt` et `json`. Par la suite, on crée l'objet MQTT qui agit comme un client auquel on accorde un identifiant. Par la suite, nous avons défini les paramètres d'authentification qui sont le nom d'utilisateur et le mot de passe. Dans l'étape suivante, nous avons configuré le port de communication « Internet Procotol » (IP) à 1883 et l'adresse « Uniform Resource Locator » (URL) du broker MQTT. Si la connexion au broker est échouée, un message d'erreur indiquant ceci est affiché dans le terminal.

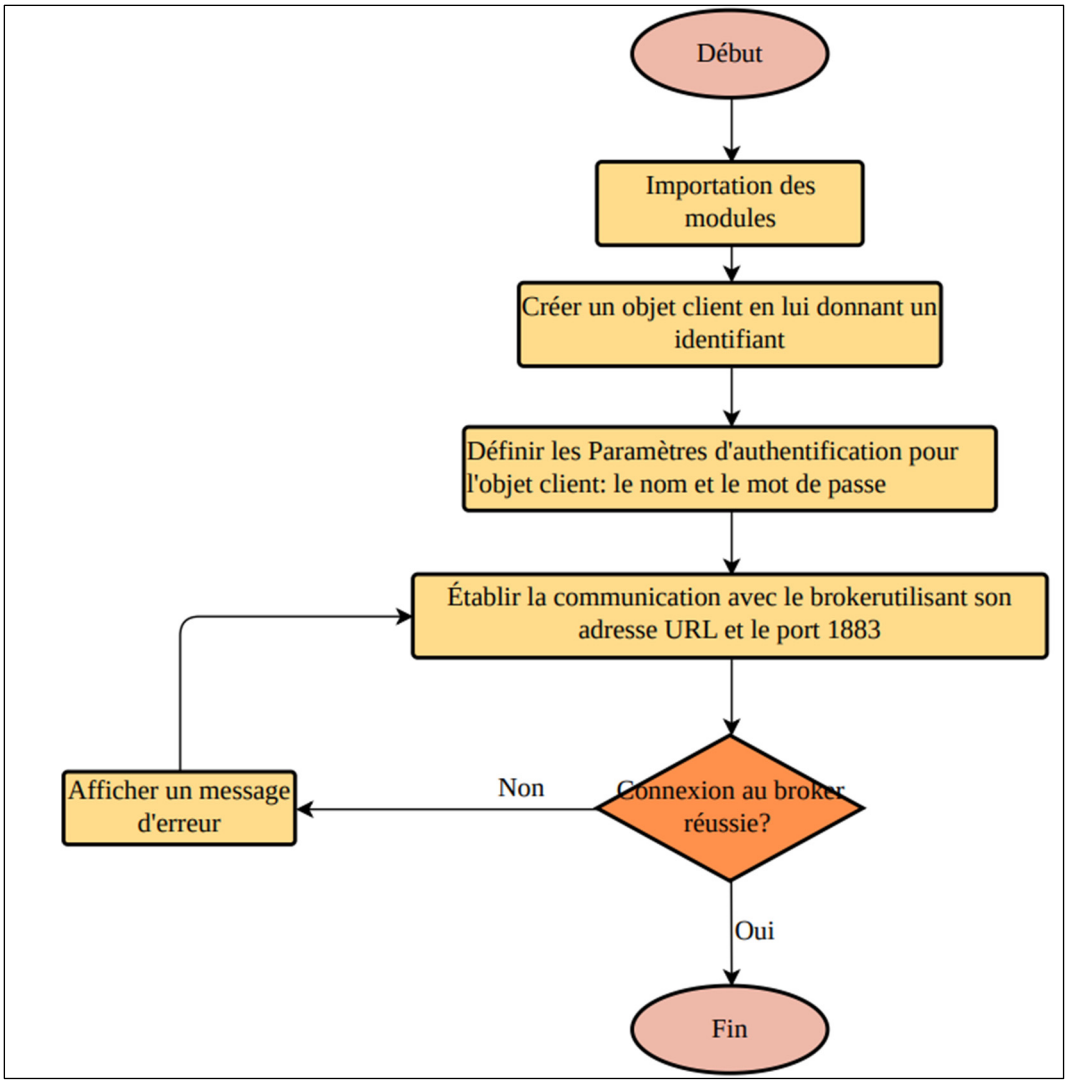

Figure 2.8 Connexion au broker MQTT

2.3.2 Publication

La Figure 2.9 explique comment la publication du message MQTT est exécutée. En effet, l'objet client se connecte au broker à l'adresse `RVE_MQTT_BROKER` sur le port portant le numéro `RVE_MQTT_PORT`. L'application prépare ensuite le message contenant la commande sélectionnée par l'utilisateur via l'interface graphique. Si la connexion est réussie, il publie la commande à envoyer vers le sujet `RVE_MQTT_REVDCMD_TOPIC`.

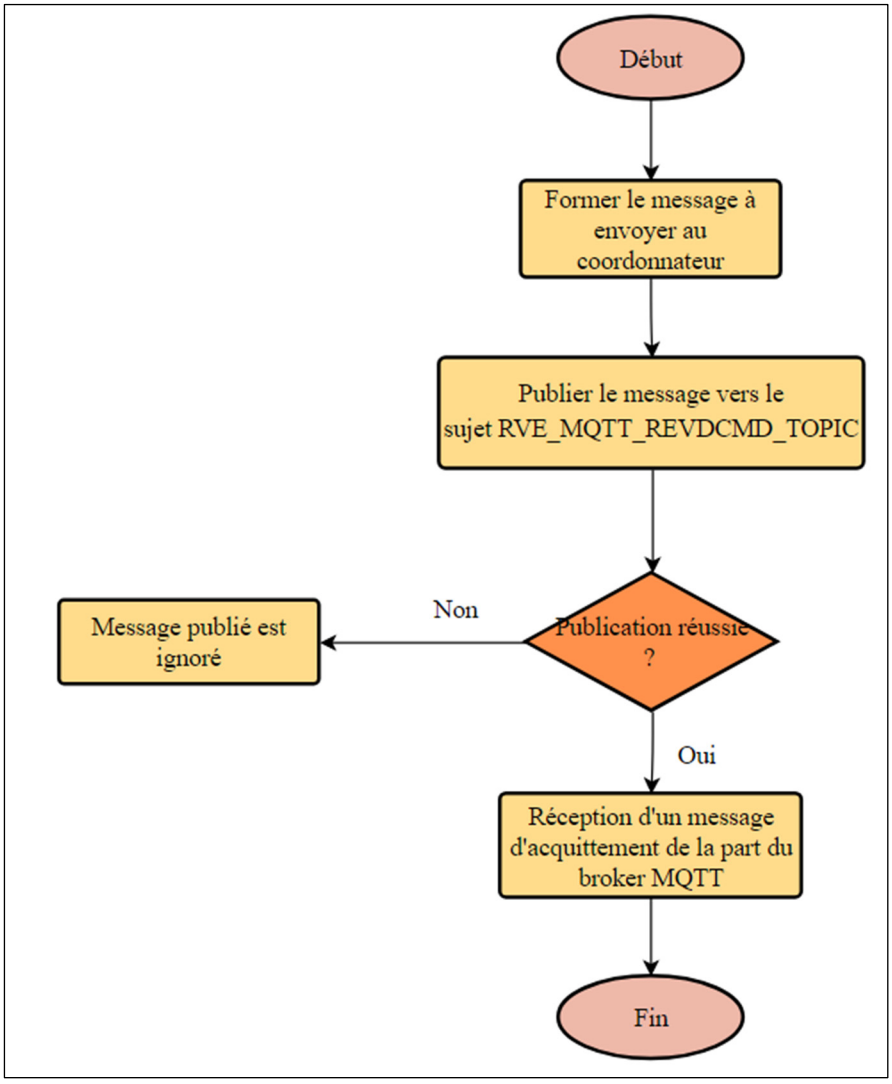

Figure 2.9 Publication MQTT

2.3.3 Abonnement

La Figure 2.10 explique comment l'outil de gestion reçoit la réponse du coordonnateur. Son objet client se connecte au broker MQTT, ensuite il s'abonne au sujet `RVE_MQTT_PROCDCMD_TOPIC` et regarde s'il y avait un message dans ce sujet. Si oui, il

traitera ce message et puis l'afficher à l'aide du widget `scrolledtext` de l'interface graphique.

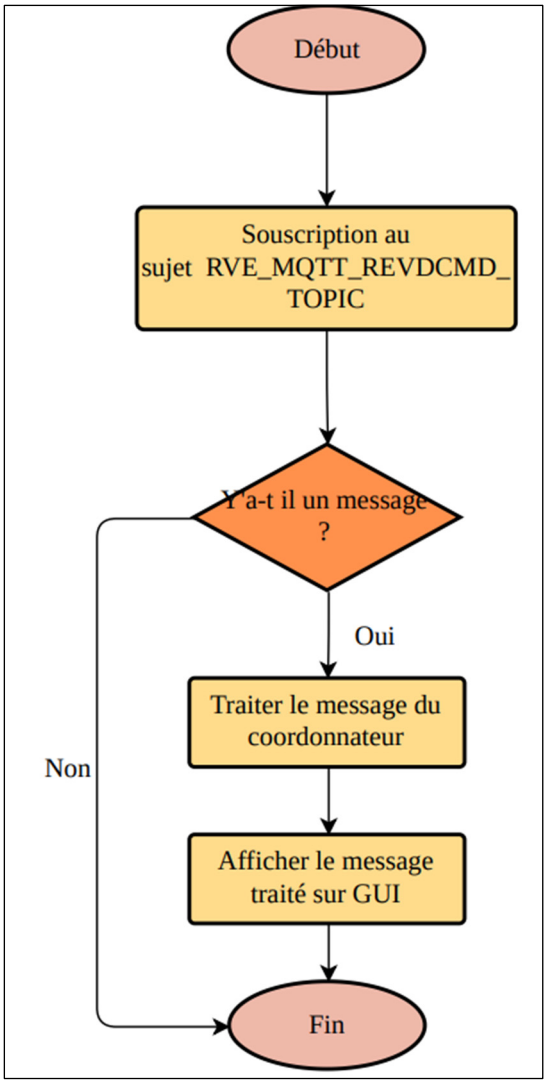

Figure 2.10 Abonnement et réception de messages MQTT

2.3.4 Communication avec les coordonnateurs

Les différentes actions de l'outil de gestion dans la communication avec un coordonnateur sont montrées dans le diagramme de séquence de la Figure 2.11.

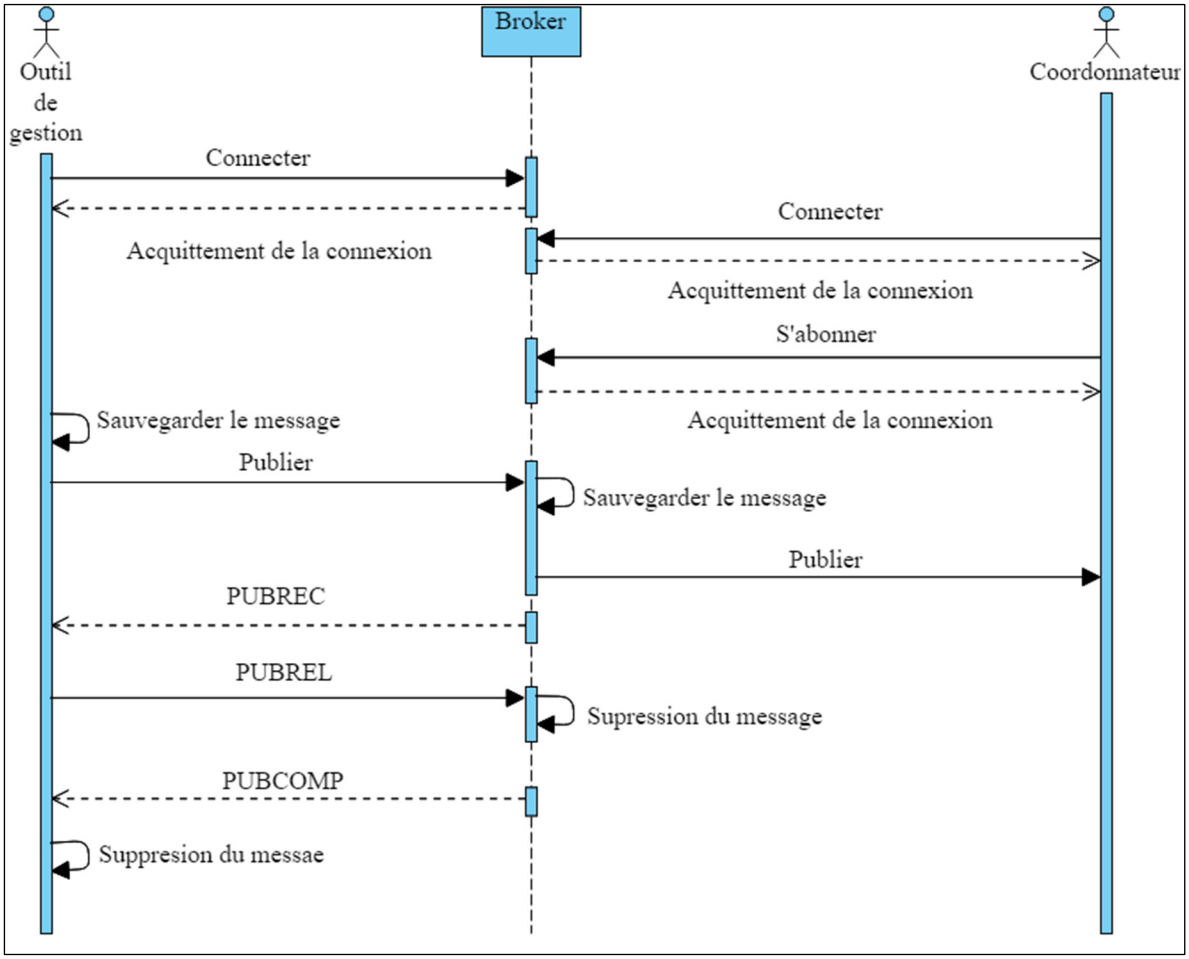

Figure 2.11 Diagramme de séquence montrant une communication MQTT entre l'outil de gestion et un coordonnateur

Au début, GUI demande une connexion au broker en s'authentifiant en utilisant son nom `RVE_MQTT_CLIENT_NAME` et le mot de passe associé `RVE_MQTT_PASSWORD`. Le coordonnateur demande également de se connecter au broker avec la même démarche.

Le broker renvoie un message d'acquittement pour confirmer l'établissement de la connexion. Par la suite, le coordonnateur s'abonne au sujet `RVE_MQTT_REVDCMD_TOPIC` avec un niveau de QoS 2, et le broker confirme l'abonnement avec un message d'acquittement. La QoS, abréviation de Qualité de Service, désigne la mesure de la capacité de transmission des données

en termes de débit, de perte de paquets et d'autres facteurs. En d'autres termes, elle évalue la qualité de service. En effet, nous avons choisi le niveau 2 de QoS car cela nous permettra d'assurer que les messages arriveront exactement une fois pour éviter la perte ou/et la duplication des messages ce qui n'est pas acceptable dans notre système. Par exemple, si la commande est perdue, le nœud ne va pas l'exécuter ou pire si elle est dupliquée, le nœud va l'exécuter plus qu'une fois ce qui va entraîner des erreurs au sein du système.

Un message `PUBREC` est la réponse à un message `PUBLISH` avec QoS niveau 2. Un message `PUBREC` est envoyé par le broker en réponse au message `PUBLISH`. Une fois que l'outil de gestion reçoit le message `PUBREC`, il supprime le message de publication initial. Ainsi, l'outil envoie un message `PUBREL` au broker pour confirmer la réception du message `PUBREC`.

Enfin, le broker envoie un message `PUBCOMP` avec l'identifiant du message initialement publié à l'outil de gestion lui indiquant que la livraison du message a réussi.

2.4 Conclusion

Dans ce chapitre, nous avons vu les étapes impliquées dans la conception de l'outil de gestion de notre projet en se basant sur la méthodologie UCD. Nous avons également pu la faire fonctionner avec le protocole MQTT en exploitant de diverses bibliothèques offertes par Python, principalement `paho-mqtt` qui prend en charge les fonctions de bases de notre protocole.

L'outil de gestion de la communication MQTT est désormais théoriquement prêt à l'emploi, il nous reste de voir en détail la conception de la partie destinataire dans le chapitre suivant et à lancer quelques tests pour observer si notre application répond conformément à nos attentes, ce qui sera couvert dans le dernier chapitre.

CHAPITRE 3

SYSTÈME DE COMMUNICATION DU COORDONNATEUR

3.1 Introduction

Le coordonnateur est une partie cruciale du système d'acquisition de données. Ce chapitre présentera de son système de communication. Nous allons ensuite montrer les différentes tâches qui peuvent être réalisées par le coordonnateur conçu et les différentes interactions nécessaires à l'accomplissement de ces tâches que soit l'interaction avec l'interface déjà présentée dans le chapitre 2, avec les nœuds de contrôle qui présentent les dispositifs IIoT ou avec la plateforme infonuagique. Nous allons également expliquer comment ces diverses communications ont été réalisées. Nous allons ainsi aborder l'aspect matériel pour étudier les composants qui sont utilisés dans la conception de la passerelle. De plus, nous allons traiter l'aspect logiciel pour étudier la façon dont les différentes fonctionnalités ont été implémentées dans le programme du coordonnateur et pour explorer comment les protocoles de communication utilisés ont été incorporés dans le code.

3.2 Système de communication

Le coordonnateur a la responsabilité de gérer les nœuds de contrôle dans le système d'acquisition des données. Le système de communication du coordonnateur comprend les protocoles MQTT et LoRa. Le protocole LoRa est utilisé dans la communication entre les coordonnateurs et ses nœuds de contrôles. Le protocole MQTT, quant à lui, est utilisé dans la communication entre les coordonnateurs, la plateforme infonuagique ThingSpeak et l'outil de gestion. Dans ce contexte, le coordonnateur joue aussi le rôle d'une passerelle de communication puisqu'il traduit les données de LoRa en MQTT et vice versa entre les réseaux locaux et l'Internet. Les différentes tâches de communication s'exécutant sur le coordonnateur sont illustrées dans le diagramme des cas d'utilisation de la Figure 3.1.

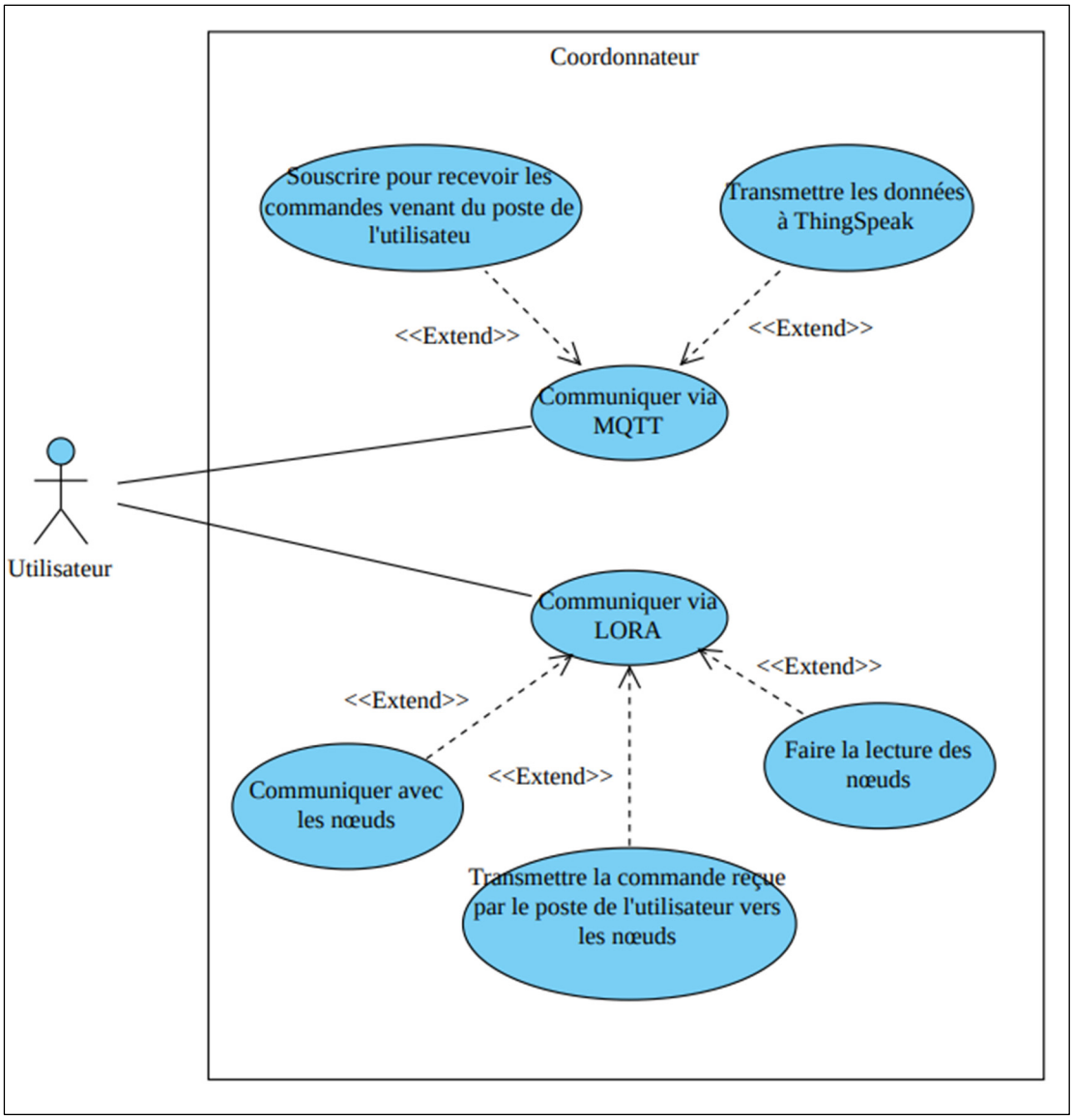

Figure 3.1 Diagramme des cas d'utilisation des tâches de communication du coordonnateur

Le coordonnateur utilise aussi la communication LoRa pour coordonner la lecture des nœuds de contrôle selon un temps d'échantillonnage constant. C'est en recevant la commande de lecture que les nœuds activent les capteurs reliés aux dispositifs DCC. Cette activation des capteurs permet au système d'acquisition des données de recevoir la valeur des puissances transférées à la borne de recharge des véhicules électriques. La conception de la communication via MQTT et LoRa sera présentée en détail dans les prochaines sections.

3.2.1 Conception de la communication coordonnateur-nœuds de contrôle

La communication coordonnateur-nœuds de contrôle s'effectue par la communication LoRa. Cette technologie fait partie d'une famille de procédés de communication par l'étalement de spectre (*spread spectrum*). La communication sans fil LoRa peut former des réseaux de communication de très grande distance pouvant atteindre jusqu'à 3 km en milieu urbain. De par sa longue portée et sa faible consommation d'énergie, la technologie LoRa est souvent déployée dans des applications IIoT (Zourmand et al., 2019).

Afin de mettre en œuvre cette communication, nous avons utilisé le module SX126X LoRa HAT de la compagnie Waveshare enfichable sur le coordonnateur Raspberry Pi (SX1262 915M LoRa HAT - Waveshare Wiki, 2023). La Figure 3.2 est une photo de ce module LoRa.

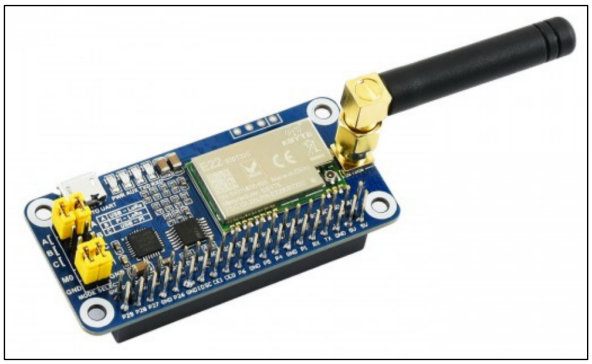

Figure 3.2 Module SX126X LoRa HAT
(SX1262 915M LoRa HAT -
Waveshare Wiki, 2023)

Ce module utilise le circuit E22 900T22S de EByte (E22-900T22S1B-, 2018) qui ajoute un convertisseur UART-USB et les broches GPIO pour rendre compatible son application sur un PC/Raspberry Pi/STM32. Ce module possède quatre modes d'opération présentés dans le Tableau 3.1 et qui sont: Normal/Transmission, WOR (*Wake Over Air*), Configuration et Hibernation (*Deep sleep*).

Tableau 3.1 Les modes d'opérations du circuit E22 900T22S

Mode	Fonctionnement
Normal/Transmission	Les transmissions sans fil et UART sont activées pour recevoir des données et les envoyer.
WOR (*Wake Over Air*)	Dans ce mode l'utilisateur doit ajouter des codes de réveil avant de transmettre les données. La réception est identique au Mode Normal.
Configuration	Ce mode permet de régler les divers paramètres de E22 900T22S.
Hibernation (*Deep sleep*)	Ce mode désactive la transmission et la réception du module et le met en mode veille profonde.

Dans ce projet de recherche, nous nous sommes intéressés aux modes Normal/Transmission et Configuration. Le mode Configuration sert à régler les paramètres de fonctionnement du circuit E22 900T22S tandis que le mode Normal/Transmission permet la transmission et la réception des paquets de données via la communication sans fil. Le module SX126X LoRa offre dans le mode Normal/Transmission, le mode d'opération appelé « Fixed Transmission ». Ce mode d'opération sera expliqué dans la sous-section suivante.

3.2.1.1 Mode d'opération Normal/Transmission

Lors de l'intégration du module SX126X LoRa, il est important de positionner ses cavaliers de manière appropriée (montrés en jaune dans la Figure 3.2). C'est par ces cavaliers que l'on règle les modes de fonctionnement du module. Pour activer le mode Normal/transmission, on doit désactiver M0 et M1. Pour définir la plateforme de contrôle de la carte SX126X LoRa Hat, on doit mettre les cavaliers à la position B qui est désignée pour la configuration de la carte Raspberry Pi. M0, M1 et la position B sont montrés dans la Figure 3.3, encerclés en rouge. Ce mode présente un mode de communication appelé « Fixed Transmission ».

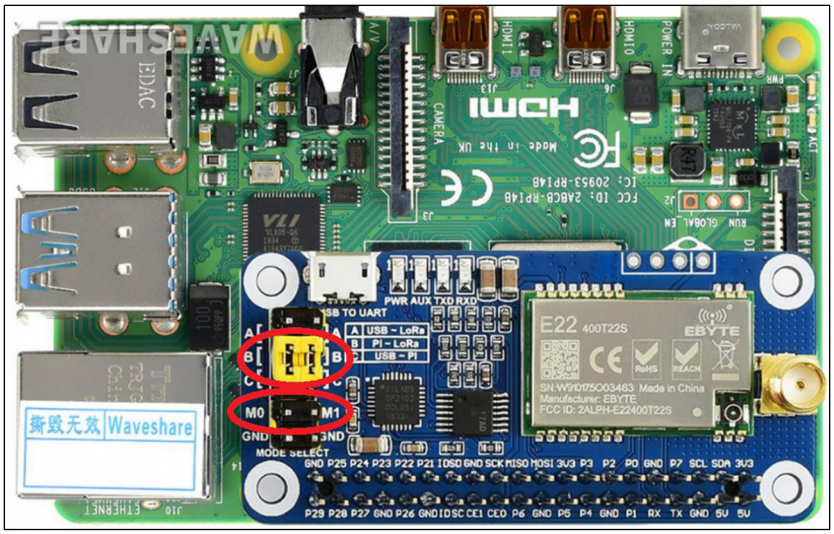

Figure 3.3 Broches M0 et M1 et la position B (File, 2021)

Ce type de transmission permet la communication un à un en spécifiant l'adresse du destinataire et un canal de communication. Nous avons ainsi utilisé ce mode pour pouvoir transmettre les données à un nœud spécifique en définissant son adresse au niveau de code. Ce mode est illustré dans la Figure 3.4 qui est tirée du manuel d'opération du module SX126X LoRa (E22-900T22S1B-, 2018). Dans cette figure, l'expéditeur est à l'adresse 0x0001 du canal 0x02 et le destinataire est le nœud se trouvant à l'adresse 0x0003 du canal 0x04. Les autres modules SX126X LoRa à des adresses différentes ou canaux différents ne pourront recevoir les paquets de données envoyés par l'expéditeur. Le format d'une trame de transmission de ce mode est montré dans le Tableau 3.2.

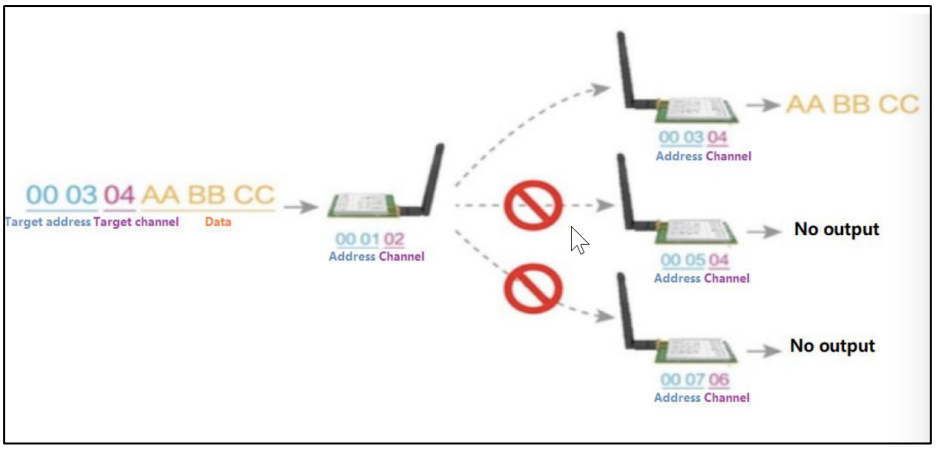

Figure 3.4 Mode « Fixed Transmission » (E22-900T22S1B-, 2018)

Tableau 3.2 Format de transmission en mode « Fixed Transmission »

Champ	Signification
Dest_addr (16 bits)	C'est l'adresse du nœud destinataire.
Dest_chan (8 bits)	C'est le numéro du canal de transmission du nœud.
Payload	Il s'agit des données à transmettre au nœud destinataire.

3.2.1.2 Mode d'opération Configuration

Ce mode permet la configuration du circuit E22 900T22S du module SX1262 915M. Cette configuration s'effectue une seule fois puisque le circuit dispose d'une mémoire non volatile. Pour activer ce mode, nous avons mis M0 et M1 respectivement à 0 et 1. De plus, pour définir la plateforme de contrôle de la carte SX126X LoRa Hat qui la carte Raspberry Pi, on doit mettre les cavaliers à la position B. Les paramètres de configuration et leur réglage sont énumérés dans le Tableau 3.3.

Tableau 3.3 La configuration du coordonnateur

Variable	Valeur	Signification
Baud Rate	57600 bps	C'est la vitesse de transmission
Parity	8N1 (données 8 bits, pas de bit de parité, 1 bit d'arrêt)	Pour la détection des erreurs. Cependant, nous en avons eu besoin juste pour utiliser le champ conservé

		aux données et le bit stop indiquant la fin de la trame de données envoyée.
Variable	**Valeur**	**Signification**
Air Rate	2400 bps	Vitesse de transfert
Packet Size	240 octets	La taille du buffer qui contiendra le message envoyé.
Power	22 dBm	Puissance de transmission
Transmission mode	Fixed	Mode de Transmission Fixed.
Relay	Désactivé	
LBT (*Listened Before Talk*)	Désactivé	Si cette fonctionnalité est activée, elle permet l'envoi des messages en retard s'il y a beaucoup de bruit dans l'environnement.
Packet RSSI	Activé	Cette valeur est une mesure relative pour mesurer la capacité d'un récepteur à entendre un signal et si c'est suffisamment fort pour obtenir une bonne connexion sans fil à partir d'un émetteur.
Channel RSSI	Activé	Le canal qui va recevoir le signal/message.
Address	10, 20 ou 30	C'est l'adresse sur 16 bits du coordonnateur. Les nœuds doivent avoir des adresses différentes.
Channel	18	C'est le canal de communication de valeur 0 à 80. Ça doit être le même pour les nœuds.

3.2.1.3 Lecture des nœuds de contrôle

Dans la procédure de lecture des nœuds de contrôle, le coordonnateur transmet la commande de lecture aux nœuds selon le modèle « chacun son tour » (*round-robin*). La Figure 3.5 illustre le fonctionnement de Round-Robin dans notre cas d'utilisation. La lecture s'effectue par l'envoi de la commande READ vers le nœud cible. Le nœud destinataire répond au coordonnateur en lui retournant des valeurs lues de ses capteurs. Le coordonnateur confirme ainsi au nœud cible la réception des valeurs par la commande ACK.

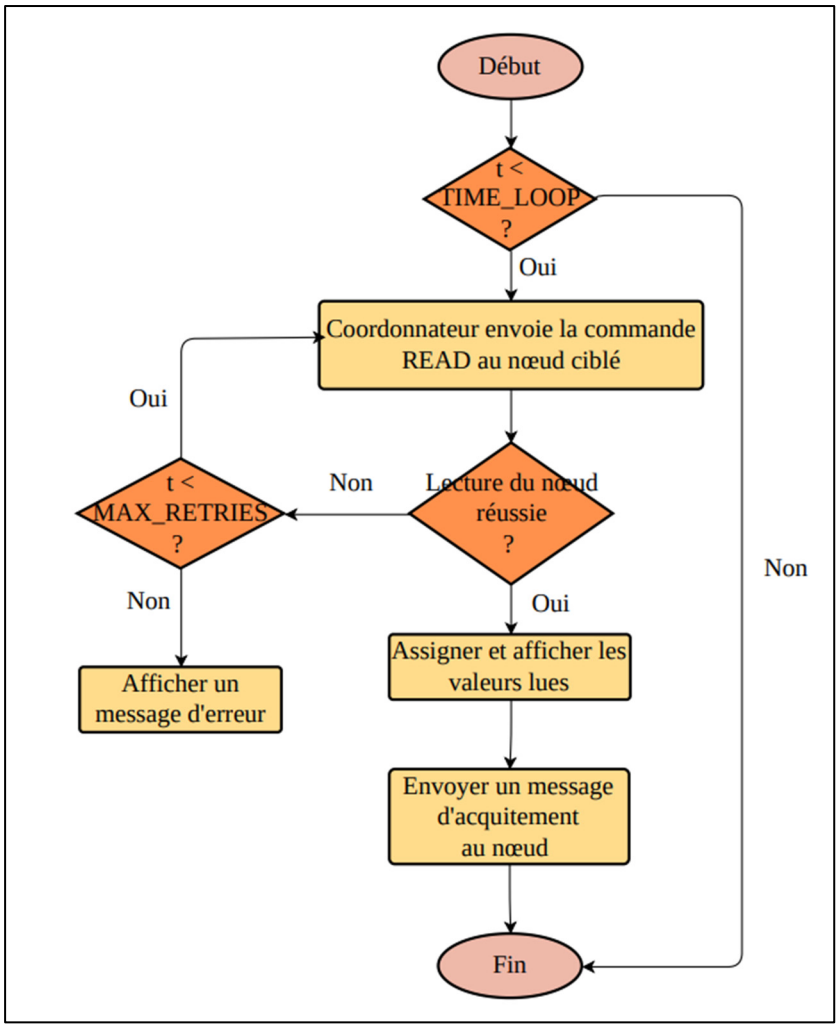

Figure 3.5 Modèle chacun son tour utilisé dans la lecture des nœuds de contrôle

À toutes les `TIME_LOOP` secondes, le coordonnateur transmet la commande `READ` au nœud de contrôle ciblé en utilisant le canal de communication LoRa que nous avons déjà configuré à 18. Si la lecture du nœud est réussie, le nœud assigne et affiche les valeurs lues à partir des capteurs liés au DCC au coordonnateur. Le coordonnateur envoie ainsi un message d'acquittement au nœud pour lui confirmer la lecture des valeurs. Si la lecture échoue et le nombre de tentatives `MAX_RETRIES` n'est pas dépassé, le coordonnateur va encore essayer de lire la réponse du nœud encore une fois. La variable `MAX_RETRIES` indique le nombre de

tentatives autorisées au coordonnateur pour l'envoi de la commande READ. Si le coordonnateur n'a plus le droit à un autre essai, un message d'erreur sera affiché.
Le coordonnateur réalise cette procédure en insérant la commande READ dans la partie payload du paquet de transmission LoRa (voir Tableau 3.2). L'adresse du nœud de destination, quant à elle, est placée dans les 16 premiers bits du paquet de transmission.

3.2.1.4 Justification

Nous avons choisi la carte Raspberry Pi pour jouer le rôle d'un coordonnateur, car elle est peu coûteuse, très répandue et résistante aux perturbations électromagnétiques. En effet, les coordonnateurs sont installés dans des chambres électriques des immeubles. Les tests de compatibilité électromagnétique ont montré une bonne performance des cartes Raspberry Pi. Elles peuvent résister à des interférences électriques, dont l'intensité pouvant aller jusqu'à 10 V/m (Mach et al., 2017). Cette résistance est amplement suffisante pour notre application.

La technologie LoRa est imposée par les concepteurs du système d'acquisition. Les raisons justifiant le choix de cette technologie sont :

- La plupart des chambres électriques, sous-sols, garages de stationnement n'ont pas accès aux réseaux locaux;
- La longue portée de la communication LoRa peut être bénéfique pour couvrir les vastes distances des condominiums et des immeubles multilogements;
- Le réseau formé par la technologie LoRa est indépendant des réseaux LAN ou Wi-Fi du lieu d'installation. Ce qui augmente grandement la sécurité des accès du système d'acquisition des données.

3.2.2 Configuration d'un canal de ThingSpeak

Après la réception des données venant de la communication LoRa des nœuds de contrôle, le coordonnateur doit pouvoir publier ces données vers le broker MQTT d'un canal ThingSpeak.

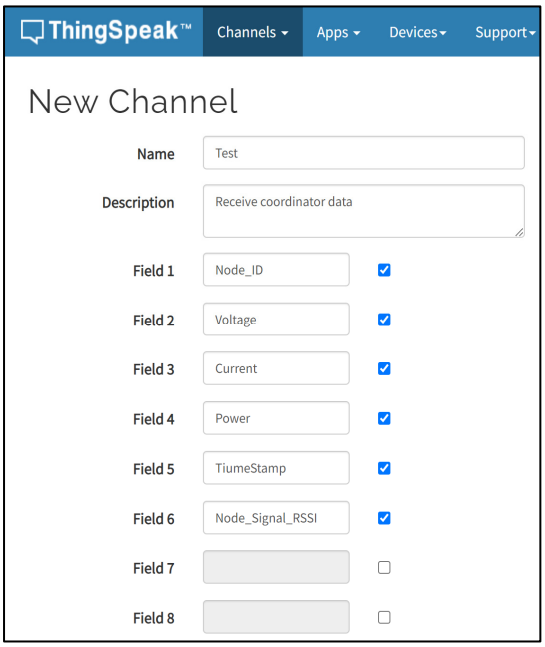

Figure 3.6 Configuration du canal ThingSpeak

Pour ce faire, il faut créer et configurer correctement un canal dédié à cette tâche. La Figure 3.6 montre la configuration du canal utilisé dans ce projet de recherche. Les champs de ce canal sont énumérés dans le Tableau 3.4.

Tableau 3.4 Les champs du canal ThingSpeak utilisés

Champ	Signification
Node_ID	Numéro du nœud de contrôle identifiant une borne de recharge électrique.
Voltage	Tension en volt des phases 1, 2 et 1-2 (c'est un triplet)
Current	Courant en ampère des phases 1, 2 et 1-2 (c'est un triplet)
Power	Puissance en watt des phases 1, 2 et 1-2 (c'est un triplet)

TimeStamp	Le nombre de secondes écoulées depuis 01-01-1970 en format secondes fractionnaires.
Node_Signal_RSSI	La force du signal reçu par le nœud de contrôle de 0 à 255 avec 255 la force maximale du signal.

Après la création du canal, ce dernier aura un identifiant `TS_CHANNEL_ID` et une clé `TS_API_KEY` qui lui sont propres. Les transferts de données vers le canal sont identifiés par ces deux paramètres. Pour pouvoir établir la communication MQTT, il nous faut les paramètres du broker MQTT de ThingSpeak. Le Tableau 3.5 énumère l'identification du broker de ThingSpeak ainsi que les paramètres d'authentification pour le client MQTT.

Tableau 3.5 Paramètres du broker MQTT de ThingSpeak

Paramètres d'identification	**Signification**
TS_MQTT_CLIENT_ID	L'identifiant du client MQTT.
TS_MQTT_BORKER	L'adresse URL du broker MQTT.
TS_MQTT_USERNAME	Le nom du client MQTT.
TS_MQTT_PASSWORD	Le mot de passe du client MQTT.

Chaque lecture de données d'un nœud de contrôle provoque une publication de message MQTT au broker de ThingSpeak. Ainsi, à chaque période d'échantillonnage, *N* messages MQTT sont publiés sur le canal ThingSpeak où *N* est le nombre de nœuds de contrôle. Chaque message MQTT contient des données pour les six (6) champs du canal (voir Tableau 3.4).

3.2.2.1 Justification

Le protocole MQTT, présenté dans la section 1.3.1 de ce mémoire, est tout à fait convenable pour notre environnement d'application. La transmission des données peut être interrompue momentanément à cause des interférences émanant des chambres électriques. Puisque les messages MQTT sont entretenus sur le broker, les abonnés peuvent les recevoir après le rétablissement de la communication. De plus, la communication MQTT peut être étendue sans modifier l'organisation des publicateurs et d'abonnés. En effet, on peut augmenter le nombre de coordonnateurs et de nœuds de contrôle en créant des clients MQTT et en s'abonnant aux

sujets MQTT appropriés définis par le système d'acquisition des données. Enfin, le protocole MQTT est compatible avec la plateforme infonuagique ThingSpeak. Ce qui simplifie grandement l'implantation du système de communication du coordonnateur.

3.3 Conclusion

Pour conclure, dans ce chapitre, nous avons pu définir notre objectif de cette recherche, à savoir être capable de concevoir une passerelle IIoT qui fonctionne avec les protocoles de communication MQTT et LoRa et qui peut être contrôlée à distance par une GUI.

Nous avons ainsi traité l'interaction du coordonnateur avec les nœuds de mesure établie par LoRa. Ceci était possible dû à l'exploitation de la carte SX126X LoRa HAT qui était facile à connecter au coordonnateur grâce à sa compatibilité avec la carte Raspberry Pi. De plus, plusieurs démos ont été offerts dans le site du fournisseur de ce module, ce qui nous a facilité la programmation de cette communication et de comprendre les différents modes qui peuvent être exploités dans notre cas de projet.

L'intégration d'une plateforme IIoT est aussi nécessaire pour la conception complète de notre passerelle. Nous avons donc choisi la plateforme ThingSpeak qui semble suffisante pour répondre à nos besoins. Quelques réglages ont été nécessaires au niveau de cette plateforme pour permettre de configurer la communication MQTT avec le coordonnateur.

De plus, la passerelle présente d'autres fonctionnalités en dehors des protocoles de communication. En effet, une interface graphique a été développée pour commander et surveiller l'état du système en temps réel. Cependant, l'intégration de MQTT au niveau de cette interface était nécessaire pour établir l'interaction avec le coordonnateur.

En résumé, notre système est théoriquement prêt et il nous reste un certain ensemble de tests à faire pour valider notre travail. Nous allons aborder tous ces détails dans le chapitre suivant.

CHAPITRE 4

ÉVALUATION DU SYSTÈME CONÇU

4.1 Introduction

Nous avons présenté la conception de l'outil de gestion et du système de communication des coordonnateurs dans les chapitres 2 et 3. Ces composants du système d'acquisition des données ont été implantés sur les coordonnateurs Raspberry Pi et sur un poste de travail Windows. Pour pouvoir les intégrer au système d'acquisition et les déployer sur le terrain, il nous faut évaluer l'exactitude de fonctionnement de ces composants. Ce chapitre sert ainsi à présenter le fonctionnement du système dans la pratique et à l'évaluer.

Dans un premier temps, nous allons montrer les tests préliminaires réalisés qui consistent à valider le fonctionnement du module LoRa. Par la suite, nous allons aborder les expérimentations réalisées pour valider la communication LoRa entre le coordonnateur et les nœuds de contrôle. À la fin, nous allons évaluer l'outil de gestion tout en examinant sa communication MQTT avec le coordonnateur. Nous allons également couvrir la gestion des erreurs et des exceptions qui peuvent procurer.

4.2 Validation des modules LoRa

LoRa est un protocole de communication s'exécutant sur un modem à étalement de spectre. Il est donc nécessaire de configurer et tester ce matériel avant son intégration dans le système d'acquisition de données. Ainsi, le test préliminaire consiste à valider le fonctionnement des modules SX1262 915M LoRa HAT à l'aide des cartes Raspberry Pi.

4.2.1 Modules SX1262 LoRa

Les modes d'opération du module SX1262 LoRa sont expliqués dans la section 3.2.1 du chapitre 3. Le mode « Fixed Transmission » a été sélectionné pour réaliser la communication entre un coordonnateur et ses nœuds de contrôle du système d'acquisition de données. Donc, deux modules SX1262 LoRa, l'un jouant le rôle du coordonnateur et l'autre jouant le rôle d'un nœud de contrôle, sont nécessaires pour valider le fonctionnement de ce mode de communication. Les paramètres de configuration de ces deux modules sont ceux donnés dans le Tableau 3.3 du chapitre 3. Les adresses utilisées dans ces tests sont respectivement 50 et 701.

Figure 4.1 Position des cavaliers
des modules LoRa

Pour pouvoir relier les modules SX1262 LoRa à des cartes Raspberry Pi, il est nécessaire de placer les cavaliers à la position B et les broches M0 et M1 à la position ouverte. Ce branchement physique est montré dans la photo de la Figure 4.1.

4.2.1.1 Test du mode d'opération « Fixed Transmission »

Dans ce test, le coordonnateur à l'adresse 50 débute par la transmission d'une commande « PING » qui un code numérique défini dans le système d'acquisition de données. Le nœud à l'adresse 701, en recevant cette commande, doit répondre par le code « ACK » (*acknowledge*) au coordonnateur. Le contenu des champs de la transmission est montré dans le Tableau 4.1.

Tableau 4.1 Trame de transmission dans le scénario de test

Champ	Valeur
Adresses	Coordonnateur vers nœud : 701 (adresse du nœud) Nœud vers coordonnateur : 50 (adresse du coordonnateur)
Canal de communication	18
Payload LoRa	Commande « PING »

4.2.1.2 Résultat du test

Les Figures 4.2 et 4.3 montrent les résultats de la communication LoRa « Fixed Transmission » de ce test.

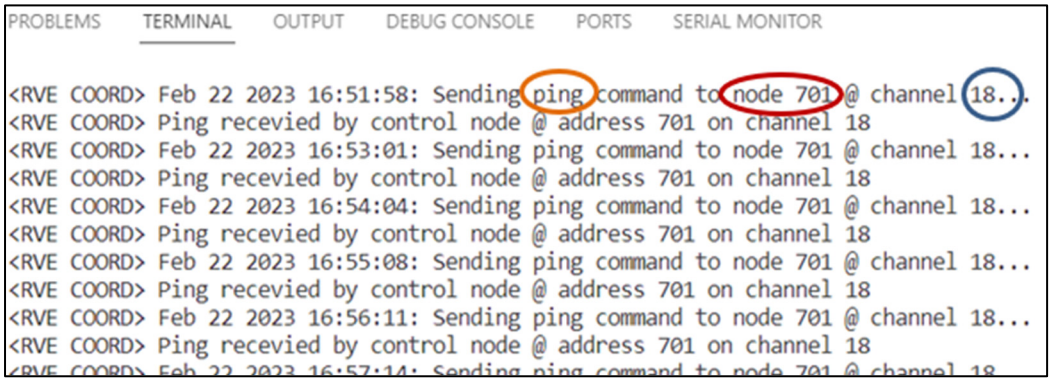

Figure 4.2 Coordonnateur transmet périodiquement la commande « PING » au nœud à l'adresse 701 du canal 18

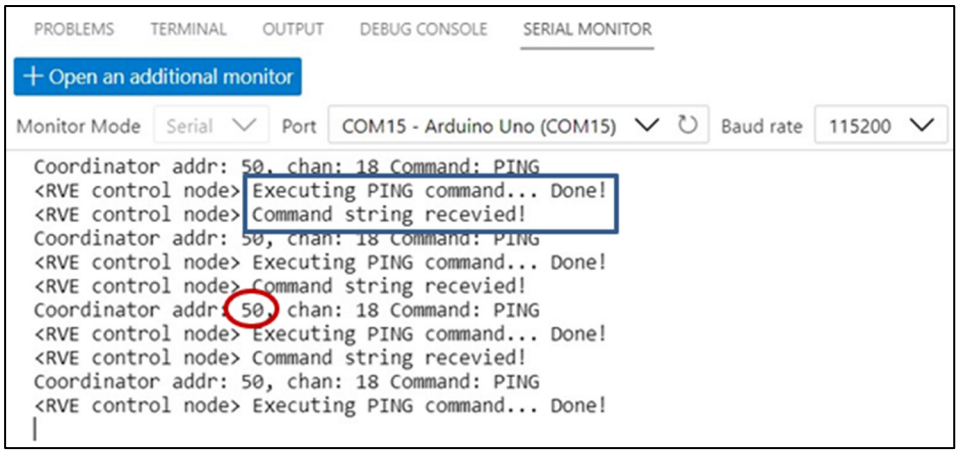

Figure 4.3 Réception de la commande « PING » du coordonnateur par un nœud

La Figure 4.2 montre que le coordonnateur a réussi à transmettre périodiquement la commande « PING » au nœud et reçoit correctement sa réponse « ACK ». Le nœud reçoit avec succès la commande du coordonnateur comme en témoigne la Figure 4.3. On voit d'après ce résultat, le matériel et la programmation de ce module fonctionnent comme prévu.

4.3 Validation de la communication coordonnateur – nœuds de contrôle

Nous poursuivons le travail d'évaluation en intégrant les modules LoRa dans la boucle de communication entre un coordonnateur de ses nœuds de contrôle. Rappelons que la commande coordonnateur – nœuds s'effectue selon le modèle « chacun son tour ». La logique de cette boucle de communication a été illustrée dans la Figure 3.4 du chapitre 3. À toutes les `TIME_LOOP` secondes, le coordonnateur transmet la commande « READ » aux nœuds de contrôle ciblés. Dans le test présent, la valeur du `TIME_LOOP` est réglée à 60 secondes. Les valeurs lues des nœuds de contrôle sont affichées par le coordonnateur. Ce dernier transmet ainsi un message d'acquittement aux nœuds pour leur signifier la bonne réception des données.

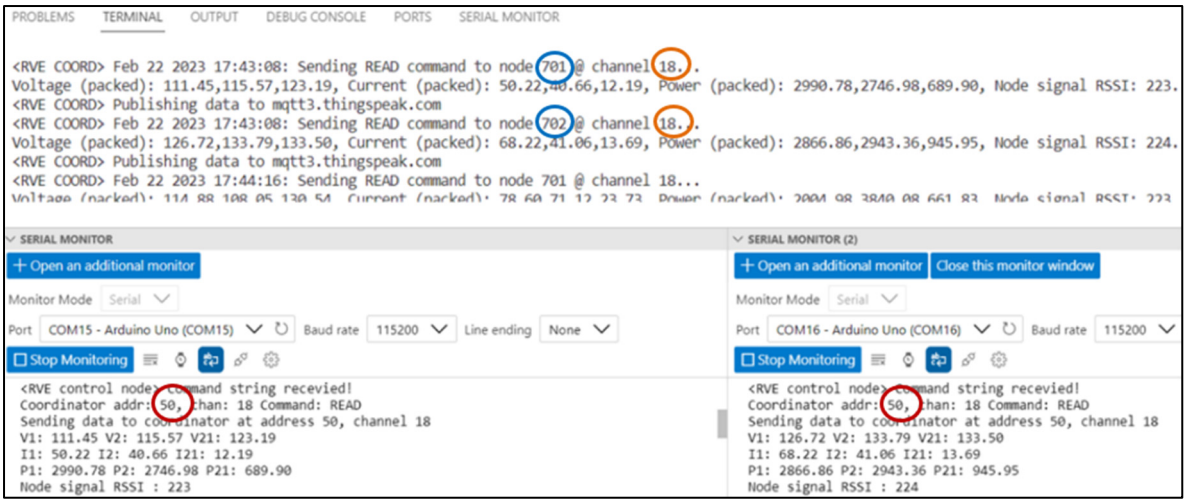

Figure 4.4 Transmission de la commande « READ » aux nœuds et la réception des valeurs lues

La Figure 4.4 montre le résultat d'un test où le coordonnateur à l'adresse 50 envoie la commande READ séquentiellement aux nœuds 701 et 702. Les nœuds de contrôle affichent un message indiquant la bonne réception de la commande et transmettent les données échantillonnées par ses capteurs au coordonnateur. La même procédure se répète après `TIME_LOOP` secondes. Advenant le cas où un nœud ne répond pas à la commande, le coordonnateur doit répéter la transmission de la commande `MAX_RETRIES` fois avant de déclarer forfait et déclarer l'échec de lecture pour le nœud.

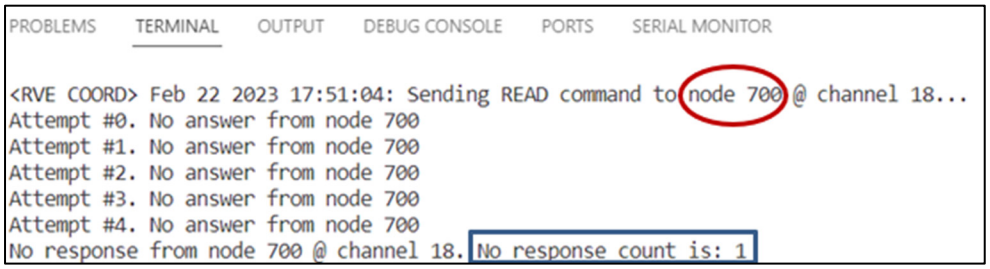

Figure 4.5 Affichage du coordinateur après `MAX_RETRIES` essais sans réponse du nœud

À noter que le coordonnateur maintient le compte de sans réponse pour chacun des nœuds à sa charge. De cette façon, l'outil de gestion peut interroger le coordonnateur pour connaître le nombre de fois qu'un nœud n'a pas répondu à la commande de lecture.

4.4 Validation de l'outil de gestion

L'outil de gestion est une partie cruciale dans le système d'acquisition de données, car c'est avec laquelle que l'utilisateur pourra contrôler et interroger l'état du système d'acquisition des données. L'évaluation de l'outil de gestion développée est divisée en deux parties : test de fonctionnement l'outil de gestion et test d'utilisabilité.

4.4.1 Tests de l'outil de gestion

Ces tests concernent les fonctionnalités de l'outil de gestion. Il s'agit d'identifier les problèmes de l'outil et de vérifier si l'application répond aux exigences spécifiées. Parmi les problèmes potentiels, on peut citer les erreurs logiques d'exécution et les erreurs d'affichage. Le premier test vérifie le bon fonctionnement des listes déroulantes utilisées dans le choix des paramètres du système d'acquisition. La Figure 4.4 montre l'exécution correcte des listes déroulantes pour le choix du coordonnateur et des commandes.

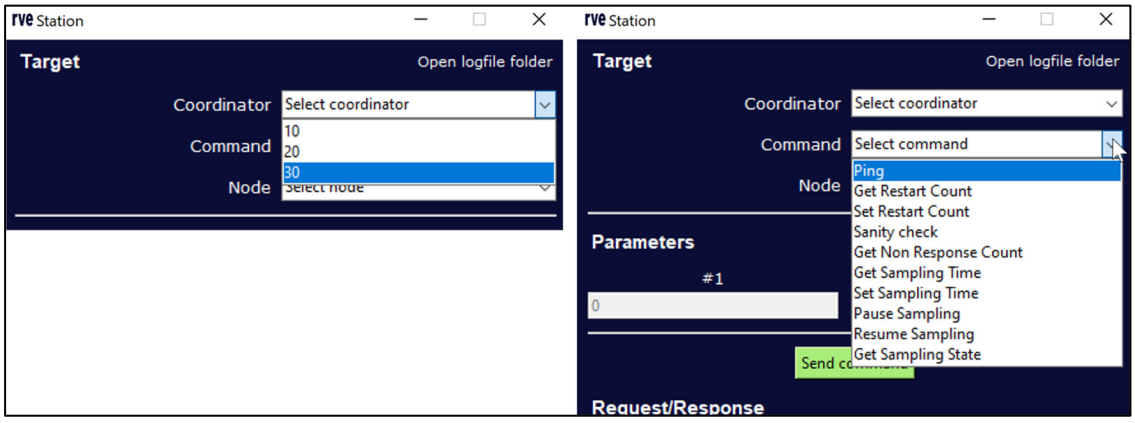

Figure 4.4 Exécution des listes déroulantes

Rappelons que la liste des coordonnateurs représente l'adresse des coordonnateurs. Ces derniers sont installés dans des immeubles en copropriété ou des bâtiments multilogements. La liste des commandes, quant à elle, contient les commandes définies par le système d'acquisition et ont été présentées dans le Tableau 2.1.

Le deuxième test vérifie le changement des nœuds en fonction de la sélection du coordonnateur. Dans le système d'acquisition des données, chaque coordonnateur est affecté à un ensemble de nœuds de contrôle.

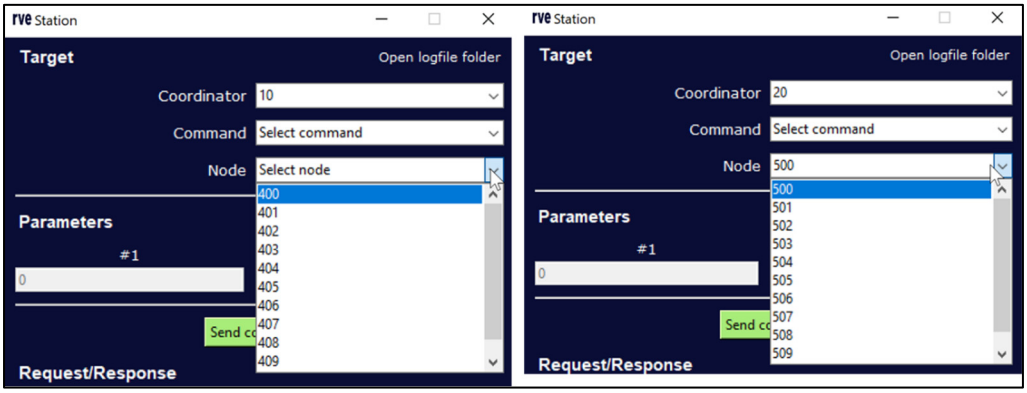

Figure 4.5 Changement des adresses des nœuds en fonction de la sélection du coordonnateur

Le fonctionnement du « Fixed transmission » de la communication LoRa et le modèle publication – abonnement de la communication MQTT font en sorte que chacun des coordonnateurs et chacun des nœuds de contrôle doit avoir une adresse différente. La Figure 4.5 montre le résultat de ce test où l'adresse des nœuds est modifiée par la sélection d'un coordonnateur.

Le troisième test effectué détermine l'exactitude de la séquence d'activation de `param1` et de la liste des nœuds de contrôle. Cette séquence d'activation a été décrite dans la section 2.2.4 du chapitre 2. Aussi, la logique d'implantation de cette séquence a été présentée dans la Figure 2.5. Selon cette séquence d'activation, la liste déroulante des nœuds doit être désactivée

lorsque la commande sélectionnée est « Get Sampling Time », « Set Sampling Time », « Pause » ou « Resume ». La Figure 4.6 est un exemple montrant ce résultat.

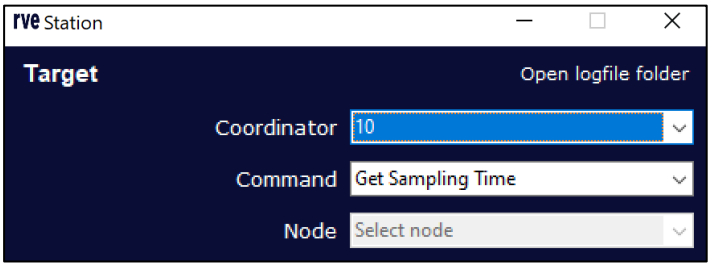

Figure 4.6 Désactivation de la liste des nœuds après la sélection de la commande « Get Sample Time »

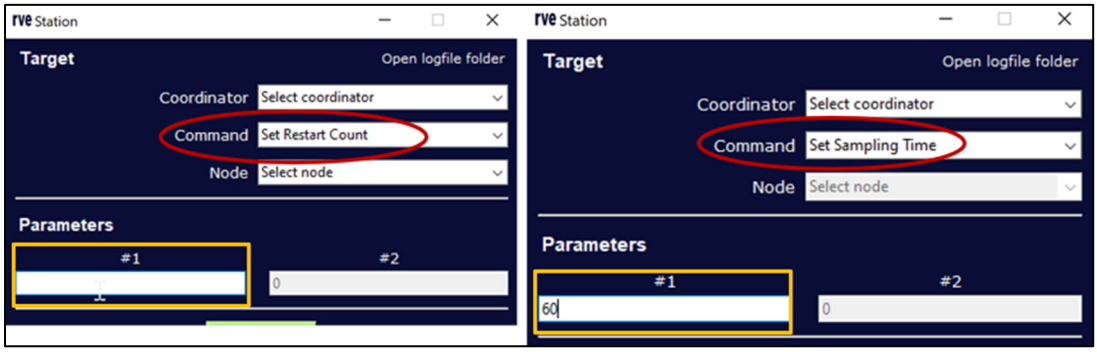

Figure 4.7 Activation du champ Param1 lors de la sélection des commandes « Set Restart Count » et « Set Sampling Time »

Toujours selon cette séquence d'activation, le champ représentant `Param1` doit être activé après la sélection des commandes « Set Restart Count » et « Set Sampling Time ». Ce résultat est montré dans la Figure 4.6.

4.4.1.1 Gestion des erreurs et des exceptions

Les erreurs peuvent être nombreuses dans un système de ce type. Leurs origines sont également très diverses, elles peuvent provenir d'erreurs matérielles, logicielles ou humaines. Nous avons veillé à ce que notre coordonnateur soit aussi robuste que possible. Dans cette sous-section, nous allons tester le traitement de différents types d'erreur.

4.4.1.2 Gestion des erreurs humaines

L'utilisateur peut commettre des erreurs lors de la saisie des données. C'est pourquoi des listes déroulantes sont appliquées pour empêcher l'erreur dans le choix des paramètres. Cependant, la plage des valeurs du champ `param1` varie selon la commande sélectionnée. Si la valeur saisie est hors de l'intervalle, un message d'erreur doit être affiché pour informer l'utilisateur. La Figure 4.8 est le résultat d'un test de validité du champ `param1`.

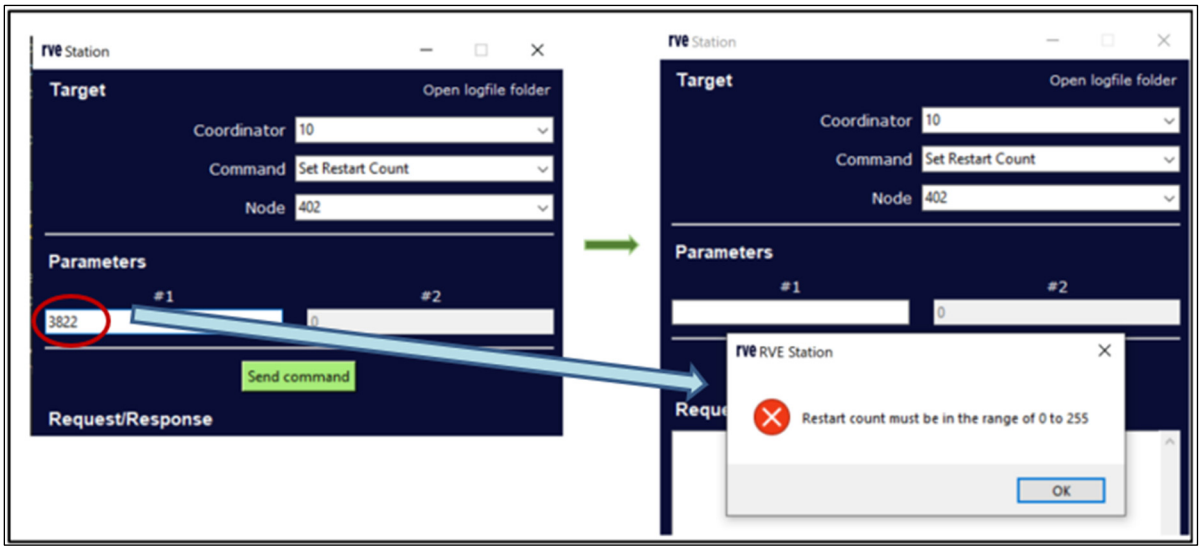

Figure 4.8 Test de gestion d'erreur quand la valeur de `param1` n'est pas valide

Aussi, l'utilisateur peut oublier la sélection de l'adresse du coordonnateur ou du nœud de contrôle. Dans ce cas, la gestion de l'erreur humaine s'exécute et le résultat est montré dans la Figure 4.9.

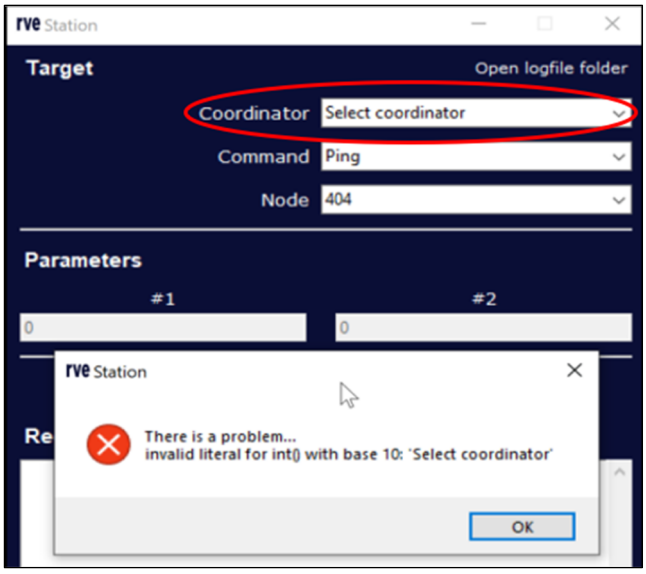

Figure 4.9 Test de gestion d'erreur pour une valeur qui est manquante

4.4.1.3 Gestion de l'erreur de connexion

L'outil de gestion repose sur la communication MQTT pour réaliser ses fonctions de contrôle et de surveillance des coordonnateurs. Ces derniers doivent se connecter au broker MQTT pour pouvoir recevoir les commandes de l'outil de gestion. Le test sur la gestion de l'erreur de connexion est réalisé en deux (2) phases : 1) test sur l'existence d'un réseau sans fil ou filaire, 2) test de connexion au broker MQTT. Dans la première phase, l'absence d'un réseau est une erreur qui provoquera la fin du programme exécutant sur les coordonnateurs. Sans un réseau de communication, les coordonnateurs ne peuvent pas réaliser leurs fonctions.

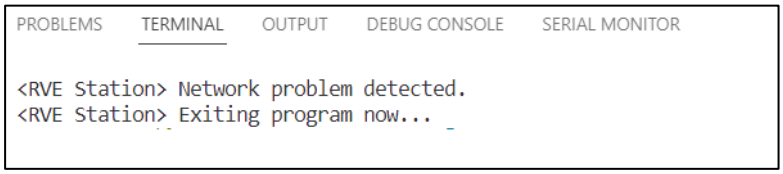

Figure 4.10 Fin de l'exécution de l'outil de gestion causée par l'absence du réseau

Il en est de même pour l'outil de gestion. Sans un réseau sans-fil ou filaire, aucune commande ne pourra être acheminée aux coordonnateurs. La Figure 4.10 montre le comportement de l'outil résultat de l'absence d'un réseau de communication.

Un autre problème potentiel de réseau est au niveau de la connexion du broker MQTT. Dans ce cas, le réseau de communication est disponible, mais le broker MQTT n'accepte pas la requête de connexion envoyée. Ce problème survient chez le broker et est hors du contrôle de l'utilisateur. Au lieu de terminer l'exécution des programmes, les coordonnateurs et l'outil de gestion feront des tentatives pour se connecter au broker. Ces tentatives seront réalisées en arrière-plan pendant l'exécution des programmes. La Figure 4.11 montrer le résultat de cette gestion de connexion par un coordonnateur. La même technique est implantée dans le système de communication de l'outil de gestion.

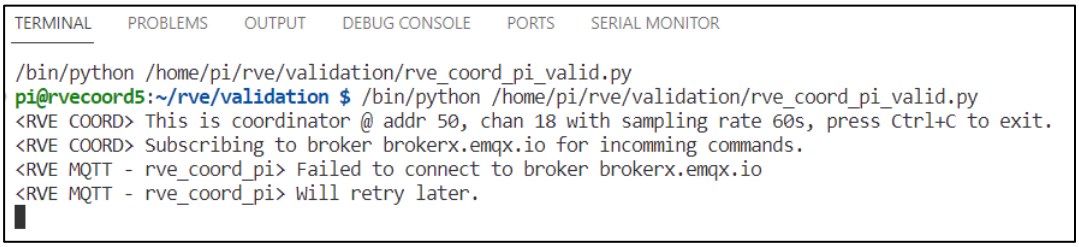

Figure 4.11 Gestion de l'erreur de connexion au broker MQTT

4.4.2 Tests d'utilisabilité

Les tests d'utilisabilité consistent à tester une application pour déterminer sa facilité d'utilisation et sa convivialité. Ce type de test est utilisé pour s'assurer que les utilisateurs peuvent facilement apprendre à manipuler l'application et effectuer des tâches de l'outil de gestion. Pour ce faire, nous avons exécuté un ensemble de manipulations qui consiste à envoyer différentes commandes aux différents coordonnateurs et nœuds de contrôle. La Figure 4.12 montre le résultat d'une commande envoyée (voir le numéro 2 dans la figure) au coordonnateur à l'adresse 10 (voir le numéro 1 dans la figure).

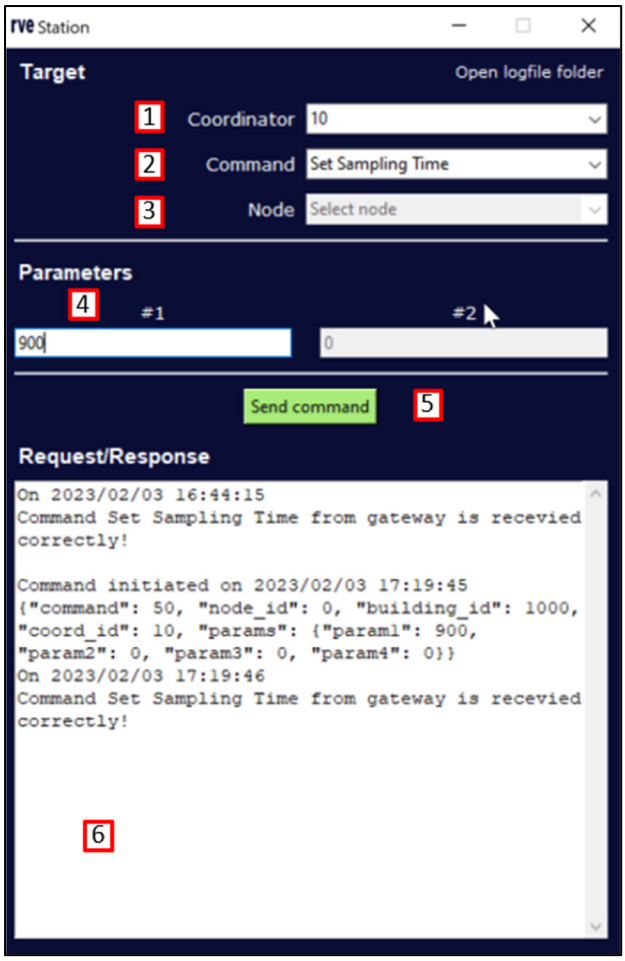

Figure 4.12 Envoi d'une commande
vers un coordonnateur

La commande « Set Sampling Time » consiste à régler la période d'échantillonnage du coordonnateur ciblé. La période d'échantillonnage est indiquée par l'utilisation dans le champ du `Param1` (voir le numéro 4 dans la figure). Rappelons que la liste des commandes du système d'acquisition de données est donnée dans le Tableau 2.1 du chapitre 2. La Figure 4.12 montre la commande dans le format JSON et la réponse du coordonnateur (voir le numéro 6 dans la figure). Ce test confirme le fonctionnement de la communication MQTT entre l'outil de gestion et le coordonnateur destinataire.

Les communications entre l'outil de gestion et les coordonnateurs sont enregistrées dans des fichiers locaux. Un fichier d'archive est créé à chaque démarrage de l'outil de gestion. Pour accéder à ces fichiers, l'utilisateur doit choisir le lien « Open logfile folder » de l'interface graphique qui le redirigera vers le dossier contenant ces fichiers. Cette fonctionnalité a été validée et le résultat est montré dans la Figure 4.13.

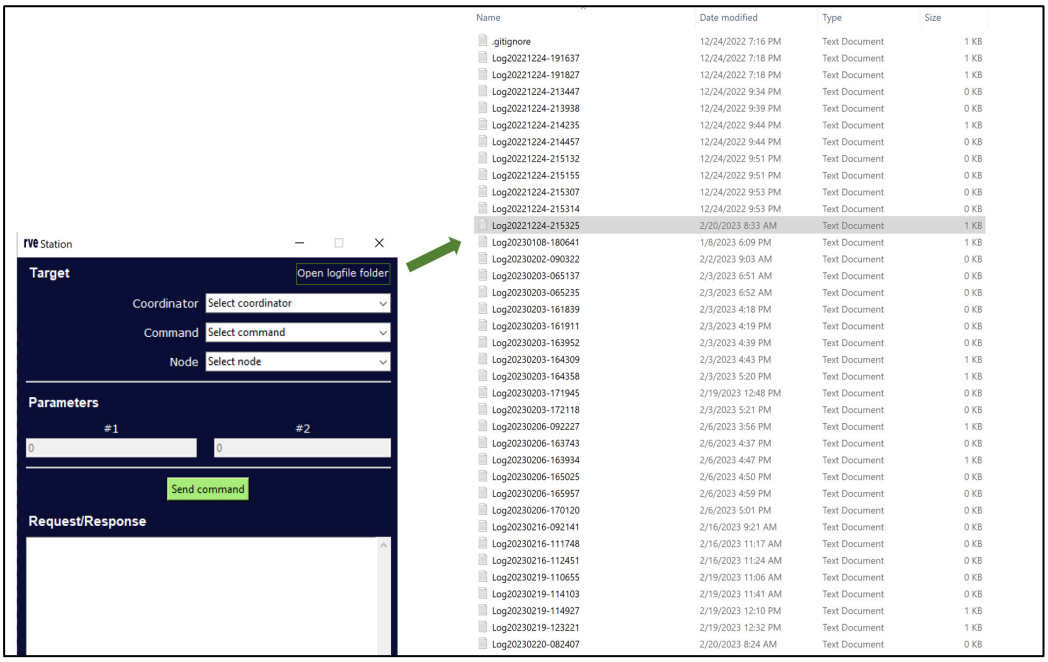

Figure 4.13 Fichiers archivant les communications

La Figure 4.14 illustre le contenu d'un fichier d'archive datant du 24 décembre 2022 à 21h 53min. On y voit deux commandes envoyées (« PING » et « Restart Count ») et les réponses reçues du coordonnateur à l'adresse 10.

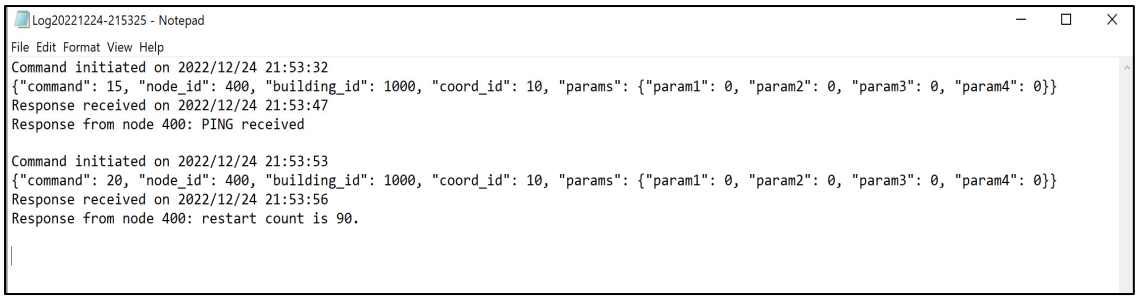

Figure 4.14 Exemple d'un fichier de stockage des commandes et réponses du coordonnateur

Après la réception des données venant de leurs nœuds de contrôle, les coordonnateurs doivent publier les données sur le canal ThingSpeak pour le système d'acquisition de données. La Figure 4.15 donne une vue des tableaux de bord de ThingSpeak générés pour ce projet de recherche.

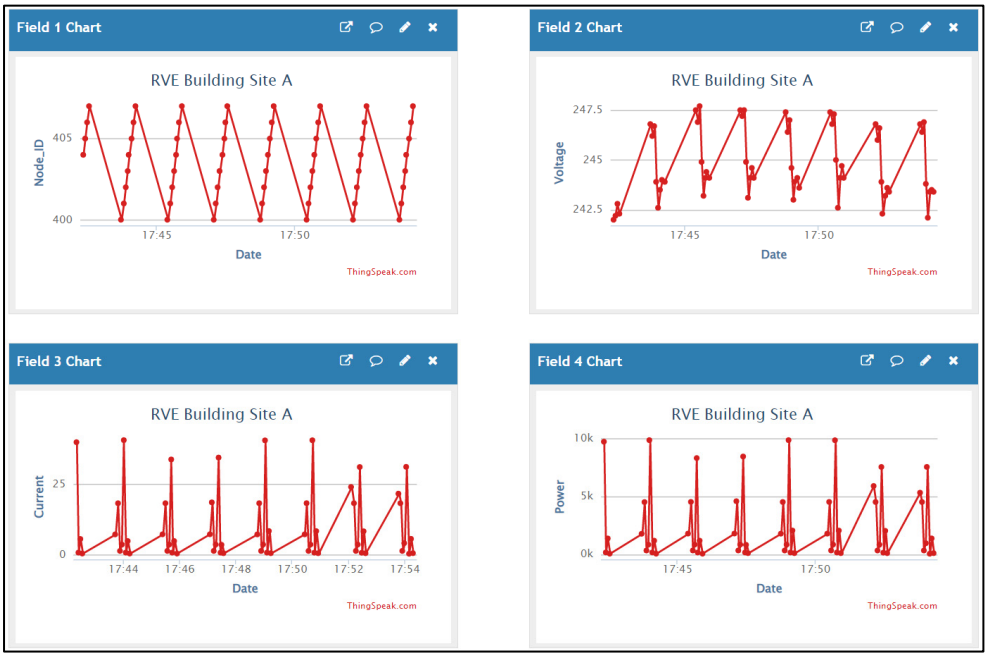

Figure 4.15 Tableaux de bord générés par ThingSpeak

La Figure 4.16 illustre le cas où il y'a un problème de connexion et de publication de données sur ThingSpeak. C'est à ce moment qu'un message indiquant l'échec de transmission des données vers ThingSpeak sera affiché.

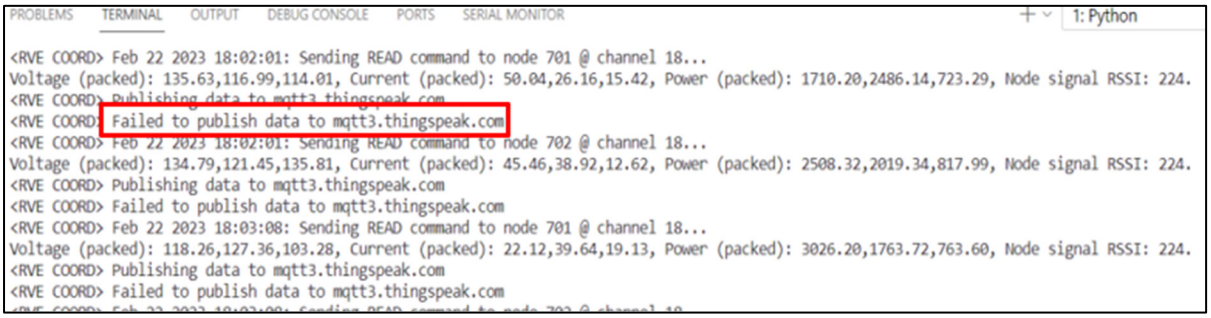

Figure 4.16 Exemple montrant un problème de transmission de données vers ThingSpeak

4.5 Conclusion

En conséquence, la passerelle IIoT conçue, offrant à l'utilisateur de diverses fonctionnalités qui ont été testées avec succès. Dans un premier temps, nous nous sommes penchés sur l'aspect matériel pour assurer que les dispositifs que nous avons fonctionnent correctement avant de les assembler. Précisément, les modules LoRa avec la carte Raspberry Pi.

Par la suite, nous avons testé la communication LoRa établie entre le coordonnateur et les nœuds de contrôle. Nous avons également testé l'interface développée en vérifiant si elle est fonctionnelle et est conforme au cahier de charge du client. L'implantation des protocoles MQTT et LoRa a bien été validée. En effet, nous avons pu recueillir des commandes venant de l'outil de gestion par notre coordonnateur via MQTT, les passer aux nœuds concernés via LoRa qui retournent à leur tour des réponses à la passerelle et enfin stocker et visualiser ces données sur ThingSpeak via MQTT.

Dans l'ensemble, le système mis en place fonctionne conformément à nos attentes. Cependant, nous avons dû gérer d'une manière efficace les exceptions qui peuvent rendre le

fonctionnement du système défectueux. De mauvaises manipulations de la part de l'utilisateur, du mauvais branchement de matériel ou des fautes de saisie de certains paramètres au niveau de code peuvent être la source d'une faille technique. Nous avons ainsi fait des traitements supplémentaires nous permettant à éviter ces situations et pour rendre notre système plus robuste.

CONCLUSION ET RECOMMANDATIONS

Le présent travail de recherche nous a ainsi amenés à la conception d'un outil de gestion pour un système d'acquisition de données appliqué à la recharge de véhicules électrique. Nous avons détaillé à travers ce mémoire les différentes étapes menant à la réalisation de ce projet de recherche. Dans un premier temps, nous avons présenté le contexte du projet, la problématique et nos objectifs. Nous avons ensuite effectué la revue de la littérature pour présenter le cadre général du projet et explorer les concepts de base et les solutions existantes pour concevoir l'outil de gestion en intégrant des technologies peu coûteuses et fiables. Le chapitre 2 a présenté les étapes de développement de l'outil sous forme d'une interface graphique et les approches utilisées. Le chapitre suivant illustre les différentes phases de la conception du coordonnateur tout en élaborant les cas d'utilisation et en expliquant les algorithmes implémentés. Le dernier chapitre a étalé les tests effectués sur le matériel et le logiciel afin de valider le fonctionnement et évaluer la performance de l'outil conçu. L'outil de gestion est par ailleurs compatible avec un usage dans des bâtiments multilogements.

Cette recherche nous a ainsi amenés à la réalisation d'un outil de gestion capable d'interagir avec des dispositifs IIoT et une plateforme infonuagique. L'outil réalisé permet à un utilisateur de contrôler à distance et de recevoir des évènements en temps réel. L'intégration du protocole de communication MQTT s'est avérée efficace pour maintenir la communication à distance entre l'outil de gestion, les coordonnateurs et les nœuds de contrôle. L'outil est capable de transmettre des commandes vers les nœuds de contrôle via les coordonnateurs et de lire leurs réponses.

En termes d'amélioration possible, on note l'utilisation d'une autre bibliothèque python plus avancée que la bibliothèque tkinter dans la création de l'interface graphique. Des travaux futurs peuvent viser à trouver des solutions pour communiquer à plus d'un coordonnateur à la fois. En effet, l'outil de gestion n'est pas capable de traiter plusieurs coordonnateurs dans le même instant. Il doit attendre la réponse du coordonnateur x avant de pouvoir recevoir la réponse

d’un autre coordonnateur *y*. Le coordonnateur *y* est alors bloqué jusqu’à ce qu’il reçoive le message d’acquittement. Ce problème de blocage pourrait se manifester lorsqu’un nombre plus grand de coordonnateurs seront en jeu. Il sera intéressant de mener une étude de recherche pour explorer les différentes solutions possibles à traiter ce problème.

Il sera aussi intéressant de concevoir une méthode qui permettra à l’outil de gestion de distinguer entre un coordonnateur inexistant et un coordonnateur qui ne répond pas à une requête. Un coordonnateur inexistant est celui qui a été enlevé du système d’acquisition. Un coordonnateur peut ne pas répondre à une requête durant sa période de lecture des capteurs. Une solution qui ne nécessite pas une refonde complète de l’outil de gestion est souhaitée. L’utilisation de l’apprentissage par renforcement (RL) pourra servir dans ce cas. Cet algorithme permettra d'apprendre et découvrir d’une manière autonome les actions qui ont eu lieu dans un environnement, sans l’intervention de l’utilisateur de l’outil de gestion.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

A. Karmakar, N. Dey, T. Baral, M. Chowdhury, & M. Rehan. (2019). Industrial Internet of Things: A Review. *2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*, 1–6. https://doi.org/10.1109/OPTRONIX.2019.8862436

Aceituno, J., & Roussel, N. (2014). Douglas Engelbart, inventeur et visionnaire. *Bulletin 1024*. https://doi.org/10.48556/SIF.1024.2.153

admin. (2018, June 19). Pros and Cons of Adaptive User Interfaces. *Nelson Miller*. Repéré à https://nelson-miller.com/pros-and-cons-of-adaptive-user-interfaces/

Amer, S. (2016). Ethical concerns regarding the use of Intelligent User Interfaces. *Proceedings on the International Conference on Internet Computing (ICOMP)*, 164.

AVÉQ. (2023). AVÉQ - Association des Véhicules Électriques du Québec. Repéré à http://www.aveq.ca/1/post/2023/02/statistiques-saaq-aveq-sur-lelectromobilite-au-quebec-en-date-du-31-decembre-2022-infographie.html

Birss, E. W. (1984). The Integrated Software and User Interface of Apple's Lisa. *Proceedings of the July 9-12, 1984, National Computer Conference and Exposition*, 319–328. https://doi.org/10.1145/1499310.1499351

Botta, C., Pierangelini, L., & Vollero, L. (2020). *IoT Gateways for Industrial and Medical Applications: Architecture and Performance Assessment* (p. 599). https://doi.org/10.1109/MetroInd4.0IoT48571.2020.9138268

Calabretta, M., Pecori, R., Vecchio, M., & Veltri, L. (2018). MQTT-Auth: A token-based solution to endow MQTT with authentication and authorization capabilities. *Journal of Communications Software and Systems*, *14*(4), 320–331.

E22-900T22S1B-. (2018). Repéré à https://www.ebyte.com/en/product-view-news.html?id=467

EMQX. (2023). [Erlang]. EMQ Technologies. Repéré à https://github.com/emqx/emqx (Original work published 2012)

Engelbart, D. C., & English, W. K. (1968). A research center for augmenting human intellect. *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, 395–410.

File:SX1268-433M-LoRa-HAT-006.jpg—Waveshare Wiki. (2021). Repéré à https://www.waveshare.com/wiki/File:SX1268-433M-LoRa-HAT-006.jpg

Gonzalez Calleros, J. M., Guerrero García, J., González, C., & Galicia, E. (2018). Is natural user interaction really natural? An evaluation of gesture-based navigating techniques in virtual environments. *Computación y Sistemas*, *22*(1), 255–269.

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., & Cajander, Å. (2003). Key principles for user-centred systems design. *Behaviour & Information Technology*, *22*(6), 397–409. https://doi.org/10.1080/01449290310001624329

Hartson, R., & Pyla, P. S. (2012). *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier.

Hydro-Québec.(2022). Repéré à https://www.hydroquebec.com/data/electrification-transport/pdf/recharge-vehicules-electriques-multilogement.pdf

J. B. Ibarra, E. Chua, J. Turingan, L. Corvera, J. C. Guanlao, C. Leuven Magampon, & M. J. Ocampo. (2019). Development of WiFi-Based Convenience Outlet with Graphical User Interface for Monitoring Power Consumption of RFID-Based Appliances. *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 1–5. https://doi.org/10.1109/HNICEM48295.2019.9073394

Jakob Nielsen. (2017). *A 100-Year View of User Experience*. Nielsen Norman Group. Repéré à https://www.nngroup.com/articles/100-years-ux/

Kim, S.-M., Choi, H.-S., & Rhee, W.-S. (2015). IoT home gateway for auto-configuration and management of MQTT devices. *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, 12–17.

Lampkin, V., Leong, W. T., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R., Kallas, G., Krishna, N., Fassmann, S., & Keen, M. (2012). *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*. IBM Redbooks.

Light, R. (2021). *paho-mqtt: MQTT version 5.0/3.1.1 client class* (1.6.1) [Python; MacOS :: MacOS X, Microsoft :: Windows, POSIX]. Repéré à http://eclipse.org/paho

Mach, V., Kovář, S., Valouch, J., Adamek, M., & Silva, R. (2017). *Electromagnetic compatibility of Raspberry Pi development platform in near and far-field* (p. 2472). https://doi.org/10.1109/PIERS-FALL.2017.8293550

Marc. (2022, June 9). Expérience Utilisateur & SEO: Comment Optimiser Votre Site ? *Twaino*. Repéré à https://www.twaino.com/seo/experience-utilisateur-seo/

Maureira, M. A. G., Oldenhof, D., & Teernstra, L. (2011). ThingSpeak–an API and Web Service for the Internet of Things. *World Wide Web*, *25*, 1–4.

Molina, A. I., Giraldo, W. J., Gallardo, J., Redondo, M. A., Ortega, M., & García, G. (2012). CIAT-GUI: A MDE-compliant environment for developing Graphical User Interfaces of information systems. *Advances in Engineering Software*, *52*, 10–29.

Nettikadan, D., & Raj, S. (2018). Smart community monitoring system using Thingspeak IoT platform. *International Journal of Applied Engineering Research*, *13*(17), 13402–13408.

Norman, D. A. (1988). *The psychology of everyday things.* Basic books.

Norman, D. A., & Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc.

P. Kumar & U. C. Pati. (2016). IoT based monitoring and control of appliances for smart home. *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 1145–1150. https://doi.org/10.1109/RTEICT.2016.7808011

pyinstaller: PyInstaller bundles a Python application and all its dependencies into a single package. (5.9.0). (2023). [C, Python; MacOS :: MacOS X, Microsoft :: Windows, POSIX, POSIX :: AIX, POSIX :: BSD, POSIX :: Linux, POSIX :: SunOS/Solaris]. Repéré à https://www.pyinstaller.org/

Rothrock, L., Koubek, R., Fuchs, F., Haas, M., & Salvendy, G. (2002). Review and reappraisal of adaptive interfaces: Toward biologically inspired paradigms. *Theoretical Issues in Ergonomics Science*, *3*, 47–84. https://doi.org/10.1080/14639220110110342

S. Dehingia, N. M. Kakoty, R. Gokhlani, G. V. Prasad, S. J. Sarma, & D. Sonowal. (2015). User interface system of an assistive kitchen for aging people. *2015 International Conference on Man and Machine Interfacing (MAMI)*, 1–5. https://doi.org/10.1109/MAMI.2015.7456596

Soni, D., & Makwana, A. (2017). A survey on mqtt: A protocol of internet of things (iot). *International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT-2017)*, *20*, 173–177.

Srinath, K. R. (2017). Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, *4*(12), 354–357.

SX1262 915M LoRa HAT - Waveshare Wiki. (2023). Repéré à https://www.waveshare.com/wiki/SX1262_915M_LoRa_HAT

The HiveMQ Team. (2015). *MQTT Publish, Subscribe & Unsubscribe - MQTT Essentials: Part 4*. Repéré à https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/

W. D. Kelley, J. W. Opiekun, & S. L. Teller. (2000). In-orbit satellite antenna measurements. *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in Conjunction with: USNC/URSI National Radio Science Meeting (C*, *2*, 542–545 vol.2. https://doi.org/10.1109/APS.2000.875200

Wigdor, D., & Wixon, D. (2011). *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture* (1st ed.). Morgan Kaufmann Publishers Inc.

Xu, H., König, L., Cáliz, D., & Schmeck, H. (2018). A generic user interface for energy management in smart homes. *Energy Informatics*, *1*(1), 55. https://doi.org/10.1186/s42162-018-0060-0

Zourmand, A., Hing, A. L. K., Hung, C. W., & AbdulRehman, M. (2019). Internet of things (IoT) using LoRa technology. *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 324–330.