الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
National Higher School of Advanced Technologies

Département Génie Électrique et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme

D'INGENIEUR d'État

-Filière-

Télécommunication

-Spécialité -

Système de télécommunications et réseaux

- Thème -

Etude et conception d'une passerelle IIoT avec transmetteur radio nRF24L01, BLE et Wi-Fi basée sur le SoC ESP32

Réalisé par

BEGHAMI Akram

SEBAA Hicham

Les membres de Jury :

BOUTERFAS Malika	Président	MCA ENSTA
ABBAD Leila	Examinateur	MAA ENSTA
HABANI Lamia	Examinateur	MAA ENSTA
SAADI Hadjer	Encadrante	MCA ENSTICP
BELLOULA Abdelmalek	Co-Encadrant	R&D Direction innovation Cevital
ZAOUI Abdelhalim	Co-Encadrant	Prof ENSTA

Alger, le 27 / 06 / 24

Année universitaire 2023 – 2024

Dédicace

Je dédié ce travail

A mes parents

Pour tous les sacrifices que j'ai faits, car sans eux je ne serais rien, et c'est grâce à eux que je m'efforce d'exceller et de les rendre fiers de moi un jour. Mes bons mentors qui m'ont soutenue à chaque étape de mon parcours. Votre amour inconditionnel et vos encouragements ont été ma force motrice.

A mes deux sœurs

Pour leur soutien inconditionnel et leurs encouragements qui ont été d'une valeur inestimable pour moi.

A mes amis

Compagnons de mes joies et de mes peines. Votre présence précieuse et vos sourires ont illuminé les moments les plus difficiles. Merci d'avoir été à mes côtés tout au long de cette aventure.

A mes enseignants

Dont l'influence et le soutien ont été fondamentaux dans mon parcours.

Akram.

Dédicace

Je dédié ce travail

A mes parents

Dont l'amour, le soutien et les sacrifices ont illuminé chaque instant de ma vie académique. Votre foi inébranlable en moi a été la force motrice derrière chaque défi que j'ai relevé.

A mes frères

Chaque moment passé avec vous a renforcé ma profonde estime pour l'aide que vous m'avez apportée. Vous avez été là pour me soutenir, me réconforter et me pousser à relever les défis.

A mes amis

Compagnons de mes joies et de mes peines. Votre présence précieuse et vos sourires ont illuminé les moments les plus difficiles. Merci d'avoir été à mes côtés tout au long de cette aventure

A mes enseignants

Dont l'influence et le soutien ont été fondamentaux dans mon parcours.

Hicham.

Remerciements

Tout d’abord, Nous tenons à remercier « Allah » le tout puissant de nous avoir donné la possibilité de réaliser ce projet, d’arriver à nos souhaits et d’atteindre nos objectifs. Nous aimerons dans ces quelques lignes remercier toutes les personnes qui d’une manière ou d’une autre, ont contribué au bon déroulement de notre travail, tout au niveau humain qu’au niveau scientifique.

Nous tenons tout d’abord à remercier très fortement nos encadrants **Mme. SAADI Hadjer**, **M. BELLOULA Abdelmalek** et **M. ZAOUI Abdelhalim**, on a pu bénéficier à la fois de leurs compétences scientifiques, et de leurs grandes disponibilités, tant pour résoudre les difficultés rencontrées lors de notre réalisation, de répondre à nos questions.

Nos remerciements s’adressent également à tous membres de Jury, qui ont accepté de nous honorer de leur présence et de juger notre travail ; Merci.

Nos remerciements vont également à nos familles et amis qui nous ont soutenus mentalement et physiquement et en nous donnant le courage et l’énergie, et le rire quand nous en avons le plus besoin merci beaucoup.

ملخص

يركز هذا المشروع على تصميم وتنفيذ بوابة لإنترنت الأشياء الصناعية .(IIoT) الهدف الرئيسي هو إنشاء بنية تحتية قادرة على جمع ومعالجة ونقل البيانات من مختلف أجهزة الاستشعار الصناعية. وتتضمن البوابة، التي تعتمد على وحدات تحكم دقيقة متطورة، تقنيات اتصالات لاسلكية لضمان نقل البيانات بشكل موثوق. يهدف المشروع إلى تحسين المراقبة والكفاءة التشغيلية في البيئات الصناعية باستخدام حلول إنترنت الأشياء المبتكرة لتحسين إدارة الموارد والعمليات.

الكلمات المفتاحية: أنترنت الأشياء، الإنترنت الصناعي للأشياء، البوابة.

Résumé

Ce projet se concentre sur la conception et la mise en œuvre d'une passerelle pour l'internet industriel des objets (IIoT). L'objectif principal est de créer une infrastructure capable de collecter, traiter et transmettre des données provenant de divers capteurs industriels. La passerelle, basée sur des microcontrôleurs avancés, intègre des technologies de communication sans fil pour assurer une transmission fiable des données. Le projet vise à améliorer la surveillance et l'efficacité opérationnelle dans les environnements industriels en utilisant des solutions innovantes d'IoT pour une gestion optimale des ressources et des processus.

Mots clé : internet des objets, internet industriel des objets, passerelle.

Abstract

This project focuses on the design and implementation of a gateway for the Industrial Internet of Things (IIoT). The main objective is to create an infrastructure capable of collecting, processing and transmitting data from various industrial sensors. The gateway, based on advanced microcontrollers, integrates wireless communication technologies to ensure reliable data transmission. The project aims to improve monitoring and operational efficiency in industrial environments by using innovative IoT solutions for optimal management of resources and processes.

Keywords: internet of things, industrial internet of things, Gateway

Table des matières

Liste des figures

Chapitre 1 : Généralités sur l'internet des objets

Chapitre 2 : Conception de la passerelle pour l'internet industriel des objets

Chapitre 3 : Résultats Expérimentaux et interprétations

Liste des tableaux

List des abréviations

A-MPDU Aggregate MAC Protocol Data Unit

A-MSDU Aggregate MAC Service Data Unit

ACK Acknowledgement

ADC Analogue to Digital Converter

API Application Programming Interface

BLE Bluetooth Low Energy

CAN Controller Area Network

CE Chip Enable

CoAP Constrained Application Protocol

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CS Chip Select

CSN Chip Select Not

DAC Digital to Analogue Converter

DHCP Dynamic Host Configuration Protocol

ESB Enhanced ShockBurst

FIFO First In First Out

GDMA General DMA

GPIO General-Purpose Input/Output

HAL Hardware Abstraction Layer

HTTP Hypertext Transfer Protocol

I2C Inter-Integrated Circuit

I2S Integrated Interchip Sound

IDE Integrated Development Environment

IoT Internet of things

IIoT Industrial Internet of Things

IP Internet Protocol

ISM Industrial, Scientific, and Medical

ISO International Organization for Standardization

JTAG Joint Test Action Group

JSON JavaScript Object Notation

LNA Low Noise Amplifier

LTE Long-Term Evolution

LTE-M LTE for Machine-Type Communications

MAC Media Access Control

MCPWM Motor Control Pulse Width Modulation

MISO Master In Slave Out

MOSI Master Out Slave In

MQTT Message Queuing Telemetry Transport

NB-IoT Narrowband IoT

OTG On The Go

PA Power Amplifier

PCF Point Coordination Function

PID Proportional-Integral-Derivative Controller

PLL Phase-Locked Loop

PSRAM Pseudo Static RAM

PWM Pulse Width Modulation

RPD Received Power Detector

ROM Read-Only Memory

RTC Real Time Clock

RTOS Real-Time Operating System

SCL Serial Clock Line

SDA Serial Data Line

SDIO Secure Digital Input Output

SPI Serial Peripheral Interface

SRAM Static Random Access Memory

SSID Service Set Identifier

SWD Serial Wire Debug

TCP Transmission Control Protocol

TWAI Two-Wire Automotive Interface

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

USART Universal Synchronous/Asynchronous Receiver-Transmitter

USB Universal Serial Bus

Wi-Fi Wireless Fidelity

XMPP Extensible Messaging and Presence Protocol

Introduction générale

Introduction générale

L'idée de connecter des objets physiques à Internet remonte aux années 1980 et 1990, mais ce n'est qu'au début des années 2000 que le terme "Internet des Objets" (IoT) a été popularisé par Kevin Ashton, cofondateur du MIT Auto-ID Center. Ashton voyait dans l'IoT un moyen de permettre aux machines de percevoir le monde directement sans intervention humaine, en utilisant des capteurs et des réseaux pour collecter et partager des données [1]. Au fil des années, l'avancée des technologies sans fil, des capteurs et des capacités de traitement a permis de concrétiser cette vision, rendant l'IoT accessible et pertinent dans de nombreux domaines [2].

Dans un monde de plus en plus connecté, l'IoT révolutionne les processus industriels et la gestion des données en temps réel. Cependant, cette croissance rapide entraîne une complexité croissante des réseaux, où différents appareils utilisent diverses technologies de communication et formats de données. Cette complexité rend essentielle la création de passerelles intelligentes pour assurer une communication fluide et sécurisée entre les dispositifs IoT et les systèmes backend.

Pour relever ces défis, il est essentiel de concevoir des passerelles IoT capables d'agir comme des interfaces intelligentes entre les différents réseaux et systèmes. Ces passerelles facilitent la communication transparente et sécurisée entre les appareils IoT, les plateformes cloud et les applications, en assurant une intégration et une interopérabilité efficaces. En intégrant les données provenant de multiples sources, les passerelles IoT contribuent à rationaliser les échanges d'informations au sein de l'écosystème IoT, améliorant ainsi l'efficacité opérationnelle et stimulant l'innovation dans les secteurs industriels et au-delà [3].

Le projet vise à concevoir une passerelle capable de gérer la communication entre les différents éléments du réseau IIoT de manière fiable et efficace. L'objectif principal est d'assurer une synchronisation et un transfert de données sans perte, essentiels pour minimiser les temps d'arrêt et maximiser la productivité des processus industriels. En garantissant une communication fluide et continue, nous visons à améliorer la performance globale des systèmes IIoT déployés dans les environnements industriels.

Un autre objectif clé du projet est l'intégration de technologies sécurisées. La passerelle doit incorporer des protocoles de sécurité robustes pour protéger les données transmises contre les accès non autorisés et garantir l'intégrité des informations échangées. Étant donné que les

données industrielles peuvent être sensibles et stratégiques, la sécurité des communications est un aspect critique du projet.

La flexibilité et l'évolutivité de la passerelle sont également des priorités. Nous visons à concevoir une architecture qui permet l'ajout facile de nouveaux nœuds de capteurs et l'intégration de nouvelles fonctionnalités sans nécessiter de modifications matérielles significatives. Cette flexibilité est essentielle pour répondre aux besoins évolutifs des environnements industriels en constante évolution.

L'interopérabilité et l'intégration sont des objectifs majeurs du projet. La passerelle doit assurer une communication transparente entre les appareils IoT, les plateformes cloud et les applications, intégrant les données provenant de multiples sources pour améliorer l'efficacité opérationnelle. En créant un écosystème harmonieux, nous visons à permettre aux différents dispositifs et systèmes de travailler ensemble de manière fluide.

La structure du projet est divisée en trois chapitres. Le premier chapitre présente les généralités sur l'internet des objets (IoT), en détaillant les principaux éléments de son architecture, de ses protocoles de communication et de ses fonctionnalités indispensables. Le deuxième chapitre est consacré à la conception de la passerelle pour l'internet industriel des objets (IIoT), en décrivant les principaux composants de notre système et les étapes de configuration de la passerelle et des nœuds de capteurs. Le troisième chapitre est consacré à l'évaluation de la passerelle avec des tests, puis optimisation pour voir le bon fonctionnement de notre passerelle.

Chapitre I

Généralités sur l'internet des objets

1.1 Introduction

L'Internet des objets IdO ou IoT pour « internet of things » représente une révolution technologique qui transforme notre manière d'interagir avec le monde physique en intégrant des dispositifs connectés intelligents à notre environnement numérique. Cette évolution va bien au-delà des simples appareils intelligents, en élargissant notre capacité à collecter, analyser et utiliser les données générées par ces dispositifs pour améliorer notre quotidien et optimiser les processus industriels.

L'IoT est devenu indispensable dans notre vie, avec l'industrie et l'électronique migrantes vers l'Internet des objets. Des avancées comme l'Industrie 4.0, les robots médicaux, les véhicules autonomes, la domotique, et bien d'autres démontrent que l'IoT est un domaine crucial pour les ingénieurs. Il est essentiel non seulement pour l'avenir de l'industrie, mais également pour la vie quotidienne de chacun.

1.2 Internet des objets

1.2.1 Définition

L'Internet des objets (IoT) est un concept qui connecte des objets physiques liés à une électronique embarquée, menus de logiciels, de capteurs et de connectivité pour permettre l'échange de données avec des fabricants, des opérateurs et/ou d'autres appareils connectés. L'IoT permet de détecter et de contrôler des objets à distance via l'infrastructure réseau existante [4].

L'IoT représente l'évolution de l'internet vers un réseau comprenant des dispositifs de différents types et tailles. L'objectif principal de l'IoT est de permettre aux objets de se connecter à n'importe qui ou n'importe quoi, à n'importe quel moment et à n'importe quel endroit, de préférence via n'importe quel réseau ou chemin de service disponible [5], comme est illustré dans la figure 1.1.

L'Internet des objets (IoT) repose sur trois composantes principales : les objets physiques, les réseaux de communications et le cloud computing. Les objets physiques sont des dispositifs équipés de capteurs, de technologies de connectivité et d'une intelligence intégrée. Les réseaux de communications transportent les données générées par ces objets. Le cloud computing fournit les outils nécessaires pour le stockage, la corrélation et l'analyse des données, permettant ainsi de prendre des décisions et d'interagir avec les objets physiques [6].

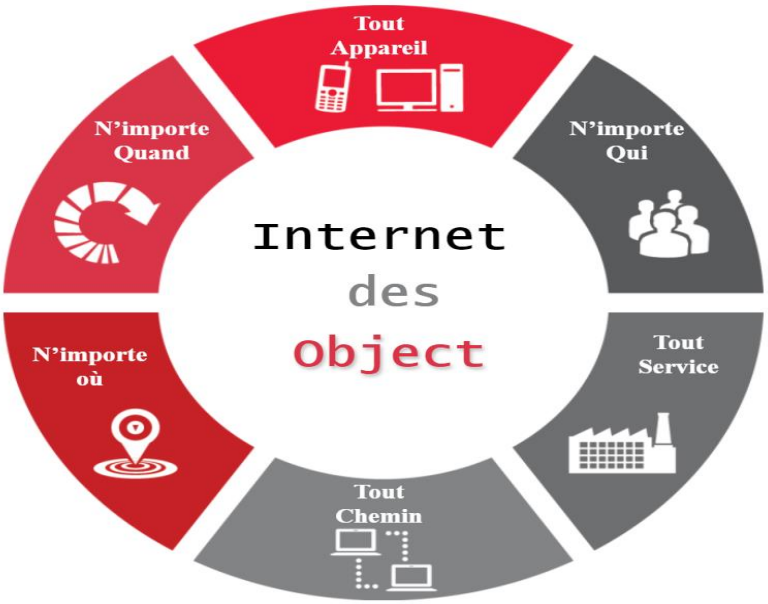

Figure 1. 1 **:** Connectivité de l'IoT [5]

1.2.2 Objet connecté

Après avoir défini l'Internet des objets, il est essentiel de comprendre le rôle des objets connectés dans cet écosystème. Un objet connecté, également appelé objet intelligent ou dispositif connecté, est un élément clé de l'IoT. Ces dispositifs sont équipés de capteurs et/ou d'actionneurs pour accomplir des fonctions spécifiques et ont la capacité de communiquer avec d'autres équipements. Ils s'intègrent dans une infrastructure plus large qui assure le transport, le stockage, le traitement et l'accès aux données générées, que ce soit par les utilisateurs ou par d'autres systèmes [7].

La figure 1.2 illustre l'interconnexion essentielle entre le monde physique et le réseau informatique.

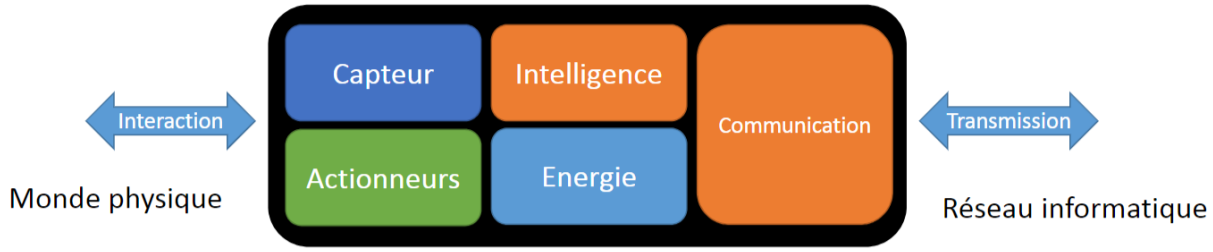

Figure 1. 2 : Interconnexion entre le monde physique et le réseau informatique [8]

1.2.3 Avantages de l'internet des objets

L'Internet des objets présente de nombreux avantages qui en font une technologie révolutionnaire et indispensable dans divers domaines :

— **Amélioration de l'efficacité opérationnelle :** Les dispositifs IoT permettent une surveillance et une gestion en temps réel des opérations, réduisant ainsi les temps d'arrêt et les coûts de maintenance. En automatisant les processus, l'IoT améliore la productivité et l'efficacité des entreprises.

— **Optimisation des ressources :** Grâce aux capteurs IoT, il est possible de surveiller et de contrôler l'utilisation des ressources, comme l'énergie et l'eau, de manière plus efficace, réduisant ainsi les gaspillages et les coûts.

— **Amélioration de la qualité de vie :** L'IoT contribue à créer des environnements de vie plus intelligents et plus confortables. Par exemple, les maisons intelligentes peuvent ajuster automatiquement les systèmes de chauffage, d'éclairage et de sécurité en fonction des préférences des occupants.

— **Prise de décision basée sur les données :** Les objets connectés collectent une grande quantité de données qui, une fois analysées, fournissent des informations précieuses pour la prise de décision. Les entreprises peuvent utiliser ces données pour comprendre les tendances du marché, améliorer les produits et services, et répondre de manière proactive aux besoins des clients.

— **Sécurité et surveillance améliorées :** L'IoT offre des solutions avancées pour la sécurité et la surveillance, comme les systèmes de surveillance vidéo intelligents, les alarmes connectées et les dispositifs de suivi en temps réel, augmentant ainsi la sécurité des biens et des personnes.

— **Innovation et nouveaux modèles économiques :** L'IoT ouvre la voie à de nouveaux modèles économiques et à des innovations dans divers secteurs, tels que la santé, l'industrie, la logistique et les transports. Par exemple, les dispositifs médicaux connectés permettent une surveillance continue des patients à distance, améliorant ainsi les soins de santé.

1.2.4 Domaines d'application de l'internet des objets

Le potentiel transformateur de l'internet des objets est clairement mis en évidence par ces bénéfices. À la différence de concepts abstraits ou de théories futuristes, l'IoT trouve des applications concrètes et diversifiées dans notre vie quotidienne. Sa répercussion est déjà visible dans de nombreux domaines, modifiant nos modes de vie et de travail. L'internet des objets est appliqué dans plusieurs secteurs [5], comme illustré dans la figure 1.3.

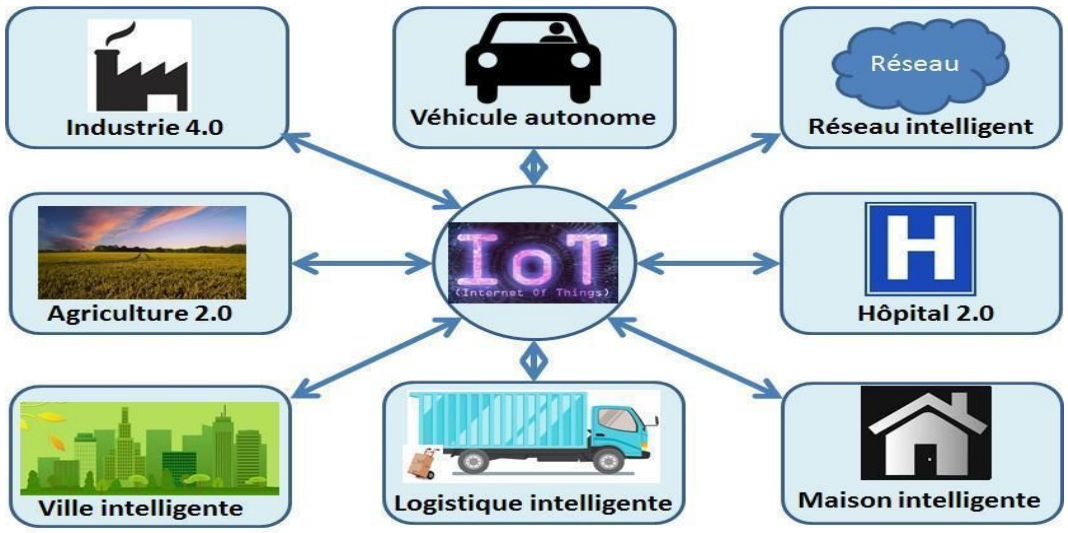

Figure 1. 3 : Exemples d'application de l'internet des objets

— **Industrie 4.0 :** L'Industrie 4.0 intègre des technologies avancées comme l'IoT pour connecter les équipements de production, permettant une automatisation accrue, une maintenance prédictive et une optimisation des processus industriels.

— **Agriculture 2.0 :** L'Agriculture 2.0, ou agriculture de précision, utilise des capteurs connectés pour surveiller les conditions environnementales, optimiser l'utilisation des ressources et améliorer les rendements agricoles de manière plus efficace et durable.

— **Villes Intelligentes :** Les Villes Intelligentes intègrent des solutions IoT pour améliorer la gestion des services urbains, la sécurité publique, la durabilité environnementale et la qualité de vie des citoyens en utilisant des capteurs connectés pour surveiller et optimiser les infrastructures urbaines.

— **Logistique intelligente** : La Logistique Intelligente utilise des technologies IoT pour suivre en temps réel les marchandises, optimiser les itinéraires, gérer les stocks de manière automatisée et améliorer l'efficacité opérationnelle de la chaîne logistique.

— **Maison Intelligente :** La Maison Intelligente intègre des appareils connectés pour automatiser les tâches domestiques, améliorer la sécurité, optimiser la consommation d'énergie et offrir un confort personnalisé aux habitants.

— **Hôpital 2.0 :** L'Hôpital 2.0 utilise des solutions IoT pour surveiller les patients à distance, optimiser les processus hospitaliers, réduire les erreurs médicales et offrir des soins de santé plus personnalisés en utilisant des dispositifs médicaux connectés et des systèmes de surveillance intelligents.

— **Réseaux Autonomes :** Les Réseaux Autonomes exploitent l'IoT pour créer des réseaux capables de s'auto-configurer, de s'auto-réparer et de s'auto-optimiser sans intervention humaine, améliorant ainsi l'efficacité et la fiabilité des communications.

— **Véhicule autonome :** Les Véhicules Autonomes intègrent des technologies IoT pour collecter et analyser des données en temps réel sur la route, le trafic, l'environnement et les conditions de conduite, permettant aux véhicules de se déplacer de manière autonome en utilisant des systèmes informatiques avancés.

1.3 Architecture de L'IoT

L'architecture IoT est un cadre qui définit les composants physiques, l'organisation fonctionnelle et la configuration du réseau, les procédures opérationnelles et les formats de données à utiliser pour collecter, stocker et traiter les données des objets générées par les capteurs, et pour exécuter les commandes envoyées via une application utilisateur [9].

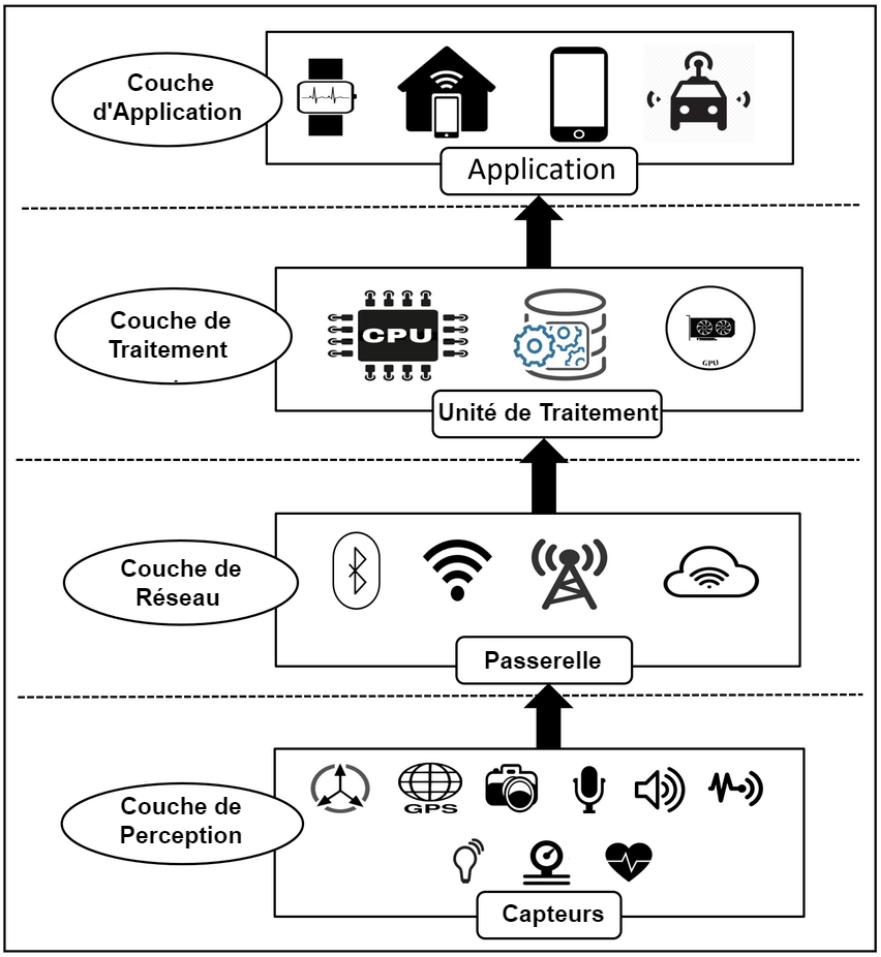

Figure 1. 4 : Architecture de l'IoT [10]

1.3.1 Couche de perception

La couche de perception des systèmes IoT est responsable de convertir des signaux analogiques en données numériques, et vice versa, en utilisant des capteurs et des actionneurs pour la collecte et le traitement des données. Cette couche, caractérisée par de multiples capteurs, est interconnectée par des hubs de capteurs qui servent d'interface centrale pour l'agrégation et la transmission des données à l'unité de traitement de l'appareil. Les hubs de capteurs utilisent des protocoles de communication tels que le bus I2C Inter Integrated Circuit ou l'interface périphérique série (SPI) pour faciliter le flux de données entre les capteurs et les applications. Ces protocoles établissent un canal de communication pour une collecte efficace des données des capteurs [10].

1.3.2 Couche réseau

La couche réseau assure la transmission des données collectées par la couche de perception vers les unités de traitement centralisées ou le cloud. Elle est intégrée dans les appareils IoT en utilisant différentes technologies de communication. Grâce à ces derniers, cette couche garantit une connexion fiable et sécurisée entre les dispositifs IoT [10].

Les différentes technologies sont utilisées pour les communications entre les appareils et les services cloud ou les passerelles :

— **WiFi :** Le Wi-Fi, abréviation de Wireless Fidelity, est une technologie de réseau sans fil basée sur les normes IEEE 802.11. Elle est largement utilisée avec divers appareils, notamment les ordinateurs, les tablettes, les smartphones et même les véhicules tels que les voitures, les bus et les drones.

— **Zigbee et Z-wave :** Il s'agit de deux protocoles de transmission sans fil conçus spécifiquement pour la domotique, offrant une portée moyenne de 100 mètres et une bande passante limitée pour le transfert de petites quantités de données. Les appareils Zigbee sont énergivores, conçus pour les échanges de données à faible débit et adaptés aux capteurs alimentés par pile ou batterie qui nécessitent une consommation d'énergie réduite.

— **Bluetooth :** Un protocole standard de transfert de données sans fil a été créé en 1994 par l'entreprise suédoise Ericsson. Sa bande passante limitée lui permet de transmettre une petite quantité de données sur de courtes distances, mais sa faible consommation d'énergie le rend pratique.

— **Réseaux cellulaires :** Les technologies les plus répandues à l'heure actuelle sont la 5G et la 4G, qui garantissent un transfert de données fiable et une couverture quasi mondiale. Il y a également deux normes cellulaires spécifiquement conçues pour les objets IoT. LTE-M (évolution à long terme des machines) offre aux dispositifs la possibilité de communiquer directement avec le cloud et d'échanger de grandes quantités de données. Les canaux basse fréquence utilisés dans NB-IoT ou Narrow Band IoT permettent d'envoyer de petits paquets de données.

Il y a également d'autres technologies qui facilitent la connexion.

1.3.3 Couche de traitement

Les données provenant de la couche précédente sont accumulées, stockées et traitées par la couche de traitement. En général, toutes ces tâches sont effectuées à l'aide de plateformes IoT et se divisent en deux étapes principales.

— **Étape d'accumulation des données :** Les données en temps réel sont capturées via une API et stockées pour répondre aux besoins des applications non temps réel. Cette étape agit comme une interface entre la génération de données basée sur les événements et la consommation de données basée sur les requêtes. Elle détermine la pertinence des données pour les besoins de l'entreprise et les stocke dans diverses solutions de stockage, y compris des « data links » capables de gérer des données non structurées comme des images et des flux vidéo. L'objectif principal est de trier et de stocker efficacement une grande quantité de données diverses.

— **Étape d'abstraction des données :** Cette étape finalise la préparation des données pour les rendre utilisables par les applications grand public. Elle implique la combinaison de données provenant de différentes sources, la normalisation de différents formats de données, et l'agrégation des données en un seul endroit ou leur rendu accessible de manière virtualisée, indépendamment de leur emplacement. De plus, les données collectées au niveau de la couche application sont reformatées pour être transmises au niveau physique afin que les appareils puissent les interpréter. Globalement, les étapes d'accumulation et d'abstraction des données simplifient la complexité matérielle, améliorant ainsi l'interopérabilité des appareils intelligents.

1.3.4 Couche d'application

La couche d'application met en œuvre et expose les résultats du traitement des données pour exécuter une variété d'applications provenant des appareils IoT. Cette couche, axée sur

l'utilisateur, exécute différentes tâches pour les utilisateurs. Les applications IoT sont diverses, couvrant des domaines tels que le transport intelligent, la domotique, les soins personnels, la santé, etc.

1.4 Passerelle

Après avoir passé en revue les différentes couches de l'architecture de l'IoT, qui repose sur une infrastructure réseau robuste et hétérogène pour supporter le volume massif de données générées par les capteurs. Avec le développement des applications IoT avancées, de multiples réseaux utilisant diverses technologies et protocoles doivent interagir de manière fluide.

Dans ce contexte complexe, les passerelles IoT facilitent la communication entre les différents réseaux et en assurant une interopérabilité efficace. Agissant comme des points d'accès sans fil, elles permettent une connectivité transparente et sécurisée entre les appareils IoT et le cloud. En intégrant les données des capteurs, les passerelles contribuent à rationaliser les échanges d'informations au sein de l'écosystème IoT, améliorant ainsi l'efficacité opérationnelle et stimulant l'innovation dans les secteurs industriels. Une passerelle IoT est essentiellement un hub central intelligent pour les appareils IoT, les plateformes backend (données, appareils et gestion des abonnés) et les appareils finaux (capteurs, actionneurs ou dispositifs plus complexes) se connectent à la passerelle IoT via des technologies de communication [11].

La mise en place de passerelles assure la compatibilité entre les divers protocoles et services Internet. Elles agissent comme des traducteurs, permettant aux systèmes de communiquer et d'échanger des données de manière cohérente. Elles ont une importance capitale pour prévenir les problèmes de compatibilité qui pourraient impacter l'automatisation de l'installation.

1.4.1 Protocoles de communication

Plusieurs normes IoT sont suggérées afin de simplifier et de faciliter les activités des programmeurs d'applications et des prestataires de services. L'objectif de l'IoT est d'établir une communication entre chaque système et tous les autres en utilisant des protocoles communs. Voici un aperçu des principaux protocoles de communication employés dans l'Internet des objets :

— **MQTT :** Le protocole MQTT est un protocole de messagerie de couche d'application basé sur la publication/abonnement selon la norme ISO (Organisation internationale de normalisation), il fonctionne sur la famille de protocoles TCP/IP. Le protocole MQTT est un protocole de transmission d'informations très simple et très léger, conçu pour être utilisé avec des équipements restreints, une bande passante faible, une latence élevée et un temps d'attente

élevé. Faible bande passante, à forte latence ou sur des réseaux peu fiables. MQTT vise à minimiser les exigences en matière de bande passante et de ressources d'équipement, tout en garantissant la fiabilité et un certain degré de livraison. Tout en assurant la fiabilité et un certain degré de garantie de livraison. Est idéal pour les appareils IoT qui nécessitent de la bande passante et des batteries [12].

— **CoAP :** est un protocole de couche d'application conçu pour les applications IoT. Il définit un protocole de transfert Web basé sur les fonctionnalités HTTP, mais qui est intégré par défaut à UDP (et non à TCP), ce qui le rend plus adapté aux applications IoT. De plus, CoAP modifie certaines fonctionnalités HTTP afin de répondre aux exigences de l'IoT, telles que la réduction de la consommation d'énergie et l'opération en présence de liens peu fiables et bruyants.

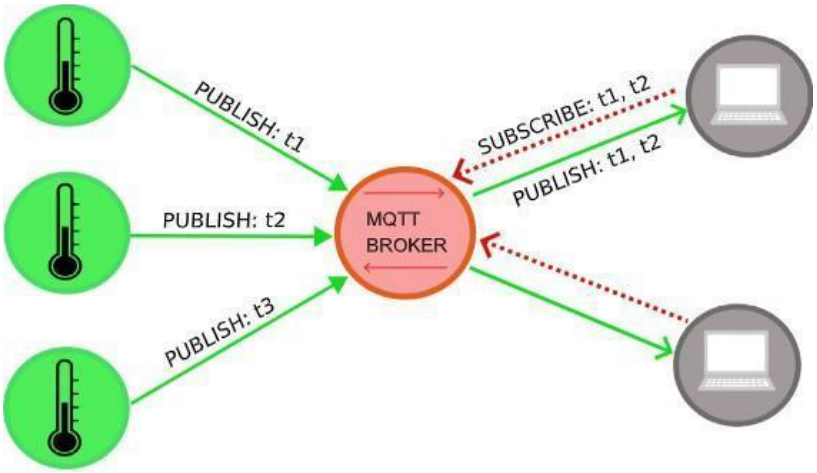

Figure 1. 5 : Exemple de réseaux MQTT [13]

— **XMPP :** est un protocole de communication instantanée (IM) qui permet des échanges multipartites, des appels vocaux et vidéo, ainsi que la présence à distance. Il permet de communiquer instantanément en utilisant des messages instantanés sur Internet, peu importe le système d'exploitation. L'authentification, le contrôle d'accès, la confidentialité mesurée, le chiffrement hop-by-hop et la compatibilité avec d'autres protocoles sont des fonctionnalités proposées par XMPP.

— **HTTP :** est un protocole de communication de couche d'application largement utilisé dans l'Internet des objets (IoT). Initialement conçu pour le transfert de documents hypertextes, il est devenu essentiel pour la communication entre les systèmes IoT en raison de sa simplicité et de sa flexibilité.

Tableau 1. 1: Comparaison des protocoles IoT typiques [12]

Standard	Modèle	Taille du paquet	Protocole de la couche de transport
MQTT	Publication/Abonnement	Très petit	TCP
CoAP	Requête/Réponse	Petit	UDP
XMPP	Basé sur XML, orienté messagerie	Petit à moyen	TCP
HTTP	Requête/Réponse	Très Grand	TCP

1.5 Conclusion

L'Internet des Objets représente une évolution significative de l'infrastructure numérique mondiale, connectant des objets physiques à travers des réseaux intelligents et des plateformes cloud. Ce rapprochement offre non seulement une meilleure surveillance et contrôle des environnements physiques, mais il bouleverse également plusieurs domaines tels que l'industrie, l'agriculture, la santé et les infrastructures villes. En rendant l'IoT plus facile à automatiser, à optimiser les ressources et à améliorer la qualité de vie, il ouvre la voie à de nouveaux modèles économiques et à des innovations perpétuelles.

En prenant en compte ces progrès, il est primordial de garantir une connexion efficace et sécurisée entre les dispositifs IoT. La création d'une passerelle IoT est alors nécessaire, jouant un rôle essentiel dans la gestion et l'intégration des données entre les périphériques IoT et les systèmes cloud. Le chapitre suivant examinera en profondeur la conception d'une passerelle IIoT, en mettant en évidence les principaux éléments de son architecture, de ses protocoles de communication et de ses fonctionnalités indispensables.

Chapitre II

Conception de la passerelle pour l'internet industriel des objets

2.1 Introduction

Dans le chapitre précédent, nous avons donné un aperçu général de l'Internet des objets (IoT). Ce chapitre se concentre sur notre conception spécifique des avancées techniques et technologiques dans le domaine de l'internet industriel des objets (IIoT). Notre objectif est de développer une passerelle IIoT fonctionnelle ainsi que des nœuds de capteurs correspondants.

Nous commencerons par décrire les principaux composants de notre système et justifierons nos choix pour chacun d'entre eux. Ensuite, nous nous pencherons sur les processus d'assemblage et leur importance dans nos efforts de développement. Nous aborderons également les éléments externes faisant partie intégrante de la conception de la passerelle, y compris les dispositifs et périphériques nécessaires à la communication. En outre, nous examinerons l'aspect logiciel, en détaillant la phase de programmation et la conception des fonctionnalités de la passerelle IIoT. Nous fournirons par une description complète du système que nous avons conçu, en mettant l'accent sur l'intégration des composants matériels et logiciels, ainsi que sur les décisions technologiques prises pour assurer une communication efficace et fiable au sein de notre passerelle IIoT et de ses nœuds de capteurs associés.

2.2 Description du système

Notre système est conçu pour faciliter la collecte et la transmission efficaces de données dans un environnement d'internet industriel des objets (IIoT). Au cœur du système, la passerelle joue le rôle de hub central, construite pour gérer et organiser la communication au sein du réseau. Des nœuds de capteurs répartis sont stratégiquement placés pour collecter des données pertinentes à partir de diverses sources. La communication entre la passerelle et les nœuds de capteurs s'effectue à l'aide de la technologie RF, ce qui garantit un transfert de données fiable et rapide. Une fois les données collectées par la passerelle, elles sont publiées dans le Cloud, ce qui permet une surveillance et une analyse en temps réel. Pour mieux comprendre la sélection des composants de notre système, nous allons examiner en détail les microcontrôleurs utilisés.

Pour mieux comprendre la sélection des composants de notre système, nous allons examiner en détail les microcontrôleurs utilisés. La figure 2.1 illustre la présentation générale du système ou le schéma de principe de passerelle que nous avons conçu et réalisé.

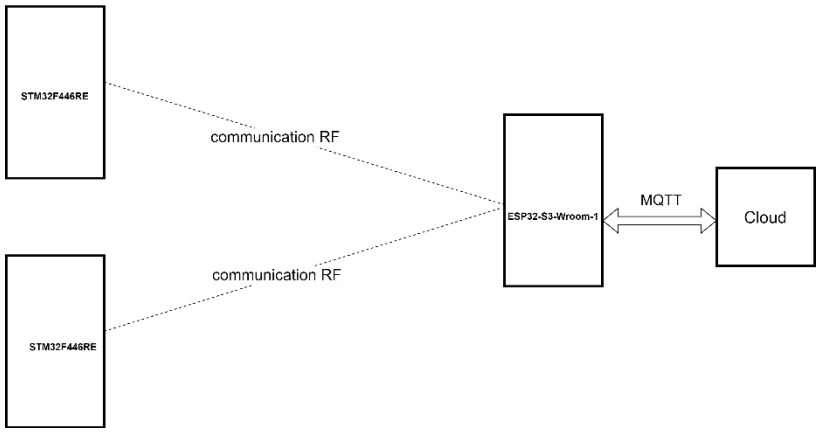

Figure 2. 1 : Synoptique global du système conçu

2.3 Microcontrôleurs ESP32

L'ESP32 est un microcontrôleur conçu par Espressif System, une entreprise chinoise basée à Shanghai. Il se présente comme une solution de mise en réseau Wi-Fi autonome, faisant office de passerelle entre les microcontrôleurs existants et le Wi-Fi. L'ESP32 est également capable d'exécuter des applications autonomes. La production en volume de l'ESP32 n'a démarré qu'à la fin de l'année 2016, ce qui en fait une nouvelle entrée dans la gamme des processeurs [14].

Lorsqu'on examine un nouvel appareil électronique, on s'intéresse toujours à ses spécifications, soit l'ensemble des caractéristiques décrites par le fabricant. L'ESP32 est un processeur double cœur exécutant les instructions Xtensa LX6. Les cœurs sont appelés "PRO_CPU" et "APP_CPU" [14].

La question de déterminer combien de temps un ESP32 peut fonctionner sur batterie est intéressante. La consommation du courant est loin d'être constante. Lors de la transmission à pleine puissance, il peut consommer 260 mA, mais en mode de veille prolongée, seulement 20 µA. C'est une différence considérable. Cela signifie que l'autonomie d'un ESP32 sur une réserve de courant fixe n'est pas seulement une fonction du temps, mais aussi de ce qu'il fait pendant ce temps. L'ESP32 est conçu pour être utilisé avec un module de mémoire partenaire, le plus souvent de la mémoire flash [14].

2.3.1 Environnement de développement pour les microcontrôleurs ESP32

Pour les microcontrôleurs ESP32, l'environnement de développement permet d'écrire des applications qui peuvent s'exécuter nativement sur le dispositif. Il est possible de compiler des applications écrites en langage C et de les déployer sur le microcontrôleur à travers un processus

appelé "flashing". Cela signifie que le code compilé est transféré dans la mémoire flash de l'ESP32, permettant ainsi son exécution autonome [14].

L'ESP32 vous permet d'écrire des applications qui peuvent s'exécuter nativement sur l'appareil. Vous pouvez compiler des applications écrites dans le langage de programmation C et les déployer sur le périphérique via un processus appelé flash [14].

Cet environnement de développement comprend généralement un éditeur de code, un compilateur C, des bibliothèques et des outils de débogage pour faciliter la création et le test des applications ESP32. Les développeurs peuvent ainsi tirer parti des capacités Wi-Fi et Bluetooth de l'ESP32 pour créer des solutions IoT innovantes [14].

2.3.2 Choix du microcontrôleur

L'ESP32-S3-WROOM-1-N16R8 est un module de microcontrôleur autour de la série de SoC ESP32-S3 fabriqué par « Espressif Systemes ». Il est basé sur le processeur Xtensa LX7 dual-core 32 bits, cadencé à une fréquence pouvant atteindre 240 MHz. Il offre également des capacités de connectivité sans fil étendues [15].

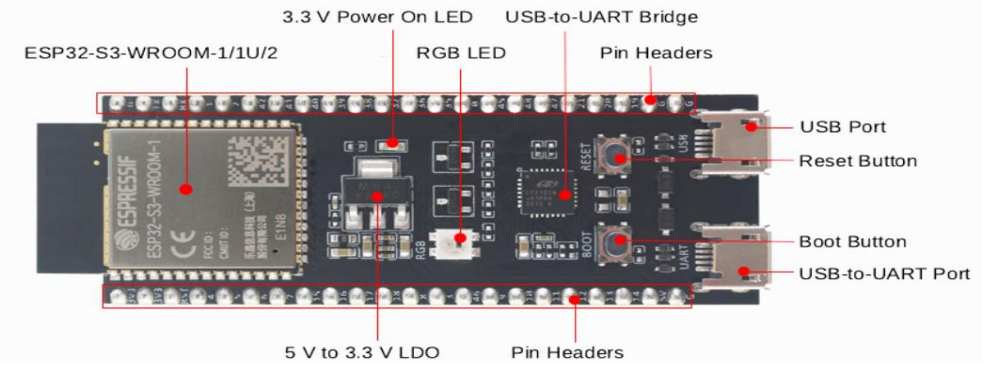

Figure 2. 2 : Module de microcontrôleur ESP32-S3 [16]

Voici quelques-unes de ses caractéristiques principales [15] :

— **Microprocesseur :** Xtensa LX7 double cœur 32 bits à 240 MHz : possibilité d'éteindre le CPU et d'utiliser le coprocesseur basse consommation pour surveiller en continu les périphériques afin de détecter les changements ou les dépassements de seuils.

— **Mémoire intégrée :** 384 Ko de ROM pour le démarrage et les fonctions de base, 512 Ko de SRAM pour les données et les instructions, fonctionnant à une fréquence configurable allant jusqu'à 240 MHz, 16 Ko de SRAM en RTC, 8 Mo de PSRAM.

— **Richesse des interfaces périphériques :** Intègre divers périphériques, dont SPI, LCD, caméra, UART, I2C, I2S, télécommande, compteur d'impulsions, LED PWM, USB

Serial/JTAG, MCPWM, SDIO, GDMA, TWAI® (ISO 11898-1), ADC, capteur tactile et de température, temporisateurs, chiens de garde, et jusqu'à 45 GPIO.

— Il comprend également une interface USB 2.0 On-The-Go (OTG) à pleine vitesse pour permettre la communication USB.

— Tension de fonctionnement/Alimentation 3,0~3,6 V, température ambiante de fonctionnement –40 ~ 65 °C pour la version R8.

— **Wi-Fi :** 802.11 b/g/n avec un débit pouvant atteindre 150 Mbps, prise en charge de l'agrégation A-MPDU et A-MSDU, et intervalle de garde de 0,4 µs.

— Antenne PCB intégrée.

— **Bluetooth :** Bluetooth LE 5.0, Bluetooth Mesh, avec des diffèrent débits, extensions de publicité, ensembles de publicité multiples et algorithme de sélection de canal.

Sur la base des caractéristiques et des critères mentionnés, voici pourquoi nous avons choisi l'ESP32-S3-WROOM-1-N16R8 pour concevoir une passerelle IIoT :

— Le processeur Xtensa LX7 dual-core 32 bits à 240 MHz fournit une puissance suffisante pour la gestion de la connectivité et le traitement des données dans notre passerelle IIoT.

— Le module offre une connectivité Wi-Fi et Bluetooth étendue, facilitant l'intégration avec une variété d'appareils IoT.

— Le module est optimisé pour une faible consommation, avec des modes de veille à 20 µA et une gestion efficace de l'énergie en transmission.

— L'ESP32-S3-WROOM-1 est un module intégrant mémoire flash et antenne, simplifiant l'intégration dans notre passerelle IoT.

— La richesse des interfaces périphériques.

— Espressif Systems fournit une documentation détaillée, des bibliothèques de logiciels et un large support communautaire, facilitant le développement et le dépannage.

— Le module dispose de certifications RF et vertes, assurant la conformité aux normes industrielles et la fiabilité dans diverses conditions de fonctionnement.

— L'ESP32-S3 dispose de GPIO (General-Purpose Input/Output) pouvant fonctionner en mode RTC et réveiller l'appareil lorsqu'il est en veille.

2.4 Microcontrôleurs STM32

Le STM32 est une gamme de microcontrôleurs lancée par ST en 2007, divisée en neuf sous-familles, chacune avec ses propres caractéristiques et basée sur un cœur Cortex-M. Chaque

microcontrôleur intègre un cœur de processeur, de la mémoire vive et flash, une interface de débogage, et divers autres périphériques. La gamme s'étend également pour inclure des dispositifs basse consommation. Les STM32 offrent donc une variété d'options en termes de performances et de consommation, adaptées à de nombreuses applications embarquées grâce à leurs architectures avancées et aux innovations de ST [17].

2.4.1 Environnement de développement pour les microcontrôleurs STM32

La programmation des microcontrôleurs STM32 se fait principalement en C ou C++, des langages appréciés dans le domaine de l'embarqué pour leur efficacité, leur flexibilité et leur portabilité. Pour développer des programmes optimisés, un environnement de développement complet est essentiel [14]. STMicroelectronics, fabricant des STM32, propose une gamme d'outils et de logiciels pour simplifier la programmation :

- **STM32CubeMX :** Outil graphique de configuration permettant de définir les paramètres des périphériques du microcontrôleur (E/S, Timers, UART, etc.) de manière intuitive.
- **STM32Cube IDE :** Environnement de développement intégré (IDE) conçu pour la création, la compilation et le débogage des programmes STM32. Il intègre un éditeur de code, un compilateur, un débogueur et d'autres outils pour assister les développeurs.
- **Bibliothèques et middleware :** STMicroelectronics fournit des bibliothèques pré-écrites pour les tâches courantes (E/S, communication série, interruptions, etc.) et des middlewares pour des fonctionnalités spécifiques (protocoles de communication, gestion réseau, etc.).

En somme, la programmation des STM32 est simplifiée par l'utilisation de C/C++ et d'un ensemble complet d'outils et de logiciels (STM32CubeMX, STM32CubeIDE, bibliothèques, middleware), permettant aux développeurs de concevoir efficacement des applications fiables pour les microcontrôleurs STM32.

2.4.2 Choix de microcontrôleur

Le STM32F446RE est un microcontrôleur de la série STM32F4 de STMicroelectronics, réputé pour ses performances et sa polyvalence. Ce microcontrôleur est basé sur le cœur Arm Cortex-M4, cadencé à une fréquence allant jusqu'à 180 MHz, et intègre une variété de

périphériques qui le rendent idéal pour les applications de capteurs dans les nœuds IoT [18].

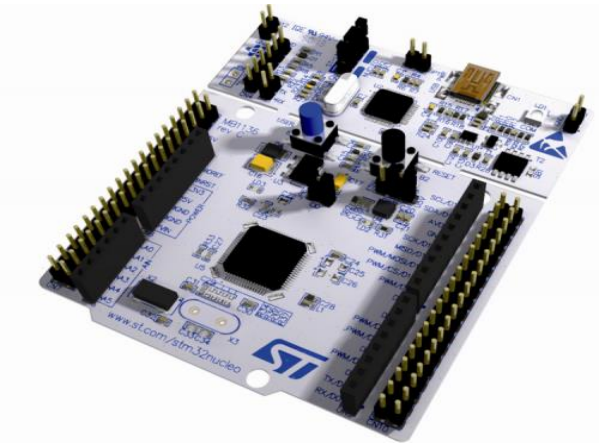

Figure 2. 3 : Microcontrôleur STM32F446RE

Voici les principales caractéristiques du STM32F446RE [18] :

- **Cœur :** Arm Cortex-M4 avec FPU (Floating Point Unit) simple précision, cadencé à 180 MHz.
- **Mémoires :** Mémoire flash jusqu'à 512 Ko, SRAM de 128 Ko, et contrôleur de mémoire externe.
- Jusqu'à 114 pins d'entrées/sorties avec possibilités d'interruption.
- **Alimentation :** Tension de fonctionnement de 1,7 V à 3,6 V.
- **Périphériques de Communication :** Interface Ethernet MAC, ports série (USART, UART, I2C, SPI), ports USB, interface CAN, et contrôleur SDIO.
- **Périphériques Analogiques :** ADC 12 bits, DAC 12 bits, et capteurs de température.
- **Interfaces de Débogage et Programmation :** Interfaces JTAG/SWD pour le débogage et la programmation.

Sur la base des caractéristiques et des critères mentionnés, voici pourquoi nous avons choisi le STM32F446RE pour notre projet de passerelle IoT :

- **Performance et Flexibilité :** Le cœur Arm Cortex-M4 cadencé à 180 MHz offre une bonne performance de traitement pour les applications IoT nécessitant une gestion efficace des données et des tâches de connectivité.
- **Efficacité Énergétique :** Le STM32F446RE est conçu pour une efficacité énergétique, avec des modes basse consommation et une gestion optimisée de l'alimentation.

— **Facilité d'Intégration :** La série STM32 bénéficie d'un vaste écosystème de support avec des bibliothèques logicielles, une documentation complète et une communauté active. Cela simplifie grandement le processus de développement et d'intégration, réduisant ainsi le temps de mise sur le marché.

— **Disponibilité et Coût** Le STM32F446RE est largement disponible et son coût est compétitif, ce qui en fait une option économique pour notre projet. La disponibilité sur le marché assure une continuité dans les approvisionnements, essentielle pour la production à grande échelle.

— **Ressources et Support :** STMicroelectronics fournit des outils de développement comme STM32CubeIDE, STM32CubeMX, et STM32Studio, facilitant le développement et le déploiement de notre projet.

Le choix du STM32F446RE repose sur ses performances élevées, son efficacité énergétique, sa facilité d'intégration, sa disponibilité à un coût abordable, et le soutien étendu fourni par STMicroelectronics. Ces caractéristiques le rendent parfaitement adapté aux exigences de notre projet.

2.5 Capteurs

Pour la conception de notre passerelle, l'acquisition de données est une étape essentielle, nécessitant l'utilisation de divers capteurs. Les capteurs sont des dispositifs qui détectent et mesurent des informations physiques et les convertissent en signaux électriques numérique ou bien analogique interprétables par un microcontrôleur. Dans les systèmes IIoT, permettant de recueillir des données environnementales précises et en temps réel. Parmi les nombreux capteurs disponibles, nous avons choisi d'utiliser le DHT22 et le TCS34725 pour notre projet.

2.5.1 DHT22

Le capteur DHT22 offre une mesure précise de l'humidité et de la température de l'air ambiant grâce à sa combinaison intégrée d'un capteur d'humidité capacitif et d'une thermistance et peut fonctionner sous 3,3V ou 5V. Ce capteur convient parfaitement à vos projets de domotique, de surveillance de l'environnement. Le DHT22 est idéal pour des applications de mesure de température et d'humidité dans des systèmes embarqués, domotique, météo, etc.

Figure 2. 4 : Capteur DHT22

Le capteur possède 3 broches, utilisées comme suit :

— La broche 1 est la broche d'alimentation (5 volts ou 3,3 volts).

— La broche 2 est la broche de donnée.

— La broche 3 est la masse du capteur (GND).

Tableau 2. 1 : Caractéristique de capteur DHT22

Alimentation	3.3 à 5V DC
Consommation max	1.5 mA
Consommation au repos	50 µA
Plage de mesure	Température : -40 à 80 Celsius Humidité : 0 à 100% RH
Précision	Température : ± 0.5 Celsius Humidité : ± 2%RH
Dimension	27 x 59 x 13,5 mm

2.5.2 TCS34725

Le capteur TCS34725, développé par AMS, est un capteur de couleur RGB (Rouge, Vert, Bleu) avec un filtre de blocage des infrarouges (IR) et un convertisseur analogique-numérique (ADC) intégré. Il permet de mesurer la couleur et l'intensité lumineuse d'une source de lumière ambiante, offrant ainsi une grande précision pour les applications nécessitant la détection de couleur.

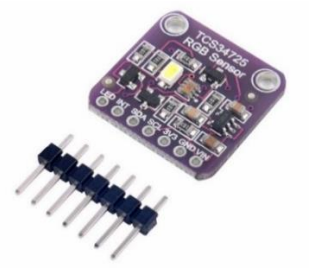

Figure 2. 5 : Capteur TCS34725

Broches principales :

— **VCC :** Alimentation du capteur (3,3V à 5V)

— **GND :** Masse

— **SCL :** Horloge I2C

— **SDA :** Données I2C

— **LED :** Contrôle de l'éclairage intégré (optionnel)

Les caractéristiques de ce capteur présentées dans le tableau suivant :

Tableau 2. 2 : Caractéristiques du capteur TCS34725

Caractéristiques	Description
Type de capteur	Capteur de couleur RGB avec filtre IR
Filtre IR	Capteur de couleur RGB avec filtre IR
Interface de communication	16 bits, haute précision
Adresses I2C	Configurables, permettent l'utilisation de plusieurs capteurs sur le même bus I2C
Éclairage intégré	LED blanche intégrée, activable pour des mesures fiables en conditions de faible luminosité
Applications typiques	Détection de couleur et tri, contrôle d'éclairage ambiant, écrans et affichages, industrie alimentaire et agriculture

2.6 Module nRF24L01+PA/LNA

Le module nRF24L01, développé par « Nordic Semiconductor », est un transmetteur radio fonctionnant sur la bande de fréquences de 2,4 GHz. Il utilise le protocole de communication propriétaire « ShockBurst », permettant à plusieurs transmetteurs d'échanger des données avec adressage, gestion des erreurs de transmission et retransmission automatique en cas de non-réponse du destinataire. Le module fonctionne avec une tension de 1,9 à 3,6 volts, avec des broches logiques qui peuvent supporter jusqu'à 5 volts, il n'est plus nécessaire d'utiliser un traducteur de niveau logique. Il est possible d'ajuster la puissance de sortie. Lors de la transmission, le module ne consomme que 12 mA à sa puissance maximale. En mode veille, sa consommation est de 26 µA et en mode hors tension, de 900 nA, ce qui en fait un choix optimal pour les applications IoT nécessitant une faible consommation d'énergie. Le nRF24L01 est largement utilisé dans les applications IoT pour sa communication sans fil fiable entre les nœuds de capteurs et la passerelle, grâce à son protocole ShockBurst qui assure une transmission robuste des données.

Le module nRF24L01 communique avec le microcontrôleur via l'interface SPI « Serial Peripheral Interface » à 4 broches, offrant un débit de données maximal de 10 Mbps. Tous les paramètres, y compris le canal de fréquence (125 canaux sélectionnables), la puissance de sortie (0 dBm, -6 dBm, -12 dBm ou -18 dBm) et le débit de données (250 kbps, 1 Mbps ou 2 Mbps), peuvent être configurés via cette interface. Le bus SPI utilise le concept d'un maître et d'un esclave.

Le brochage est défini comme suit :

— Vcc (Alimentation) 3V3
— CE (Reset)
— GND (Masse)
— MOSI (Master Output Slave Input)
— MISO (Master Input Slave Output)
— SCK (Serial Clock)
— CS (chip select)

Figure 2. 6 : Module nRF24L01 + PA/LNA

Voici les principales caractéristiques du module nRF24L01 :

Tableau 2.3 : Caractéristiques du module nRF24L01 [19]

Fréquence de fonctionnement	Bande ISM 2.4 GHz
Débit de données Max	2 Mb/s
Courant en mode transmission	11.3 mA
Courant en mode réception	12.3 mA
Courant en mode hors tension	900 nA
Courant en mode veille	26 µA

Tension d'alimentation	1.9 V à 3.6V
Entrées tolérantes	5V
Portée des communications	Jusqu'à 1000m

Sachant que les modules nRF24L01+ fonctionnant dans une plage de tension de 1,9 à 3,6 volts pour l'alimentation. Cependant, de nombreux microcontrôleurs et autres composants du système IoT utilisent une alimentation de 5V à 12V, ce qui crée une incompatibilité directe avec le module nRF24L01+. Pour résoudre ce problème, un adaptateur de dérivation est nécessaire. Cet adaptateur permet de réguler la tension et de faciliter les connexions entre le module nRF24L01+ et les autres composants du système.

2.6.1 Adaptateur pour nRF24L01

L'adaptateur de dérivation nRF24L01 permet d'alimenter le module nRF24L01+ avec une tension de 5V à 12V, en fournissant une tension bien régulée de 3,3V à la prise du module. Il brise également les broches pour un branchement plus facile et dispose d'une LED de mise sous tension pour indiquer l'état de l'alimentation.

Figure 2. 7 : Adaptateur pour le module nRF24L01

L'adaptateur dispose également d'un connecteur mâle 1 × 6 broches pour établir des connexions avec le microcontrôleur. À également un connecteur femelle 2 × 4 broches qui s'accouple avec le module nRF24L01+. Ces broches reflètent les connexions établies sur l'en-tête 1 × 6 et l'en-tête d'alimentation 1 × 2. Il est important d'observer la bonne orientation lors de l'insertion du module dans l'adaptateur. Le module doit s'étendre à l'extrémité de l'adaptateur et non au-dessus.

2.6.2 Principe de fonctionnement du module nRF24L01

Le débit de données, ou taux de signalisation modulé, est un paramètre fondamental dans le fonctionnement du nRF24L01+. Il détermine la vitesse à laquelle les données sont transmises

et reçues par le module. Le nRF24L01+ supporte trois débits de données aériens : 250 kbps, 1 Mbps et 2 Mbps.

- **250 kbps :** Utilisé pour une meilleure sensibilité du récepteur. Idéal pour des applications nécessitant une portée plus étendue.
- **1 Mbps :** Offre un bon équilibre entre portée et consommation d'énergie.
- **2 Mbps :** Permet une consommation de courant moyenne inférieure et réduit la probabilité de collisions de données en vol, bien que la portée soit réduite par rapport aux débits inférieurs.

La sélection du débit de données air est effectuée via le bit « RF_DR » dans le registre « RF_SETUP ». Pour une communication réussie entre un émetteur et un récepteur, les deux doivent être programmés avec le même débit de données air. Ce choix du débit impacte également la consommation d'énergie et la probabilité de collisions, influençant ainsi les performances globales du réseau.

Le module nRF24L01+ transmet et reçoit des données sur une fréquence spécifique, appelée canal. Pour qu'une communication soit établie entre deux modules ou plus, ils doivent être réglés sur le même canal. Ce canal peut être défini sur n'importe quelle fréquence de la bande ISM « Industrial, Scientific, and Medical » de 2,4 GHz, plus précisément, entre 2,400 GHz et 2,525 GHz (2400 à 2525 MHz).

Chaque canal occupe moins de 1 MHz de bande passante, ce qui permet la création de 125 canaux distincts avec un espacement de 1 MHz. En d'autres termes, le nRF24L01+ peut fonctionner sur 125 canaux différents, permettant ainsi la mise en place d'un réseau de 125 modems fonctionnant indépendamment dans un même espace.

Cette flexibilité de canaux offre la possibilité de construire des réseaux complexes avec de nombreux modules sans interférences, assurant ainsi une communication fiable et stable entre les dispositifs IoT.

Nous aborderons à présent les diverses modes de fonctionnement de l'émetteur-récepteur radio nRF24L01+. Le nRF24L01+ intègre une machine d'états qui assure la gestion des transitions entre ses différentes opérations. En utilisant les valeurs des registres programmés par l'utilisateur et des signaux internes, cette machine d'états peut déterminer le mode de fonctionnement adéquat.

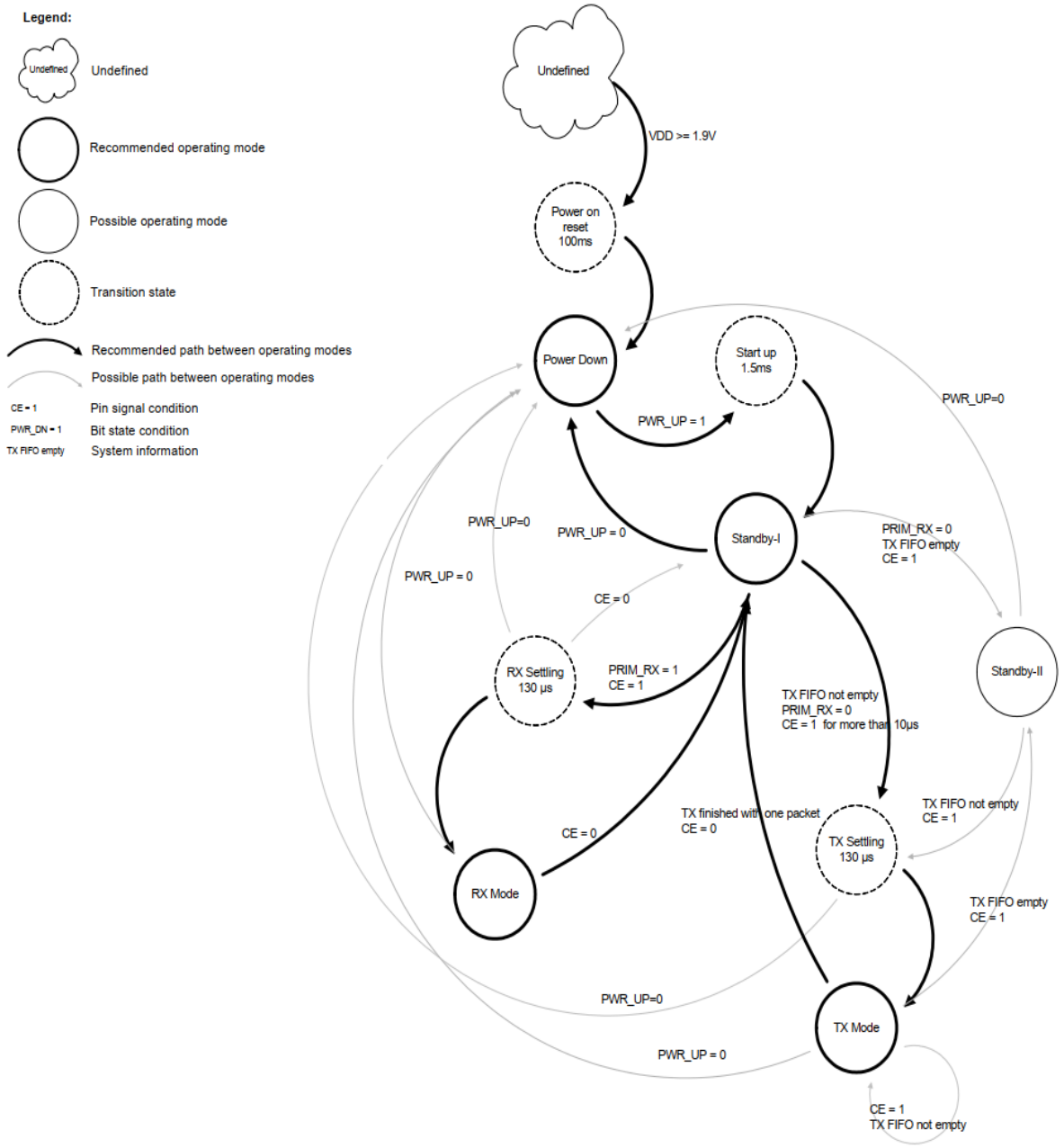

Figure 2. 8 : Diagramme d'état [19]

— **En mode « power down » :** le NRF24L01 est désactivé en utilisant une consommation de courant minimale. Toutes les valeurs de registre disponibles sont maintenues et le SPI reste activé, ce qui permet de modifier la configuration et de charger/décharger des registres de données.

— **En mode « Standby- I » :** est utilisé pour minimiser la consommation moyenne de courant tout en gardant des temps de démarrage courts. Dans ce mode, seule une partie de l'oscillateur à cristal est active. Le passage aux modes actifs n'a lieu que si CE est passé à

l'état haut et lorsque CE est passé à l'état bas, le nRF24L01 revient au mode standby-l à partir des modes TX et RX.

— **En mode « standby- II » :** des buffers d'horloge extra sont actifs et plus de courant est utilisé par rapport au mode standby-l. Le nRF24L01+ entre en mode standby-ll si CE est maintenu haut sur un dispositif PTX avec une TX FIFO vide. Si un nouveau paquet est chargé dans la FIFO TX, la PLL démarre immédiatement et le paquet est transmis après le délai normal de stabilisation de la PLL (130 µs).

— **Mode RX** : est un mode actif dans lequel la radio nRF24L01+ est utilisée comme récepteur. Pour entrer dans ce mode, le nRF24L01+ doit avoir le bit PWR_UP, le bit PRIM_RX et la broche CE placés haut.

- En mode RX, le récepteur démodule les signaux du canal RF et présente en permanence les données démodulées au moteur de protocole en bande de base. Le module de protocole en bande de base recherche en permanence un paquet valide. Si un paquet valide est trouvé (par une adresse correspondante et un CRC valide), la charge utile du paquet est présentée dans un emplacement libre des FIFO RX. Si les FIFO RX sont pleines, le paquet reçu est rejeté.
- Le nRF24L01+ reste en mode RX jusqu'à ce que le MCU le configure en mode « standby-I » ou en mode « power down ». Toutefois, si les fonctions de protocole automatique (Enhanced ShockBurst) du moteur de protocole en bande de base sont activées, le nRF24L01+ peut entrer dans d'autres modes afin d'exécuter le protocole.
- En mode RX, un signal RPD « Received Power Detector » est disponible. Le RPD est un signal qui est activé lorsqu'un signal RF supérieur à -64 dBm est détecté dans le canal de fréquence de réception. Le signal RPD interne est filtré avant d'être présenté au registre RPD. Le signal RF doit être présent pendant au moins 40µs avant que le RPD ne soit activé.

— **Mode TX :** est un mode actif de transmission de paquets. Pour entrer dans ce mode, le nRF24L01+ doit avoir le bit PWR_UP positionné au niveau haut, le bit PRIM_RX positionné au niveau bas, une charge utile dans la FIFO TX et une impulsion haute sur le CE pendant plus de 10µs.

— Le nRF24L01+ reste en mode TX jusqu'à la fin de la transmission d'un paquet. Si CE est bas, il revient en mode veille. Si CE est haut, l'état de la FIFO TX détermine l'action : s'il n'est pas vide, il reste en mode TX et transmet le paquet suivant ; s'il est vide, il passe en mode « standby-II ». Le PLL de l'émetteur fonctionne en boucle ouverte en mode TX. Il ne

doit jamais rester en mode TX plus de 4 ms. Avec les fonctions « Enhanced ShockBurst » activées, il ne dépasse jamais 4 ms en mode TX.

Le passage entre les modes se fait automatiquement en fonction des signaux CE et de l'état des FIFOs. Le nRF24L01 gère de manière autonome les acquittements, retransmissions et vidage des FIFOs grâce au protocole Enhanced ShockBurst.

En résumé, le nRF24L01 est un module radio très flexible et efficace, permettant une communication sans fil bidirectionnelle à faible coût et faible consommation, grâce à son protocole propriétaire et ses différents modes de fonctionnement.

2.6.3 Protocole « Enhaced SchockBurst »

Ce protocole, créé par Nordic Semiconductor, utilise la trame de paquets de la figure 2.9. Il offre une consommation d'énergie, une latence et des coûts réduits, ainsi qu'un taux de transmission élevé. Adapté à la communication bidirectionnelle entre appareils IoT et réseaux sans fil, il supporte l'authentification, le chiffrement et les mots de passe de messagerie [20].

Préambule (1 Octet)	Adresse (3-5 Octet)	Champ de contrôle des paquets (PFC) (9 bits)	Payload (0-32 Octet)	CRC (1-2 Octet)

Figure 2. 9 : Trame de paquets du protocole Enhanced ShockBurst [20]

☐ **Fonctionnement d'un protocole de communication sans fil :**

Le protocole Enhanced ShockBurst (ESB) organise la transmission des données en trames de messages. En mettant en lumière les différentes parties de cette trame :

- **Préambule :** C'est une séquence de bits (01010101 ou 10101010) au début de la trame de message qui aide le récepteur à synchroniser et à reconnaître le début du message.
- **Adresse :** Cette section contient l'adresse du module en mode récepteur. L'adresse du récepteur dans le module transcepteur peut être configurée avec 3, 4 ou 5 octets pour identifier le destinataire du message.
- **Champ de contrôle des paquets (PCF) :** Cette partie trame contient trois composants:
  - La longueur de la charge utile du message envoyé. (Figure 2.10)
  - PID (Paquet ID) pour déterminer si le paquet reçu est nouveau ou en double, aidant à éviter l'envoi de données dupliquées.

- ACK (Acquittement) pour confirmer la réception du message, assurant ainsi une communication fiable.

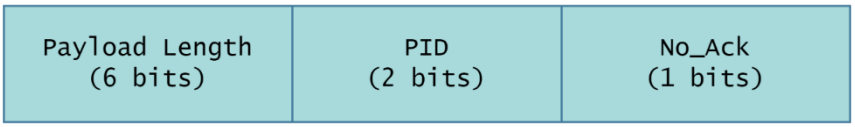

Figure 2. 10 : Détails de PCF [20]

2.7 Interfaçage du module nRF24L01

Cette section traite de l'interfaçage du module nRF24L01 avec les microcontrôleurs ESP32 et STM32, couvrant l'installation initiale, les connexions matérielles et les étapes de configuration. Pour l'ESP32, elle explique l'utilisation de l'ESP-IDF, les bibliothèques pertinentes et les affectations de broches pour le protocole SPI. Pour le STM32, elle détaille la configuration avec STM32CubeMX et l'initialisation de l'interface SPI. Des exemples pratiques et du code illustrent la transmission et la réception de données, assurant une compréhension claire du processus.

2.7.1 Interfaçage avec ESP32-S3

Dans cette partie, nous abordons spécifiquement l'interfaçage du module nRF24L01 avec le microcontrôleur ESP32. Le tableau 2.3 montre le brochage entre l'ESP32 et le module nRF24L01. Ainsi la figure 2.11 montre le montage électrique pour le module nRF24L01 avec le microcontrôleur ESP32-S3.

Tableau 2. 4: Brochage entre ESP32 et le module nRF24L01.

nRF24L01	ESP32-S3-WROOM-1
VCC	3.3V
GND	GND
CE (Chip Enable) : Active à l'état haut	GPIO 46
CSN (Chip Select Not): Active à l'état bas	GPIO 9
SCK (Serial Clock)	GPIO 12
MOSI (Master Out Slave In) : Entrée de données SPI	GPIO 13
MISO (Master In Slave Out) : Sortie de données SPI	GPIO 11

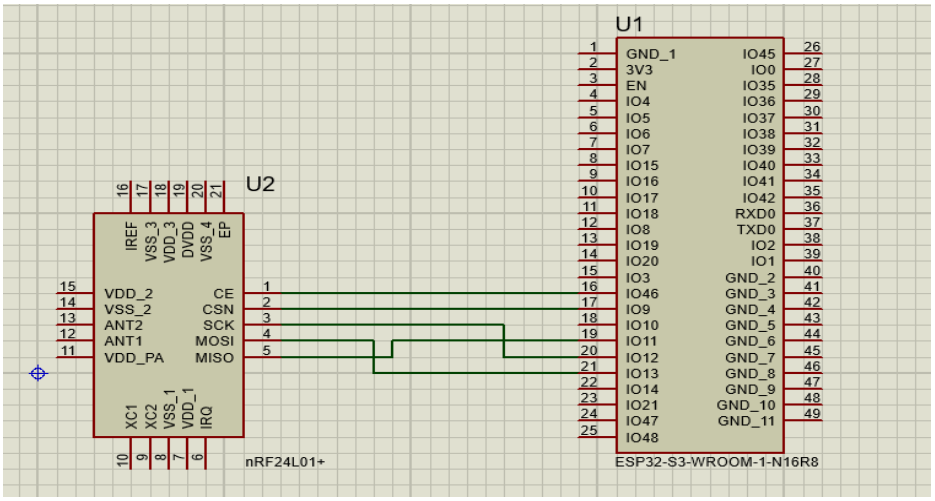

Figure 2. 11 : Schéma électrique de l'ESP32-S3 avec le module nRF24L01+

Après avoir correctement connecté le module nRF24L01 au microcontrôleur ESP32, il est essentiel de fait une configuration software pour permettre la communication via le protocole SPI. Cette configuration implique l'utilisation de l'environnement de développement ESP-IDF et l'installation des bibliothèques nécessaires.

Nous avons utilisé trois bibliothèques essentielles pour faciliter cette interface :

— **« driver/spi_master.h » :** Cette bibliothèque fournit des fonctionnalités d'interface avec les périphériques SPI (Serial Peripheral Interface).

— **« driver/gpio.h » :** Cette bibliothèque facilite la gestion et le contrôle des broches GPIO (General Purpose Input/Output) du microcontrôleur.

— **« NRF.h » :** Cette bibliothèque nous permet de contrôler efficacement nos modules NRF, en fournissant les fonctionnalités nécessaires pour une communication et une intégration transparente au sein de notre système.

Pour la configuration des pins GPIO l'extrait de code suivant illustre le processus d'initialisation des broches Chip Enable (CE) et Chip Select (CSN) utilisées par le module radio NRF24L01, en utilisant la bibliothèque « driver/gpio.h ».

```
12
13  #include <driver/gpio.h>
14  #define PIN_NUM_CSN 9
15  #define PIN_NUM_CE 46
16  void gpio_Config(void)
17  {
18  gpio_reset_pin(PIN_NUM_CE);
19  gpio_set_direction(PIN_NUM_CE, GPIO_MODE_OUTPUT);
20  gpio_set_level(PIN_NUM_CE, 0);
21  gpio_reset_pin(PIN_NUM_CSN);
22  gpio_set_direction(PIN_NUM_CSN, GPIO_MODE_OUTPUT);
23  2
24  gpio_set_level(PIN_NUM_CSN, 1);
25  }
26
```

Figure 2. 12: Initialisation des broches « CE » et « CSN »

Après l'initialisation de CE et CSN, on passe à la configuration du bus SPI pour la communication avec le module NRF24L01, la configuration est illustrée dans le code présenté dans la figure 2.13, en utilisant les paramètres suivants.

Tableau 2. 5 : Paramètre à configurer pour la communication SPI

Paramètre	Défini par
SCLK (SCLK (Serial Clock)	PIN NUM SCLK
MOSI (Master Out Slave In)	PIN NUM MOSI
MISO (Master In Slave Out)	PIN NUM MISO
Taille maximale de transfert	100 octets (réglable)
Mode SPI	Mode 0 (CPOL=0, CPHA=0 - standard pour NRF24L01)
Coefficient d'utilisation	50% (128 incréments)
Broche CS (Chip Select)	Non utilisée par ce dispositif (-1)
Taille de la file d'attente des transactions	Définie à 1

```
#include <driver/spi_master.h>
#define PIN_NUM_SCLK 12
#define PIN_NUM_MOSI 13
#define PIN_NUM_MISO 11
#define TAG "NRF24"
#define HOST_ID SPI2_HOST
static const int SPI_Frequency = 4000000;
spi_device_handle_t SPIHandle;
esp_err_t SPI_Config(void)
{
esp_err_t ret;
printf("Initialize SPI bus\n");
spi_bus_config_t buscfg = {
.sclk_io_num = PIN_NUM_SCLK,
.mosi_io_num = PIN_NUM_MOSI,
.miso_io_num = PIN_NUM_MISO,
.quadwp_io_num = -1,
.quadhd_io_num = -1,
.max_transfer_sz = 100, breackline
};
ret = spi_bus_initialize(HOST_ID, &buscfg,
SPI_DMA_CH_AUTO);
ESP_LOGI(TAG, "spi_bus_initialize=%d", ret);
assert(ret == ESP_OK); // Assert if initialization fails
spi_device_interface_config_t devcfg = {};
memset(&devcfg, 0,sizeof(spi_device_interface_config_t));
spi_device_interface_config_t
device_interface_config = {
.mode = 0,
.clock_source = SPI_CLK_SRC_DEFAULT,
.duty_cycle_pos = 128,
.clock_speed_hz = SPI_Frequency,
.spics_io_num = -1,
.queue_size = 1,
};
ret = spi_bus_add_device(SPI2_HOST,
&device_interface_config, &SPIHandle);
ESP_LOGI(TAG, "spi_bus_add_device=%d", ret);
assert(ret == ESP_OK); // Assert if device additionfails
return ret;
}
```

Figure 2. 13 : Configuration du SPI sur ESP-IDF

2.7.2 Interfaçage avec STM32F446RE

Dans cette partie, nous abordons spécifiquement l'interfaçage du module nRF24L01 avec le microcontrôleur STM32. Le tableau 2.6 montre le brochage entre STM32 et le module nRF24L01. La figure 2.14 montre le schéma électrique pour le module nRF24L01 avec le microcontrôleur STM32F446RE.

Tableau 2. 6: Brochage entre STM32 et le module nRF24L01.

nRF24L01	STM32F446RE
VCC	3.3V
GND	GND
CE (Chip Enable) : Active à l'état haut	GPIO PB4
CSN (Chip Select Not): Active à l'état bas	GPIO PB5
SCK (Serial Clock)	GPIO PB13
MOSI (Master Out Slave In) : Entrée de données SPI	GPIO PB15
MISO (Master In Slave Out) : Sortie de données SPI	GPIO PB14

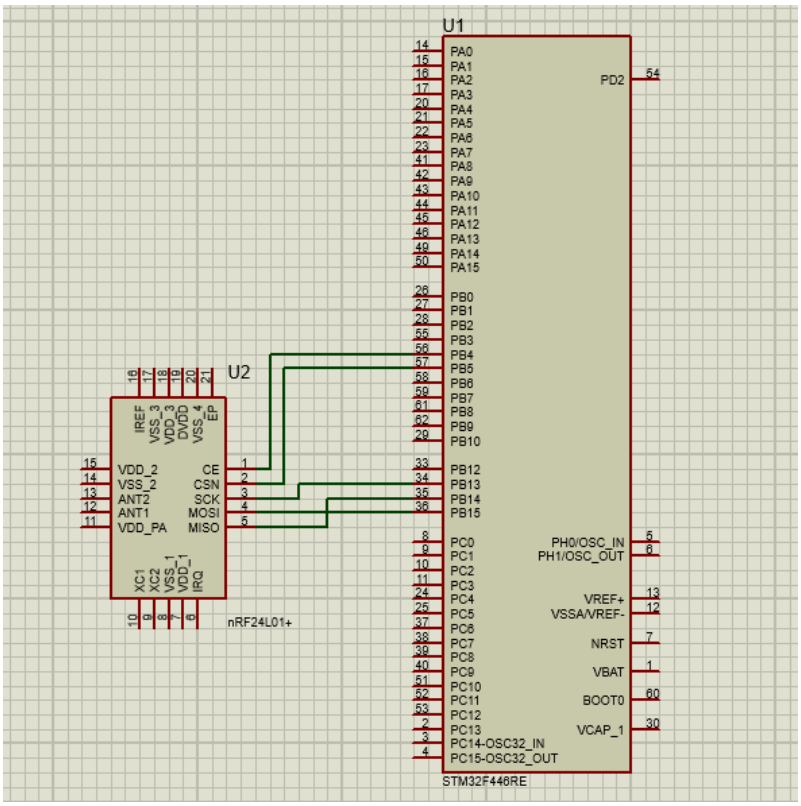

Figure 2. 14: Schéma électrique de branchement du STM32F446RE avec le module nRF24L01

Après avoir connecté le module nRF24L01 au STM32F446RE, il est essentiel de configurer le logiciel pour la communication SPI via STM32CubeIDE. Cet outil graphique simplifie la configuration des paramètres SPI et GPIO, rendant le processus plus convivial que l'ESP-IDF. Il est également nécessaire d'installer les bibliothèques nécessaires pour assurer une communication efficace, dont nous avons utilisé trois pour faciliter l'interfaçage :

— **« spi.h » :** cette bibliothèque fournit des fonctionnalités d'interface avec les périphériques SPI (Serial Peripheral Interface).

— **« gpio.h » :** Cette bibliothèque facilite la gestion et le contrôle des broches GPIO (General Purpose Input/Output) sur le microcontrôleur.

— **« MY NRF24.h » :** Cette bibliothèque nous permet de contrôler efficacement nos modules NRF, en fournissant les fonctionnalités nécessaires pour une communication et une intégration transparente au sein de notre système.

— « **HAL » :** La bibliothèque « HAL » ou (Hardware Abstraction Layer) offre de nombreux avantages qui simplifient le processus de manipulation du matériel.

Une fois que nous avons installé les bibliothèques, nous avons ensuite configuré deux broches GPIO, PB4 et PB5, avec les mêmes paramètres que les sorties, afin de les connecter

aux broches CE et CSN du module NRF. C'est ce que montre la figure 2.15.

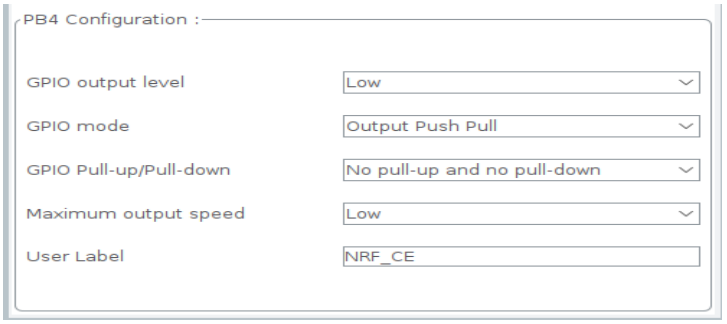

Figure 2. 15: Configuration de GPIO sur Cube MX

Par la suite, nous avons procédé à la configuration de l'interface afin de communiquer avec le NRF en utilisant SPI2 dans notre STM32. On peut le voir dans la figure 2.16.

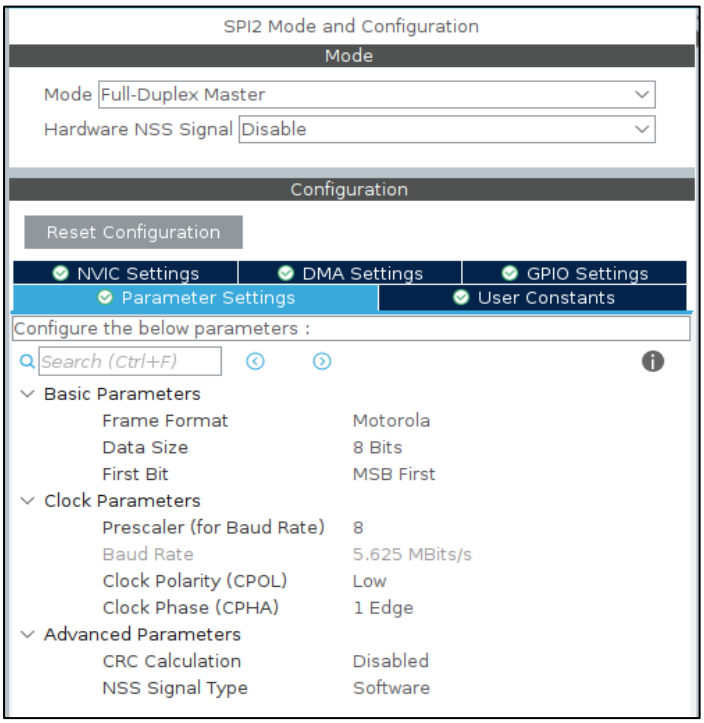

Figure 2. 16: Configuration de SPI sur Cube MX

2.8 Free real time operating system

FreeRTOS ou « Free Real-Time Operating System » est un kernel temps réel (ou planificateur) conçu pour construire des applications embarquées qui doivent répondre à des exigences strictes en matière de temps réel. Il permet aux applications d'être structurées comme une collection de threads d'exécution indépendants, chaque thread d'exécution est appelé une tâche. Sur les processeurs ayant un seul cœur, un seul thread peut s'exécuter à un moment donné. Le kernel détermine quel thread doit être exécuté en fonction de la priorité attribuée à chaque thread par le concepteur de l'application [21].

Il existe des techniques efficaces pour écrire des softwares embarqués sans kernel, qui conviennent aux systèmes simples. Pour les systèmes complexes, l'utilisation d'un kernel comme FreeRTOS est souvent préférable, bien que la décision soit subjective.

La hiérarchisation des tâches permet de respecter les délais de traitement, et l'utilisation d'un kernel offre des avantages supplémentaires [21] :

— **Abstraction temporelle :** Simplifie le code de l'application en gérant la synchronisation de l'exécution par le biais d'une API liée au temps, ce qui réduit la taille du code.
— **Maintenabilité et extensibilité** : La réduction des interdépendances entre les modules permet de contrôler l'évolution du software et d'améliorer les performances en cas de changement de hardware.
— **Modularité :** Des tâches indépendantes avec des objectifs définis améliorent l'organisation.
— **Développement d'équipes :** Des interfaces de tâches définies facilitent la collaboration.
— **Tests plus faciles :** Les softwares indépendants dotés d'interfaces propres peuvent être testés de manière isolée.
— **Réutilisation du code :** Une plus grande modularité et moins d'interdépendances simplifient la réutilisation du code.
— **Efficacité accrue :** Les softwares orientés événements permettent d'éviter les pertes de temps de traitement. Bien que les interruptions de ticks du RTOS ajoutent de la surcharge, les applications non-RTOS incluent généralement des interruptions de ticks également.
— **Temps d'inactivité :** la tâche d'inactivité mesure la capacité disponible, effectue des vérifications en arrière-plan ou met le processeur en mode basse consommation.
— **Gestion de l'énergie :** Les gains d'efficacité permettent de passer plus de temps en mode basse consommation, le mode "tick-less" de FreeRTOS réduisant encore plus la consommation d'énergie.
— **Gestion flexible des interruptions :** Des gestionnaires d'interruption courts reportent le traitement sur des tâches d'application ou sur la tâche " daemon" de FreeRTOS.
— **Exigences de traitement variées :** Supporte les traitements périodiques, continus et réguliers, répondant aux besoins de temps réel dur et mou avec des priorités de tâches et d'interruptions appropriées.

2.9 Configuration de la passerelle

Dans cette section, nous allons vous montrer comment nous configurons notre passerelle

étape par étape, en commençant par l'installation de la NRF jusqu'à l'implémentation de FreeRTOS.

2.9.1 Configuration du NRF

La configuration NRF consiste à initialiser et à configurer le module nRF24L01 pour établir la communication avec les nœuds de capteurs.

Voici les étapes de la configuration du module NRF :

— **Initialisation du module nRF24L01 :** Dans cette partie, nous incluons les bibliothèques nécessaires et initialisons les broches GPIO et l'interface SPI pour établir la communication entre la NRF et l'ESP32. Pour plus de détails, reportez-vous à l'interfaçage avec ESP32.

— **Configuration des paramètres de communication :** Le tableau 2.7 illustre le processus de configuration des paramètres de communication dans la NRF.

Tableau 2. 7: Configuration des paramètres de communication

Paramètre	Configuration
Débit de données	250 Kbit/s
Niveau de données	0 dB
Longueur de CRC	16 bits
Payload dynamique	Désactiver
Acquittement automatique	Activer
Canal	52

2.9.2 Configuration WIFI

Dans cette section, nous configurons l'ESP32 pour qu'il établisse une connexion avec un réseau Wi-Fi. Cela permet à la passerelle d'accéder à l'internet, facilitant ainsi la communication avec le cloud.

— Nous avons utilisé quatre bibliothèques essentielles pour accomplir cette tâche **:**

- **« esp_wifi » :** Gère les connexions Wi-Fi sur les appareils ESP32, offrant des fonctions pour configurer les modes Wi-Fi, les points d'accès et les connexions réseau.

- **« esp_log.h » :** faciliter le 'logging' sur l'ESP32, permettant aux développeurs de 'log' des messages à différents niveaux (debug, info, warning, error) pour le 'debugging' et le 'monitoring'.
- **« nvs_flash » :** fournit des fonctions d'accès au stockage non volatil (NVS ou Non-Volatile Storage) sur l'ESP32, essentiel pour le stockage de données persistantes.
- **« esp_event » :** Gère les événements et les gestionnaires d'événements sur l'ESP32, ce qui est essentiel pour gérer les événements liés au Wi-Fi tels que les changements d'état de connexion, l'acquisition d'une adresse IP et les déconnexions.

— **Configuration de paramètre Wi-Fi :** Dans un premier temps, nous configurons notre ESP32 en tant que station Wi-Fi en définissant le SSID (Service Set Identifier) et le mot de passe du réseau Wi-Fi à l'aide de la fonction "wifi init". Ces informations sont essentielles pour établir une connexion.

— **Connection au réseau Wi-Fi :** L'ESP32 commence le processus de connexion en lançant une tentative de connexion à l'aide du SSID et du mot de passe fournis. Il attend qu'une connexion soit établie avant de poursuivre.

— **Gestion des déconnexions et reconnexions du réseau :** Pour garantir la robustesse du système, l'implémentation incorpore des mécanismes permettant de gérer de manière élégante les déconnexions du réseau à l'aide de la fonction "esp_err_t wifi_event_handler". L'ESP32 surveille en permanence son état de connexion et tente de se reconnecter dès qu'il détecte une déconnexion.

— **Configuration de l'adresse IP :** L'ESP32 a la capacité d'obtenir dynamiquement une adresse IP du routeur par le DHCP (Dynamic Host Configuration Protocol), ce qui permet une configuration de réseau sans interruption.

— Le code suivant illustre la mise en œuvre de toutes les configurations susmentionnées dans l'ESP-IDF

```c
#include "esp_wifi.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_event.h"
#define ESP_WIFI_SSID ""
#define ESP_WIFI_PASS ""
#define MAX_RETRY 10

static int retry_cnt = 0;

static esp_err_t wifi_event_handler(void *arg, esp_event_base_t event_base, int32_t event_id, void *event_data),
{
    switch (event_id)
    {
    case WIFI_EVENT_STA_START:
        esp_wifi_connect();
        ESP_LOGI(TAG, "Trying to connect with Wi-Fi\n");
        break;

    default:
        break;
    }
    return ESP_OK;
}
void wifi_init(void)
{

    wifi_config_t wifi_config = {
        .sta = {
        .ssid = ESP_WIFI_SSID, // Set WiFi SSID
        .password = SP_WIFI_PASS, // Set WiFi password
        .threshold.authmode = WIFI_AUTH_WPA2_PSK // Set authentication mode
        }
    };

}
```

Figure 2. 17: Configuration de Wi-Fi sur ESP-IDF

2.9.3 Configuration du MQTT

Dans cette partie, nous exposons la configuration du protocole MQTT sur notre appareil ESP32 partie par partie.

— Nous utilisons toutes les bibliothèques de la section Wi-Fi et incluons « mqtt_client.h » pour réaliser cette tâche. Cette bibliothèque facilite la mise en œuvre des fonctionnalités du protocole MQTT, permettant une messagerie efficace entre les appareils IoT et les serveurs. Elle simplifie le processus d'intégration des fonctionnalités MQTT dans les projets ESP32, assurant un échange de données continu au sein des réseaux IoT.

— **Configuration et initialisation :** Pour configurer MQTT dans ESP32, il est nécessaire de spécifier les détails du broker MQTT auquel vous vous connectez, y compris l'adresse du serveur et le port, qui sont encapsulés dans « broker.address.uri ». Cette configuration est généralement réalisée à l'aide de la fonction « mqtt_app_start ».

— **Publication et abonnement :** Après avoir initialisé la connexion au broker, nous devons publier des données vers un topic spécifique et nous abonner à certains topics spécifiques. Pour ce faire, nous utilisons la fonction "esp_mqtt_client_publish" pour publier des données et la fonction "esp_mqtt_client_subscribe" pour s'abonner à des rubriques.

```
// Publish data to a GATWAY topic
esp_mqtt_client_publish(client, MQTT_PUB_GATWAY, &(ESP32_Characteristic), 0, 0, 0);

// Subscribe to a GATWAY topic
esp_mqtt_client_subscribe(client, MQTT_SUB_GATWAY, 0);
```

Figure 2. 18: Publication et Abonnement via MQTT

Le code suivant illustre la mise en œuvre de toutes les configurations susmentionnées dans l'ESP-IDF.

```
#include "mqtt_client.h"
#include "esp_event.h"
#include "esp_wifi.h"
#include "esp_log.h"
static const char *TAG = "MQTT";
uint32_t MQTT_CONNECTED = 0;
#define BROKER_URI "mqtt: //192.168.244.178:1883"
#define MQTT_PUB_GATWAY "esp32/gatway"
#define MQTT_SUB_GATWAY "esp32/gatwaycommand"
static void mqtt_event_handler(void *handler_args,esp_event_base_t base, int32_t event_id, void *event_data)
{
    ESP_LOGD(TAG, "Event dispatched from event loop base=%s,
    event_id=%ld", base, event_id);,
    esp_mqtt_event_handle_t event = event_data;
    switch ((esp_mqtt_event_id_t)event_id)
    {
    case MQTT_EVENT_CONNECTED:
    ESP_LOGI(TAG, "MQTT_EVENT_CONNECTED");
    ...
    }
}
static void mqtt_app_start(void)
{
    ESP_LOGI(TAG, "STARTING MQTT");
    // Initialize MQTT client configuration with the broker URI
    esp_mqtt_client_config_t mqttConfig = {
    .broker.address.uri = BROKER_URI // Set the URI of the
    MQTT broker (e.g., IP address or domain name),
    };
    ...
    esp_mqtt_client_start(client);
}
```

Figure 2. 19 : Configuration MQTT sur ESP-IDF

2.9.4 Implémentation freeRTOS

Cette section explique notre processus d'implémentation de FreeRTOS sur l'ESP32, visant à gérer efficacement les tâches en temps réel. Il s'agit de la bibliothèque fondamentale

que nous avons utilisée pour implémenter FreeRTOS dans la passerelle :

— Ce sont les bibliothèques essentielles que nous avons utilisées pour implémenter FreeRTOS dans la passerelle.

- **« freertos/FreeRTOS » :** cette bibliothèque est la fonctionnalité de base du noyau FreeRTOS, qui fournit des primitives d'ordonnancement, de gestion des tâches et de synchronisation.
- **« freertos/timers » :** cette bibliothèque permet de créer et de gérer des software timers dans l'environnement FreeRTOS.
- **« freertos/semphr » :** Cette bibliothèque fournit des objets sémaphores, qui sont utilisés pour la signalisation entre les tâches et le contrôle de l'accès aux ressources partagées.
- **« freertos/queue » :** Cette bibliothèque fournit des structures de données et des fonctions de file d'attente pour la communication entre les tâches.

— **Configuration de la gestion des tâches :** Pour notre mise en œuvre, nous avons créé trois tâches principales. La première tâche, qui a la priorité la plus élevée, gère la communication entre la passerelle et les nœuds de capteurs. La deuxième tâche gère le processus de publication vers un topic spécifique sur le broker MQTT. La troisième tâche est responsable de l'abonnement à un topic spécifique. Nous avons utilisé la fonction « xTaskCreate » pour créer ces tâches.

— **Communication entre tâches :** Pour la synchronisation et la communication entre les tâches, nous avons utilisé des sémaphores pour une signalisation simple et pour gérer l'accès aux ressources partagées. En outre, nous avons utilisé des files d'attente comme méthode fiable pour l'envoi et la réception de données entre les tâches.

```
void vTaskPublisherData(void *pvParameters)
{
    while (1)
    {
        ...
        if (xSemaphoreTake(MQTT_PUB_TREGER_NODES,, portMAX_DELAY) == pdTRUE)
        {
            for (size_t i = 0; i < NUMBER_OF_SLAVE; i++)
            {
                if (xQueueReceive(DATA_NODES, &(rxbuff),, (TickType_t)5))
                {
                    esp_mqtt_client_publish(client, topics[i], &(rxbuff), 0, 0, 0);
                    printf("NODE rxbuff : %s Time : %lld\n", rxbuff, esp_timer_get_time() / 1000);
                }
            }
        }
    ...
    }
}
void app_main(void)
{
    ...
    xTaskCreate(vTaskHandleNodeComm, "Task 1", 20960, NULL, 4, &myTask1Handle);
    xTaskCreate(vTaskPublisherData, "Task 2", 2048, NULL, 1,&myTask2Handle);
    xTaskCreate(vTaskSubscriberCommand, "Task 3", 2048, NULL, 1,&myTask2Handle);
    ...
}
```

Figure 2. 20: Implémentation de freeRTOS sur la passerelle

2.10 Configuration de la partie des nœuds de capteurs

Cette section détaille le processus de configuration des nœuds de capteurs, nous allons explorer le processus étape par étape, en commençant par la configuration de l'horloge du MCU jusqu'aux paramètres finaux du capteur.

Tout d'abord, nous devons configurer l'horloge de nos nœuds pour obtenir des performances maximales. Dans notre cas, la vitesse d'horloge maximale est de 180 MHz. Pour ce faire, nous utilisons CUBEMX, comme illustré dans la figure 2.21.

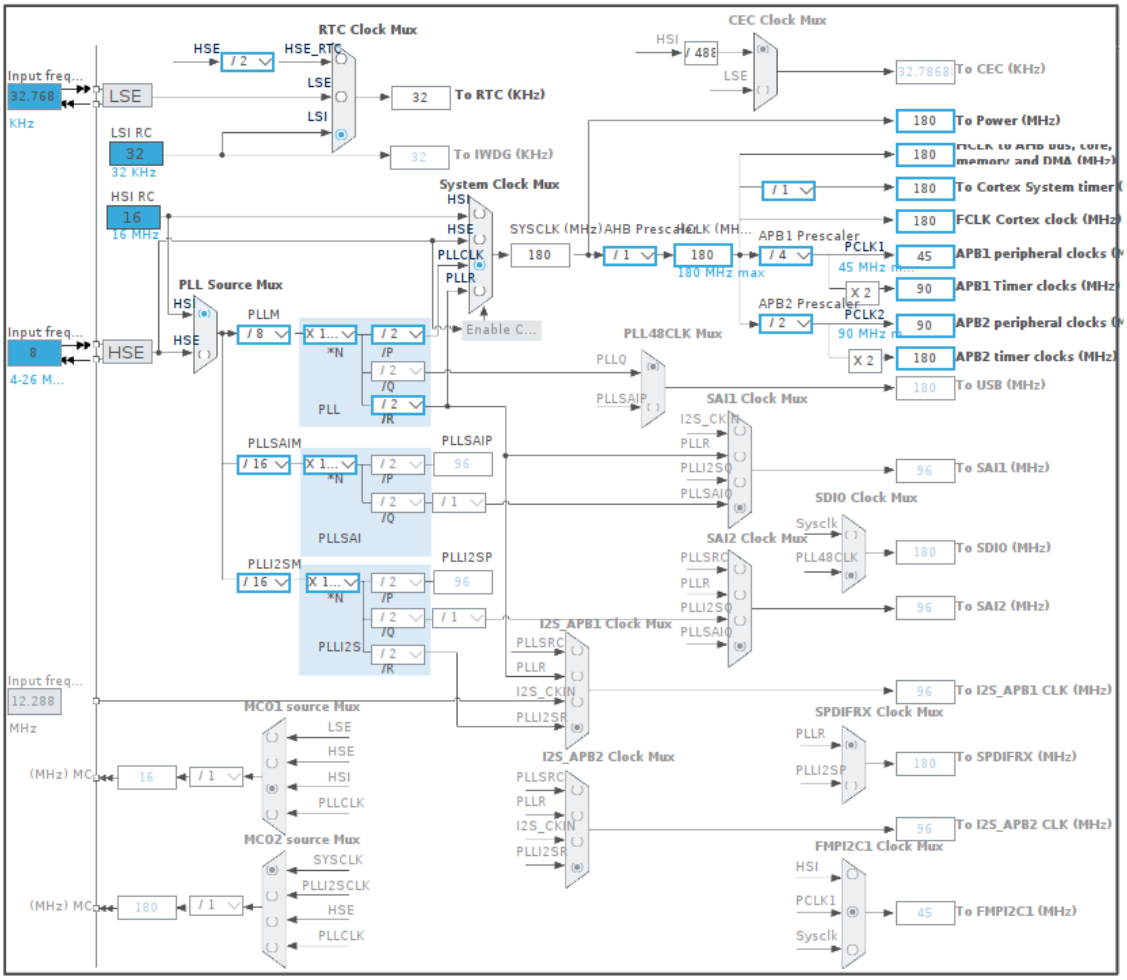

Figure 2. 21 : Configuration de l'horloge sur CubeMX

Ensuite, nous incluons les bibliothèques nécessaires et initialisons les broches GPIO et l'interface SPI pour établir la communication entre le NRF et le SMT32. Pour plus de détails, voir la section d'interfaçage du module nRF24L01 avec STM32.

Ensuite, on passe a la configuration des capteurs, pour le capteur DHT22, nous avons choisi de le connecter au port GPIO A, broche 1, en mode sortie pour établir la communication avec notre nœud, comme est illustré dans la figure 2.22.

Pour le capteur RGB, nous établissons un bus de communication série I2C pour permettre la communication entre notre MCU et le capteur RGB. Dans notre configuration, nous avons choisi I2C 1 à cette fin, simplement parce qu'il est disponible. La figure 2.23 illustre cette configuration.

Par ailleurs, le potentiomètre a été connecté à l'ADC 1 pour surveiller la variation de tension entre ses pôles. Dans notre configuration, nous avons choisi un Vcc de 3,3V. La figure 2.24 illustre la configuration de l'ADC.

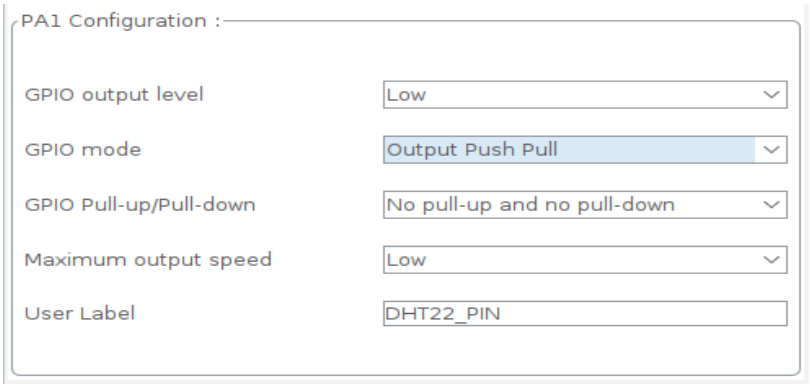

Figure 2. 22 : Configuration de pin GPIO pour le capteur DHT22

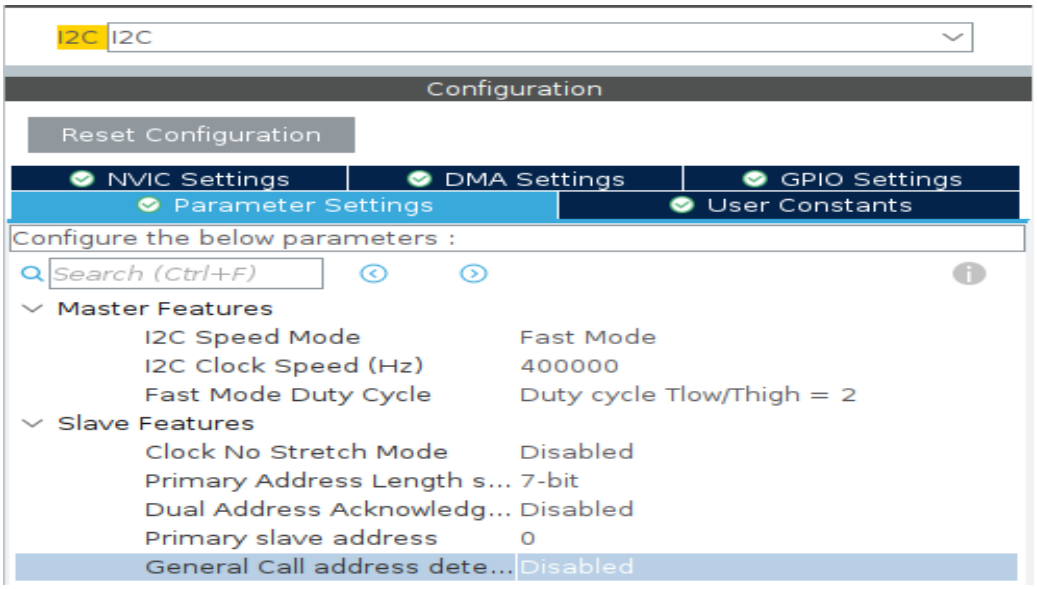

Figure 2. 23: Configuration du I2C pour le capteur couleur RGB

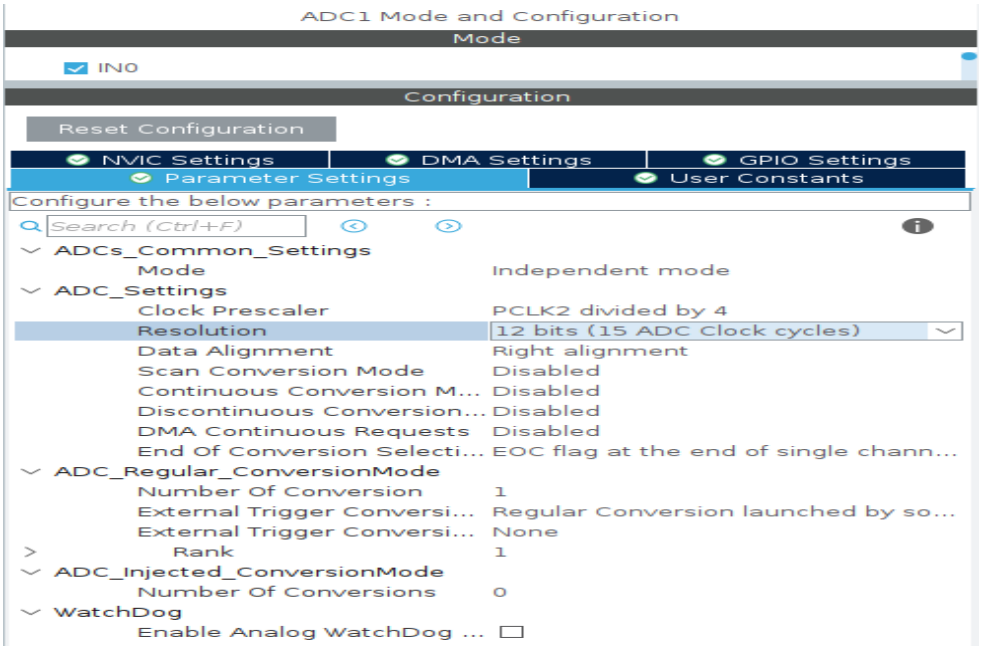

Figure 2. 24 : Configuration du ADC

Après avoir exécuté méticuleusement chacune des étapes de configuration décrites, nous sommes parvenus à une configuration complète. La figure 2.25 permet d'élucider les subtilités de la configuration globale des nœuds, en montrant l'aboutissement de nos efforts et les paramètres finement ajustés qui régissent désormais leur fonctionnement.

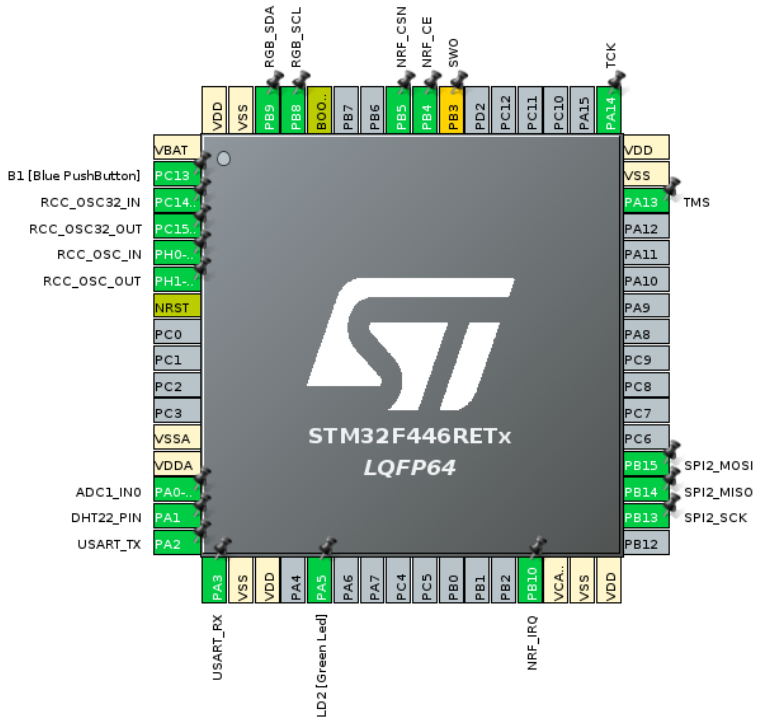

Figure 2. 25 : Configuration globale du nœud

2.11 Principe de fonctionnement du système

Après avoir détaillé, dans les sections précédentes, le fonctionnement du module nRF24L01 et son interfaçage avec les microcontrôleurs, cette section décrit le principe de fonctionnement de notre système.

Notre système fonctionne avec la passerelle en tant que maître et les nœuds de capteurs en tant qu'esclaves. Le principe opérationnel du système est divisé en deux parties qui sont la communication radio entre la passerelle et les nœuds de capteurs, et la communication Wi-Fi entre la passerelle et le cloud.

2.11.1 Communication radio entre la passerelle et les nœuds de capteurs

Au début, la passerelle effectue l'initialisation et la configuration du module NRF afin de le préparer à être utilisé. Ensuite, elle lance des demandes de données aux nœuds de capteurs. Pour ce faire, elle envoie un paquet de demande à un nœud de capteur désigné et attend un accusé de réception (ACK). Une fois la demande reçue, le nœud capteur répond rapidement en

envoyant un ACK à la passerelle. Si la passerelle ne reçoit pas d'ACK, elle répète la demande jusqu'à un maximum de cinq fois. En cas d'échec après trois tentatives, la passerelle réinitialise le module NRF, à la manière d'un mécanisme de Watchdog, afin de résoudre le problème.

Une fois acquittée, la passerelle passe en mode de réception (RX), prête à recevoir les transmissions de données du nœud de capteur. Simultanément, le nœud de capteurs passe de manière autonome en mode de transmission (TX) pour acheminer les données vers la passerelle. Le nœud de capteur transmet les données collectées à la passerelle et attend un ACK de la passerelle pour confirmer la réussite de la réception.

Après avoir reçu les données, la passerelle envoie rapidement un ACK au nœud de capteur, confirmant la bonne réception des données. Si le nœud de capteur ne reçoit pas d'ACK, il lance une retransmission des données, dans la limite de cinq itérations. Après réception de l'ACK de la passerelle, le nœud de capteur repasse en mode RX.

Ce processus répété se poursuit indéfiniment, avec un délai entre chaque demande pour garantir des mises à jour de données opportunes et cohérentes.

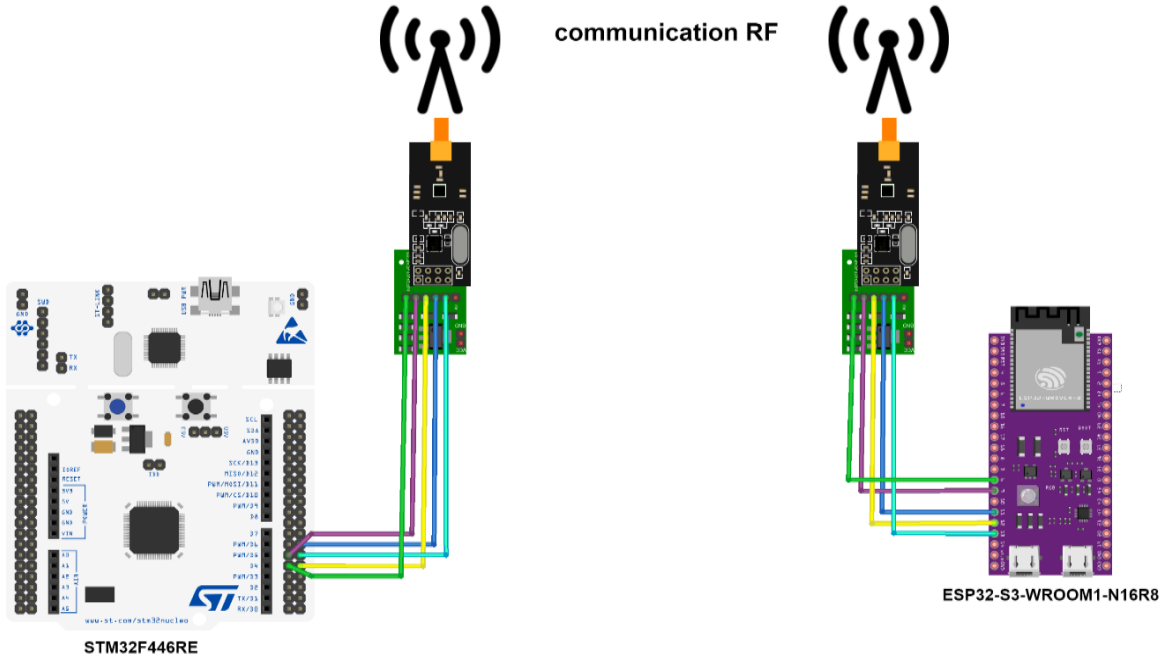

Figure 2. 26: Schéma de communication entre un nœud et la passerelle

L'organigramme dans la figure 2.27 illustre clairement le fonctionnement de la passerelle. Également, l'organigramme pour le nœud de capteurs dans la figure 2.28.

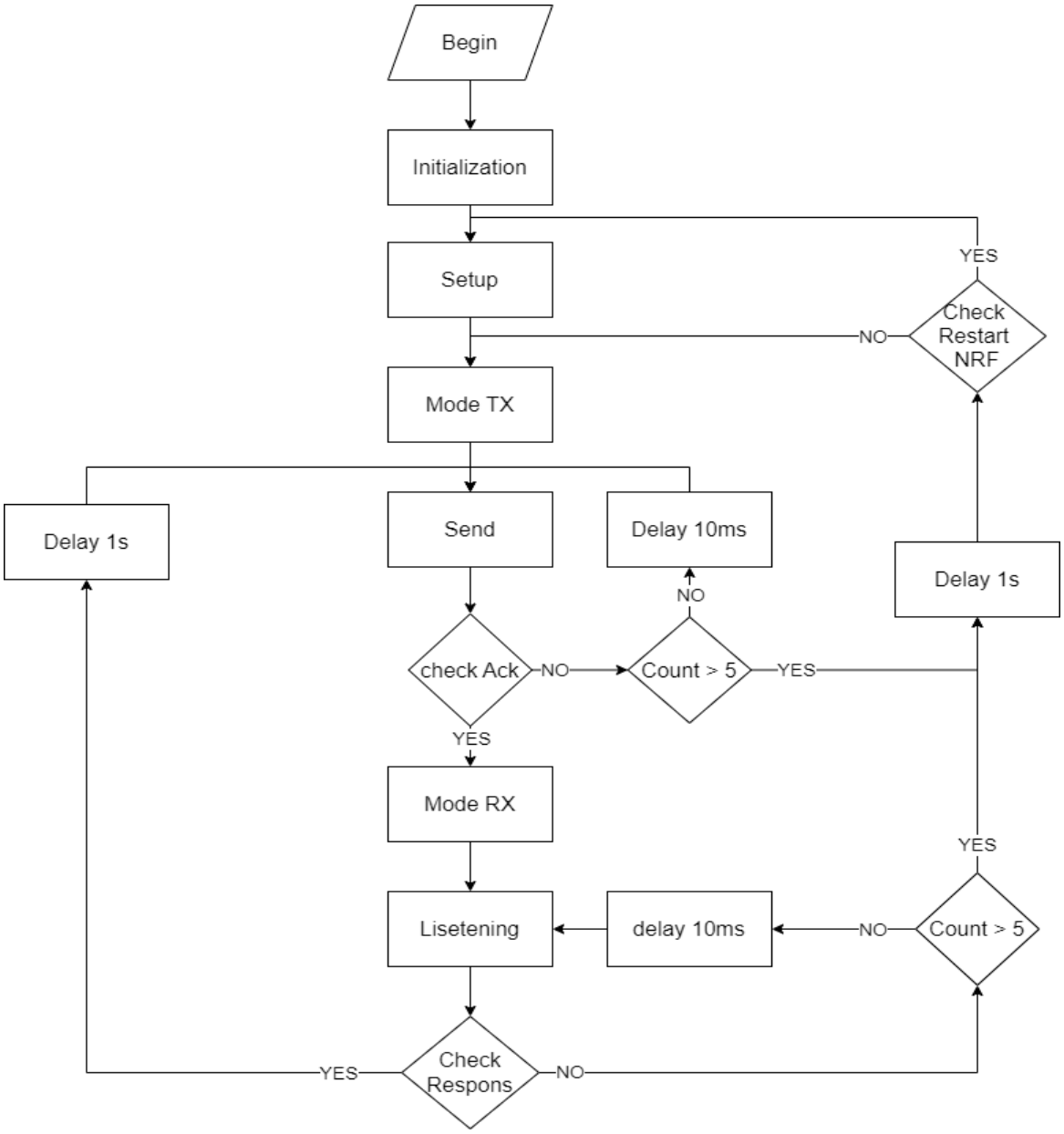

Figure 2. 27: Organigramme de fonctionnement de la passerelle

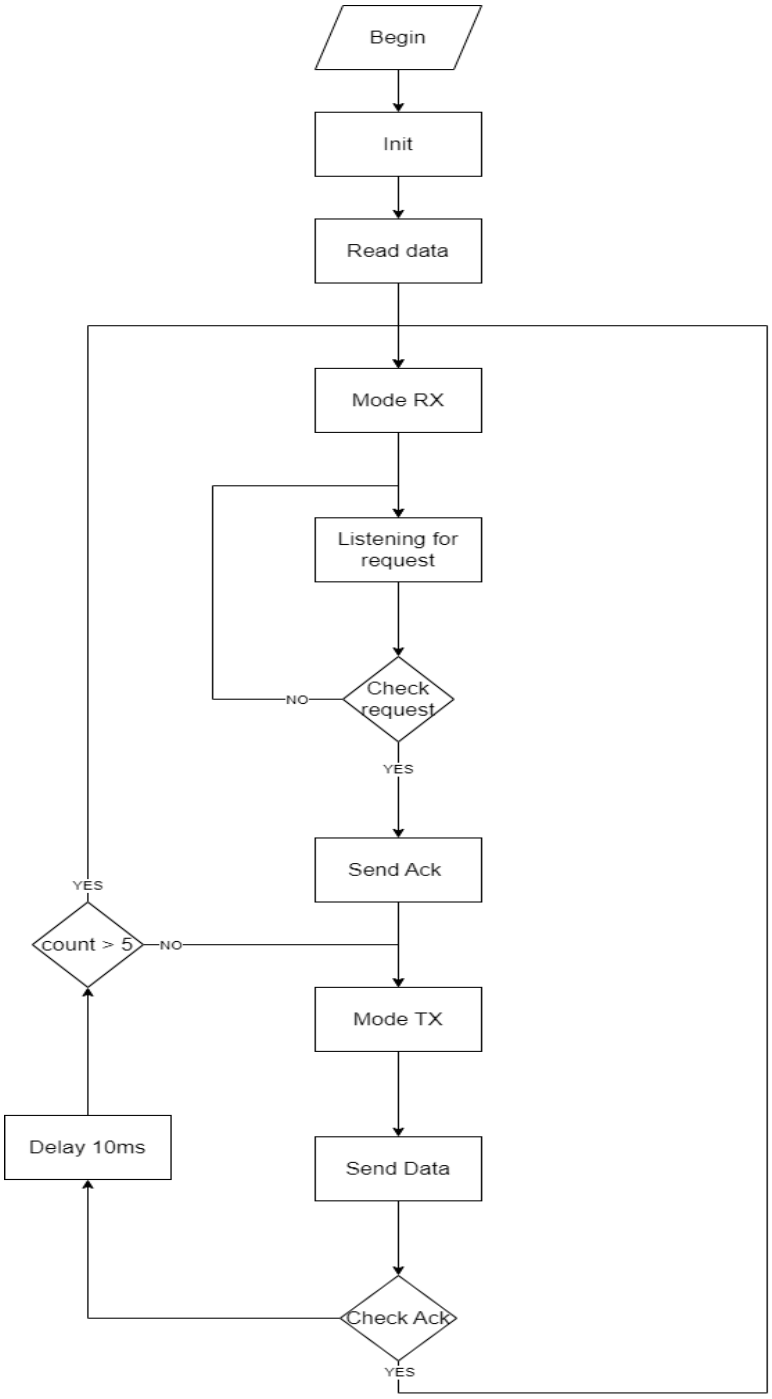

Figure 2. 28 : Organigramme de fonctionnement pour le nœud de capteurs

2.11.2 Communication Wi-Fi entre la passerelle et le cloud

La passerelle, équipée d'un microcontrôleur ESP32, utilise ses capacités Wi-Fi intégrées pour communiquer avec le Cloud. Ce processus comporte plusieurs étapes :

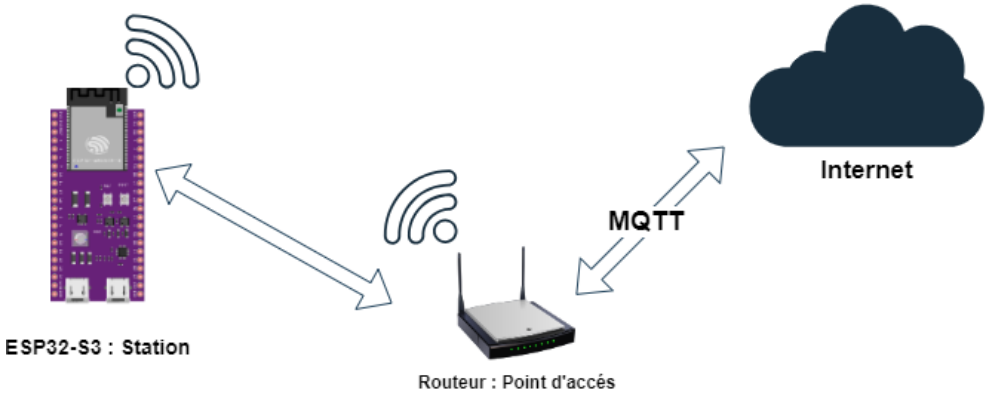

Figure 2. 29 : Schéma de connexion de l'ESP32 au cloud

— **Connection Wi-Fi :** La passerelle se connecte à un réseau Wi-Fi préconfiguré à l'aide de son SSID et de son mot de passe. Une fois connectée, elle obtient une adresse IP du routeur Wi-Fi.

— **Formatage des données :** La passerelle traite de manière experte les données des capteurs entrants, les transformant méticuleusement en un format JSON structuré, prêt pour une transmission transparente vers le nuage.

— **Protocole MQTT :** La passerelle initie de manière autonome la connexion au courtier MQTT, ce qui garantit une intégration et une communication transparentes au sein de l'architecture du réseau. La passerelle fonctionne comme un abonné et un publiant MQTT flexible, s'interfaçant dynamiquement avec divers topics tout en établissant une connexion robuste avec un broker MQTT situé dans l'environnement en cloud.

— **Transmission des données :** La passerelle publie les données formatées du capteur vers des topics MQTT spécifiques sur le broker MQTT. Le broker MQTT assure la livraison des messages aux clients abonnés (par exemple, les services en cloud) qui souhaitent recevoir des mises à jour de données.

2.12 Conclusion

Nous avons exposé dans ce chapitre la conception globale de notre système ainsi que les différentes étapes de notre travail sur ce projet. Afin d'atteindre ce stade de développement, nous avons réalisé de multiples tests et essais pour satisfaire aux exigences de notre cahier des charges, tout en cherchant toujours à optimiser le système et à ajouter de nouvelles fonctionnalités. Les tests et leurs résultats seront expliqués de manière plus détaillée dans le chapitre suivant.

Chapitre III

Résultat Expérimentaux et interprétation

3.1 Introduction

Dans le chapitre précédent nous avons détaillé la partie la conception de la passerelle pour l'internet industriel des objets, dans ce chapitre nous nous focalisons sur l'évaluation de cette passerelle avec des tests que nous essayons d'optimiser pour confirmer et garantir le bon fonctionnement de notre passerelle. Nous allons décrire l'environnement de travail et d'implémentation et les différents tests effectués, et discuter des résultats de chaque test.

3.2 Environnement de test

Pendant la réalisation des divers tests, nous avons utilisé différents éléments que nous avons déjà décrits dans le chapitre précédent. Toutes ces tâches ont été réalisées sur un ordinateur avec les caractéristiques suivantes :

— Processeur : Intel(R) Core(TM) i5-8500 U CPU @ 2.90GHz.
— Mémoire RAM installée : 16,00 Go.
— Système d'exploitation : Linux Ubuntu 20.04.

Une fois le matériel requis pour la réalisation de chaque partie du système obtenu, nous avons procédé à leur association pour effectuer les différents tests et réaliser notre passerelle. La figure 3.1 illustre le montage physique de la passerelle, composée d'une carte ESP32-S3 et d'un module nRF24L01 qui servira à la réception des données des nœuds de capteurs.

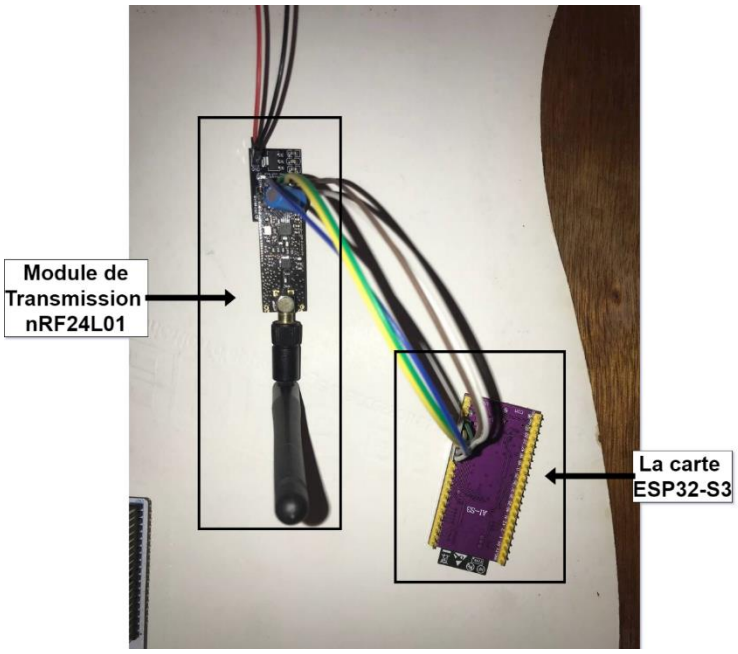

Figure 3. 1 : Montage physique pour la passerelle

De même, la figure 3.2 représente le montage physique d'un nœud, composé d'une carte STM32F446RE qui servira à la transmission des données des capteurs via le module nRF24L01.

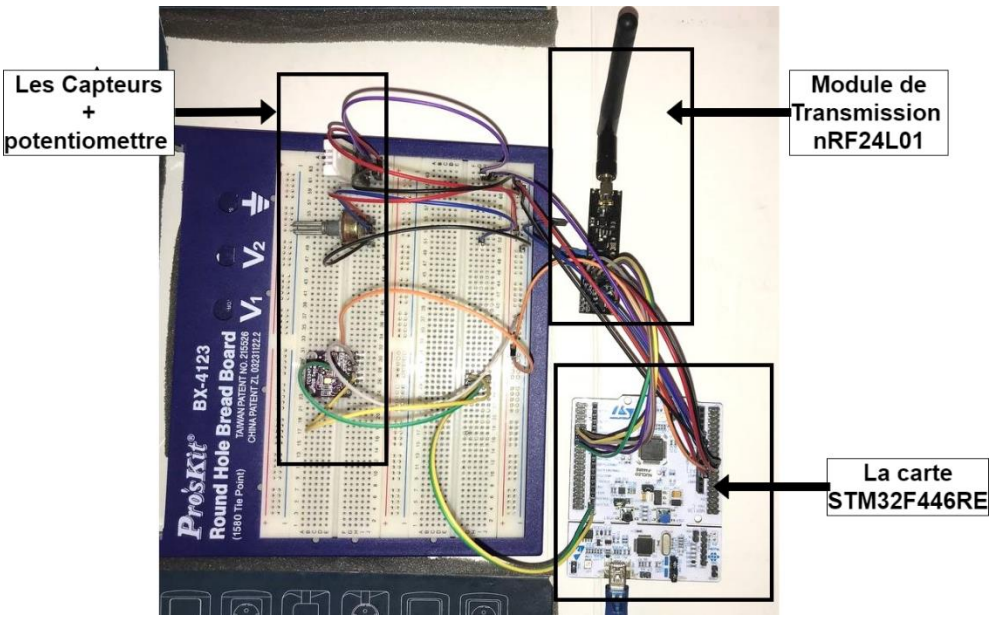

Figure 3. 2 : Montage physique d'un nœud

Nous allons utiliser les outils de développement ESP-IDF pour la carte ESP32 et STM32CUBEIDE et STM32CUBEMX pour la carte STM32, utilisant le langage C pour programmer les différentes tâches nécessaires, notamment l'émission et la réception des données via le module nRF24L01.

Après avoir réalisé notre passerelle, nous passons aux tests de performance pour évaluer son efficacité. Ces tests nous permettront de déterminer les limites de notre système et d'identifier les éventuelles optimisations nécessaires.

3.3 Test de performance de la passerelle

L'évaluation des performances de notre passerelle vise à déterminer son efficacité à communiquer avec un réseau de microcontrôleurs. Plus précisément, nous cherchons à déterminer l'intervalle de temps le plus court entre les requêtes de données consécutives envoyées par la passerelle aux nœuds de capteurs, tout en veillant à ce que les erreurs de communication ou les pertes de données ne dépassent pas 15 %. En outre, nous cherchons à comprendre le nombre maximum de nœuds de capteurs que notre passerelle peut gérer efficacement. Cette évaluation implique la réalisation de tests avant et après l'optimisation du code de la passerelle afin de mesurer l'impact de toute amélioration.

Nous visons plusieurs objectifs spécifiques afin de garantir que notre passerelle répond aux exigences de diverses applications :

— **Évaluation de l'efficacité :** Nous cherchons à évaluer l'efficacité de la communication de notre passerelle avec les nœuds des capteurs. Il s'agit d'analyser des facteurs tels que le temps de

réponse, le taux de succès et le taux d'erreur afin de déterminer les performances globales du processus de communication.

— **Optimisation de l'intervalle de temps :** Un aspect essentiel de notre évaluation consiste à déterminer l'intervalle de temps optimal entre les demandes de données successives envoyées par la passerelle aux nœuds capteurs. Cette optimisation garantit l'acquisition des données en temps voulu tout en minimisant les coûts de communication.

— **Stabilité de la communication :** Assurer la stabilité de la communication est important pour une transmission fiable des données au sein du système IoT. Nous visons à évaluer et à minimiser les erreurs de communication ou les pertes de données afin de maintenir l'intégrité et la précision des données collectées.

— **Capacité maximale des nœuds :** Comprendre le nombre maximum de capteurs que notre passerelle peut gérer efficacement est un point essentiel pour la planification de l'évolutivité. En identifiant cette capacité, nous pouvons concevoir des systèmes IoT capables de s'adapter à la croissance des déploiements de nœuds dans diverses applications.

— Nous prévoyons de comparer les performances avant et après l'optimisation du code afin de mesurer l'impact de toute amélioration. Cette comparaison nous aidera à évaluer l'efficacité des optimisations et à identifier les domaines à améliorer.

En nous concentrant sur ces objectifs, nous souhaitons garantir que notre passerelle IoT réponde aux exigences de performance de diverses applications, allant de l'automatisation industrielle, les villes intelligentes, la surveillance des soins de santé et la détection environnementale, de la santé et de la détection de l'environnement.

3.3.1 Méthodologie d'évaluation des performances

La méthodologie employée pour l'évaluation des performances de notre passerelle IoT est structurée de manière à garantir un test et une analyse complets de sa communication avec le nœud de capteurs. Les étapes suivantes décrivent notre approche :

- **Étape 1 :** consiste à établir des connexions physiques et logiques entre la passerelle IoT et les nœuds de capteurs. Il s'agit de configurer les composants matériels et d'établir des protocoles de communication pour faciliter l'échange de données en continu.
- **Étape 2 :** une fois la configuration terminée, la passerelle IoT est programmée pour générer et transmettre des requêtes de données aux nœuds capteurs à des intervalles prédéfinis. Cette configuration est essentielle pour assurer une acquisition et une transmission cohérentes des données.

- **Étape 3 :** nous ajustons itérativement l'intervalle de temps entre les demandes de données consécutives envoyées par la passerelle aux nœuds de capteurs. Ce processus itératif nous permet d'identifier l'intervalle optimal qui équilibre l'efficacité de l'acquisition des données avec une surcharge de communication minimale.
- **Étape 4** : tout au long du processus de test, nous surveillons et analysons de près les réponses reçues des nœuds de capteurs en réponse aux requêtes de données de la passerelle. Cette phase de contrôle se concentre sur l'évaluation d'indicateurs de performance clés tels que le temps de réponse, le taux de réussite et le taux d'erreur afin d'évaluer la fiabilité et l'efficacité de la communication, d'erreur afin d'évaluer la fiabilité et l'efficacité de la communication.
- **Étape 5** : nous calculons le nombre minimum attendu de demandes provenant de la passerelle dans un délai donné en fonction de l'intervalle sélectionné. Ce calcul sert de référence pour évaluer les performances de la passerelle et le respect des normes prédéfinies.
- **Étape 6** : pour valider la fonctionnalité de la passerelle, nous comparons le nombre prévu et le nombre réel de demandes reçues pendant les tests. Cette analyse comparative permet de vérifier que la passerelle fonctionne comme prévu et répond aux critères de performance prédéfinis.
- **Étape 7** : un aspect essentiel de notre évaluation consiste à déterminer le nombre maximal de nœuds de capteurs que la passerelle peut gérer efficacement dans l'intervalle de temps choisi. Cette analyse est essentielle pour planifier l'évolutivité et garantir la stabilité du système à mesure que les déploiements de nœuds de capteurs s'étendent.
- **Étape 8** : après la phase de test initiale, nous mettons en œuvre des optimisations ciblées dans le code de la passerelle afin d'améliorer les performances et l'efficacité. Ces optimisations peuvent concerner le perfectionnement des algorithmes, l'optimisation des ressources ou l'amélioration des protocoles.
- **Étape 9** : après avoir mis en œuvre les optimisations, nous effectuons des tests supplémentaires pour évaluer leur impact sur les performances de la passerelle. Ce processus itératif nous permet d'identifier d'autres possibilités d'optimisation et d'évaluer l'efficacité des changements mis en œuvre.

Grâce à cette méthodologie structurée, nous souhaitons procéder à une évaluation approfondie des performances de notre passerelle, afin de nous assurer qu'elle est adaptée à diverses applications et industries.

L'évaluation des performances de notre passerelle IIoT implique de mesurer l'efficacité et l'efficience de la communication avec le réseau de capteurs à travers plusieurs paramètres clés :

— **Temps de réponse :** ce paramètre mesure le temps nécessaire au nœud de capteurs pour répondre aux demandes de données envoyées par la passerelle. Un temps de réponse plus court indique une communication plus rapide et une meilleure performance du système. Il est essentiel pour les applications nécessitant un traitement des données en temps réel

— **Taux de succès :** représente le pourcentage d'échanges de données réussis entre la passerelle et les nœuds de capteurs. Il indique la fiabilité de la communication et de la transmission des données au sein du système. Un taux de succès élevé garantit la précision de la collecte des données et minimise le risque de perte de données.

— **Taux d'erreur :** cette mesure quantifie l'occurrence des erreurs de communication ou des pertes de données au cours du processus d'échange de données. Il est essentiel de maintenir le taux d'erreur à un niveau bas pour préserver l'intégrité des données et la fiabilité du système. En minimisant les erreurs de communication, nous garantissons l'exactitude et la cohérence des données collectées.

— **Plage acceptée :** le paramètre de portée acceptée détermine le nombre maximal de nœuds de capteurs que la passerelle peut gérer efficacement tout en maintenant des performances optimales. Il est essentiel de comprendre cette limite pour planifier l'évolutivité et garantir la stabilité du système IoT au fur et à mesure qu'il évolue. Elle permet de concevoir des systèmes robustes et résilients capables de s'adapter à la croissance des déploiements de capteurs dans diverses applications et industries.

— **Seuil :** Cette mesure évalue la performance du système par rapport à des seuils ou des objectifs de performance prédéfinis. En fixant des seuils de performance, nous pouvons établir des références pour la performance du système et identifier les domaines à améliorer. Le suivi des performances par rapport à ces seuils nous permet de nous assurer que le système répond aux normes et aux critères de performance requis, améliorant ainsi sa fiabilité et son efficacité globales.

En évaluant ces paramètres de manière exhaustive, nous obtenons des informations précieuses sur les performances de notre passerelle IoT et sur son adéquation à diverses applications dans différents domaines du paysage IoT, en améliorant les performances d'applications telles que le suivi des actifs, la surveillance à distance et la gestion de l'énergie, ainsi que l'automatisation industrielle, la surveillance des soins de santé et les systèmes de transport intelligents.

Les conclusions et les optimisations résultant de cette évaluation des performances ont des implications significatives pour diverses applications dans les systèmes IoT :

— **Amélioration des performances du système :** En optimisant l'efficacité et la fiabilité de la communication entre la passerelle et le réseau de capteurs, nous pouvons améliorer considérablement les performances globales de notre système IoT. Entre la passerelle et le réseau de capteurs, nous pouvons améliorer de manière significative les performances globales de notre système IoT. Cette amélioration peut se traduire par des temps de réponse plus rapides, une latence réduite et une précision accrue des données, au profit d'applications telles que la domotique intelligente, la surveillance industrielle et la détection environnementale.

— **Collecte efficace des données :** Une passerelle optimisée garantit une collecte efficace et opportune des données du réseau de capteurs. Ceci est essentiel pour les applications nécessitant des données en temps réel, telles que la maintenance prédictive, le suivi des actifs et la surveillance à distance. En rationalisant les processus d'acquisition de données, nous permettons une prise de décision mieux informée et une action plus réactive.

— **Planification de l'évolutivité :** Pour planifier l'évolutivité, il est essentiel de comprendre la capacité maximale de notre passerelle à gérer plusieurs nœuds de capteurs est essentielle pour la planification de l'évolutivité. Cette évolutivité est vitale pour les applications couvrant les villes intelligentes, la surveillance agricole et les systèmes de santé.

— **Validation de l'optimisation :** La validation de l'impact des optimisations de code sur les performances globales du système garantit la fiabilité et l'efficacité de notre infrastructure IoT. Ce processus de validation est essentiel pour maintenir l'intégrité du système et garantir des performances constantes dans le temps. Il fournit également des informations pour les futurs efforts d'optimisation et les mises à niveau du système.

En considérant ces applications, nous reconnaissons les implications plus larges de notre évaluation des performances et la valeur qu'elle apporte à divers domaines de IoT.

Pour la réalisation de ce test, nous avons suivi des procédures bien définies. Initialement, la passerelle envoie une requête vers le nœud de capteur. Le nœud répond par un acquittement pour indiquer qu'il a bien reçu la requête, puis il envoie les données demandées. Une fois les données reçues par la passerelle, celle-ci envoie à son tour un acquittement pour confirmer la réception des données. Ensuite, la passerelle attend un délai spécifique avant de lancer une nouvelle requête. Ce délai a été modifié à chaque test de durée de 20 minutes. La figure 3.3 illustre clairement cette procédure.

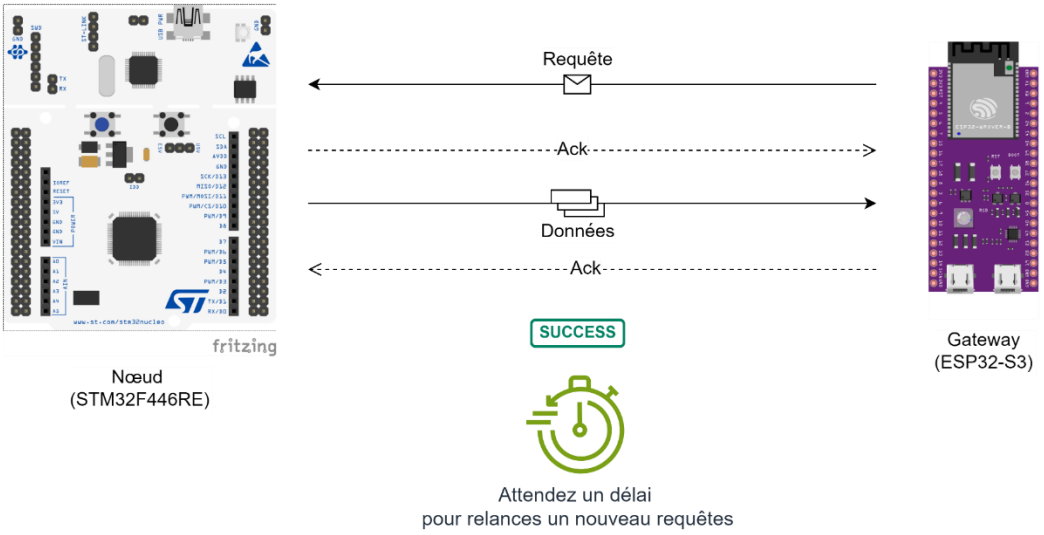

Figure 3. 3 : Schéma éxplicatif du fonctionnement de la passerelle duré de test avec un seul nœud

3.3.2 Résultat de test avant l'optimisation

Suivant la méthodologie que nous avons mentionné dans la partie précédente on fait ce premier test de performance qui se déroulera sans optimisation du code. Il maintiendra un débit de données fixe de 250 Kbps et une puissance de transmission de -18 dBm. Chaque itération de test se prolonger 20 minutes, en se concentrant sur la variation des délais entre les demandes.

La figure 3.4 illustre la corrélation entre l'intervalle de temps entre les demandes (mesuré en secondes) et le pourcentage de réponses réussies.

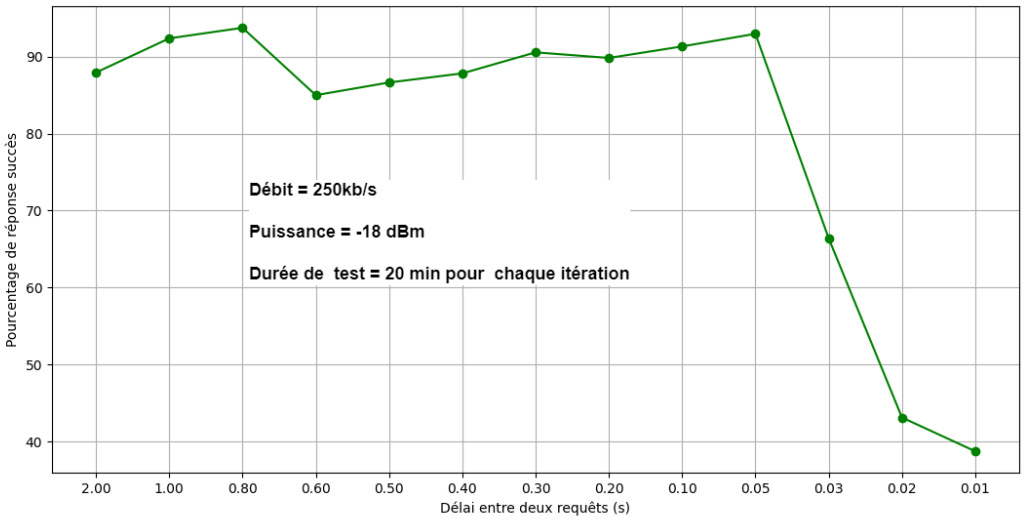

Figure 3. 4 : Pourcentage de réponses réussies en fonction de l'intervalle de temps entre les requêtes

L'analyse a montré qu'un délai de 50 ms marque le seuil pour atteindre un taux de réussite de 85 % ou plus, représentant la durée minimale entre les demandes. Cette valeur correspond à

l'état opérationnel optimal du système dans le cadre des paramètres examinés. Tout délai inférieur à 50 ms entraîne une diminution notable des taux de réussite, le taux tombant en dessous de 85 % avec un délai de 30 ms, ce qui met en évidence le point de divergence du système. Cette observation laisse entrevoir une surcharge potentielle du système ou des contraintes rencontrées lors du traitement des demandes à un régime accéléré.

Pour établir le seuil du nombre minimum de requêtes attendues de la passerelle, nous avons pris en compte le temps maximum d'extraction des données par requête, qui est de 100 ms. Nous avons ensuite pris en compte le délai entre les demandes, défini comme l'intervalle entre les transmissions. La somme de ces deux valeurs définit le temps de requêtes maximum. Enfin, en divisant la durée totale du test (20 minutes) par le " temps de requête maximum ", nous pouvons calculer la valeur seuil représentant le nombre minimum de requêtes devant être traitées au cours de la période de test.

$$seuil = \frac{20_{min} \times 60 \times 1000_{ms/min}}{100_{ms} + Délai\ d'attente}$$

Après avoir effectué les calculs, la figure 3.5 illustre le nombre minimum attendu de requêtes de la passerelle en fonction du délai entre les demandes.

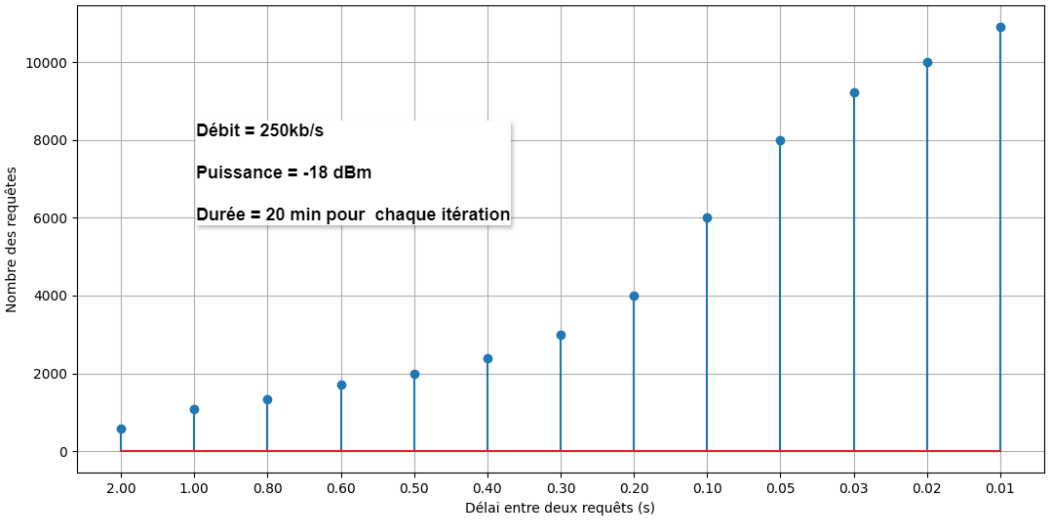

Figure 3. 5 : Minimum de requêtes attendues de la passerelle

Pour valider notre passerelle, nous devons comparer le nombre réel de requêtes (requêtes total) de notre test avec le seuil calculé à l'étape précédente. Pour ce faire, nous soustrayons la valeur du seuil aux résultats des demandes réelles.

La figure 3.6 illustre le nombre réel de demandes traitées par la passerelle. Pour plus de clarté, ce graphique représente à la fois le nombre de demandes réussies et le nombre de demandes

échouées en fonction du délai.

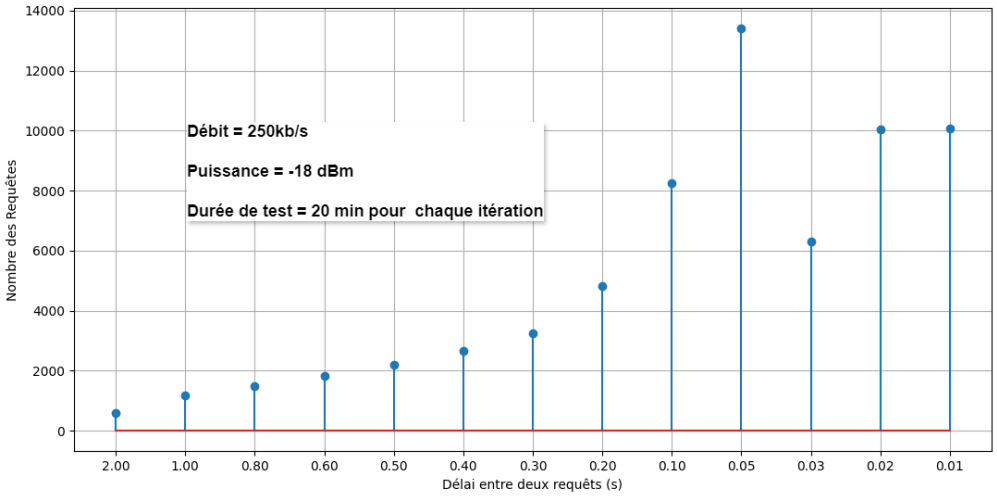

Figure 3. 6 : Total des requêtes envoyées par la passerelle

La figure 3.7 présente l'analyse comparative résultant de la soustraction du seuil du nombre réel de demandes, en fonction du délai.

$$Résultats\ de\ la\ comparaison = Requêtes\ total - Seuil$$

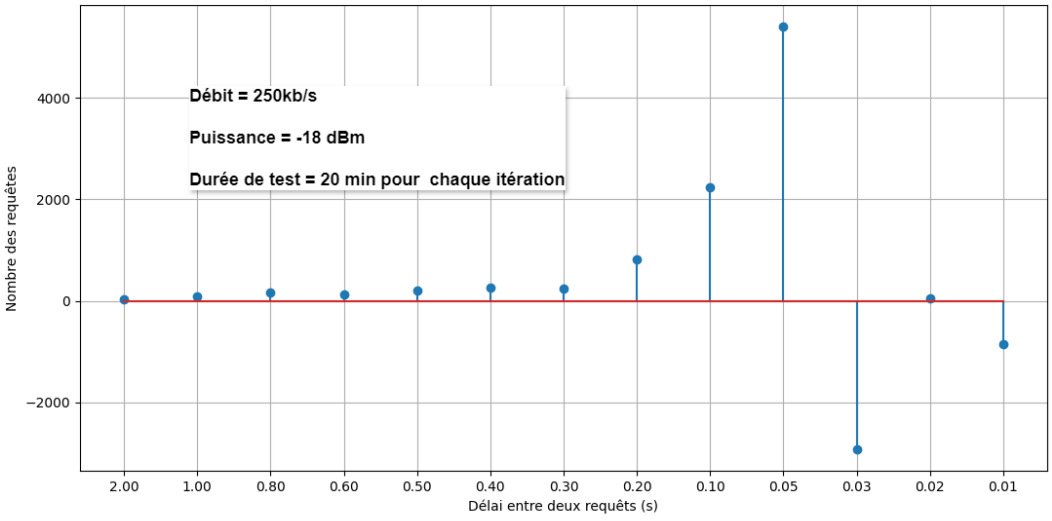

Figure 3. 7 : Comparaison des résultats des requêtes

Dans notre analyse, nous observons deux types de valeurs positives et négatives. Ces valeurs nous aident à différencier la région où notre passerelle fonctionne correctement (région de stabilité) de la région où le système ne répond pas aux performances attendues. Le graphique montre que la région de stabilité s'étend jusqu'à 50 ms. Au-delà de ce point, le système diverge et ne parvient pas à fournir les performances minimales attendues.

Nous visons à estimer la capacité maximale des nœuds que notre passerelle peut gérer efficacement, ce qui permet de mieux comprendre les limites de notre système. Pour ce faire, nous

avons fixé l'intervalle de temps entre les demandes du premier nœud à une minute, afin de garantir des intervalles de demande cohérents de la part du même esclave. Nous avons ensuite calculé la capacité maximale des nœuds à l'aide de l'équation suivante :

$$Nomber\ Maximum\ des\ noeuds = \frac{Intervalle\ entre\ les\ demandes\ \times\ Réussite\ de\ la\ réponse}{Durée\ de\ l'essai}$$

Après avoir effectué les calculs, la figure 3.8 illustre le nombre maximum de nœuds esclaves que le système peut traiter en une minute en fonction du délai entre les demandes.

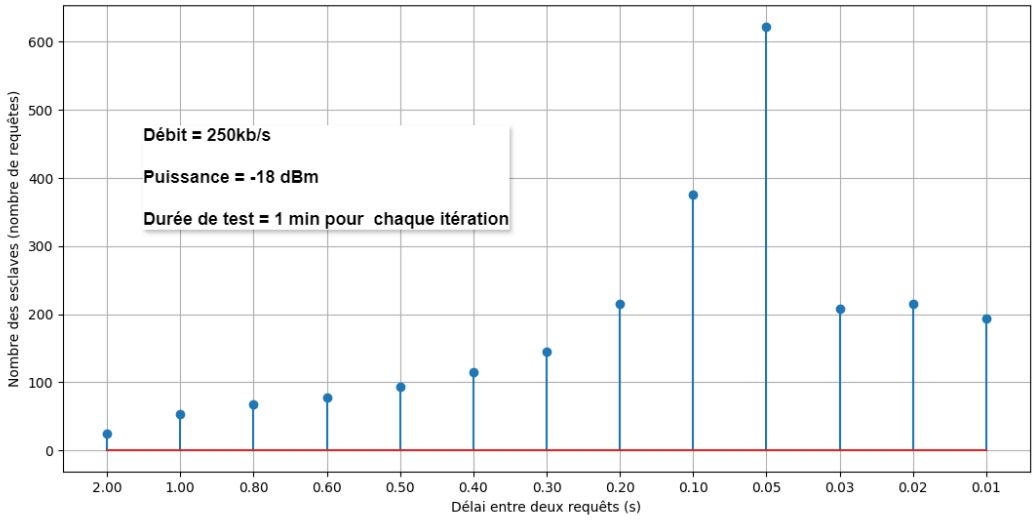

Figure 3. 8 : Nombre maximum d'esclaves que le système peut accepter

Dans notre analyse de ce graphique, nous observons qu'au moment où le délai diminue, le nombre de nœuds augmente dans la région de stabilité (où le délai est supérieur ou égal à 50 ms). Ce comportement correspond à nos attentes. Le graphique montre que la capacité maximale des nœuds est atteinte pour un délai de 50 ms, le système pouvant alors gérer jusqu'à 600 nœuds.

3.3.3 Résultat de test après optimisation

Dans cette section, nous reproduisons de manière précise chaque étape du test de performance initial, en commençant par l'étape d'ajustement de l'intervalle et en terminant par la détermination de la capacité maximale du nœud de notre passerelle. Les conditions de test restent constantes : un débit de 250 kbps et une puissance de transmission de -18 dBm. Chaque itération de test s'étendra sur une durée de 20 minutes. Cependant, le facteur distinctif est l'optimisation du code de notre système.

Nous avons optimisé le code pour minimiser la perte de données lors du premier transfert de paquets entre la passerelle et les nœuds de capteurs. Il s'agissait de trouver le moment idéal pour

que la passerelle passe en mode réception (RX) par rapport à la transition du capteur en mode transmission (TX). Nous avons mis en œuvre un mécanisme de retard contrôlé pour garantir la synchronisation des deux dispositifs lors de l'échange initial de données. Cette approche réduit efficacement le risque de manquer le premier paquet en raison de décalages temporels.

La figure 3.9 illustre la corrélation entre l'intervalle de temps entre les requêtes (mesuré en secondes) et le pourcentage de réponses réussies suite à l'optimisation du code.

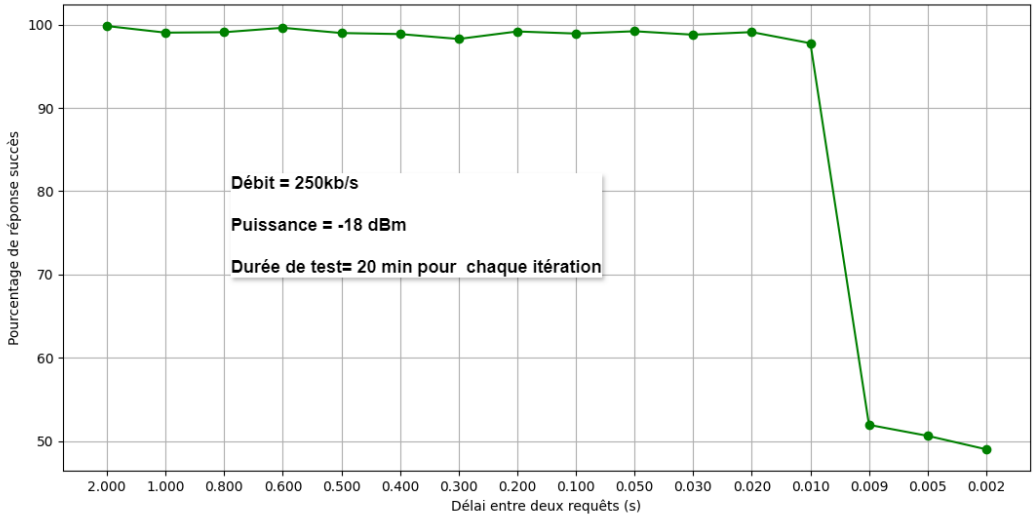

Figure 3. 9 : Pourcentage de réponses réussies (après optimisation du code)

Notre analyse révèle qu'un délai de 10 ms est nécessaire pour atteindre un taux de réussite de 97 % ou plus, ce qui indique l'écart de temps minimum entre les demandes pour une performance optimale. Cette valeur correspond à l'état opérationnel optimal du système en fonction des paramètres examinés. Les délais inférieurs à 10 ms entraînent une diminution significative des taux de succès ou de réponses réussies, qui chute en dessous de 85 % pour un délai de 9 ms, ce qui marque le point de divergence du système. Cette observation suggère une surcharge potentielle du système ou des contraintes lors du traitement des demandes à un rythme accéléré.

Nous avons effectué la même procédure pour le seuil que nous avons déjà fait avec le test avant l'optimisation. Après avoir effectué les calculs, la figure 3.10 illustre le nombre minimum attendu de requêtes de la passerelle en fonction du délai entre les requêtes.

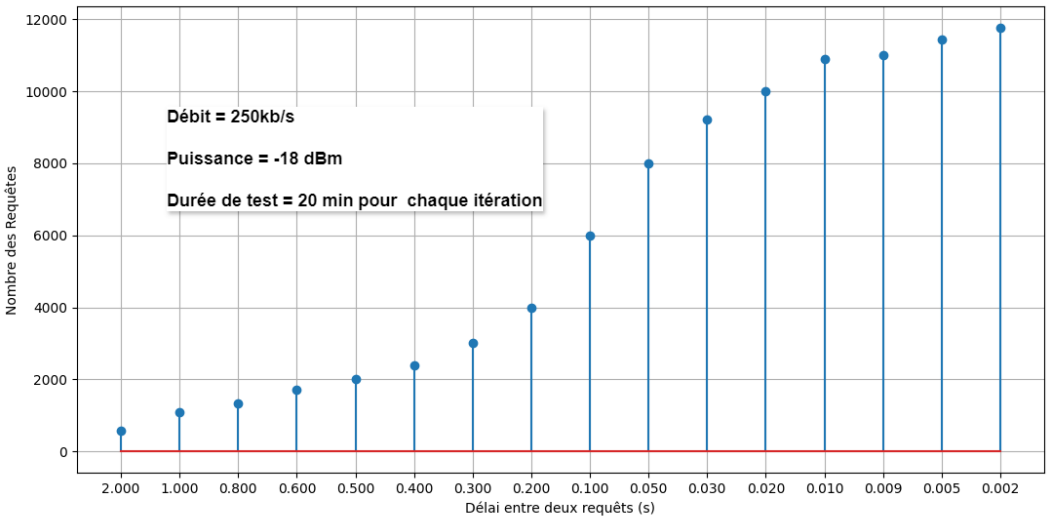

Figure 3. 10 : Minimum de demandes attendues de la passerelle

Nous effectuons une comparaison entre le nombre réel de requêtes ("Total des requêtes") de notre test et le seuil calculé à l'étape précédente. Cette comparaison est réalisée en soustrayant la valeur du seuil des résultats des demandes réelles. La figure 3.11 illustre le nombre réel de requêtes traitées par la passerelle. Pour plus de clarté, ce graphique représente à la fois le nombre de demandes réussies et le nombre de requêtes échouées en fonction du temps d'attente.

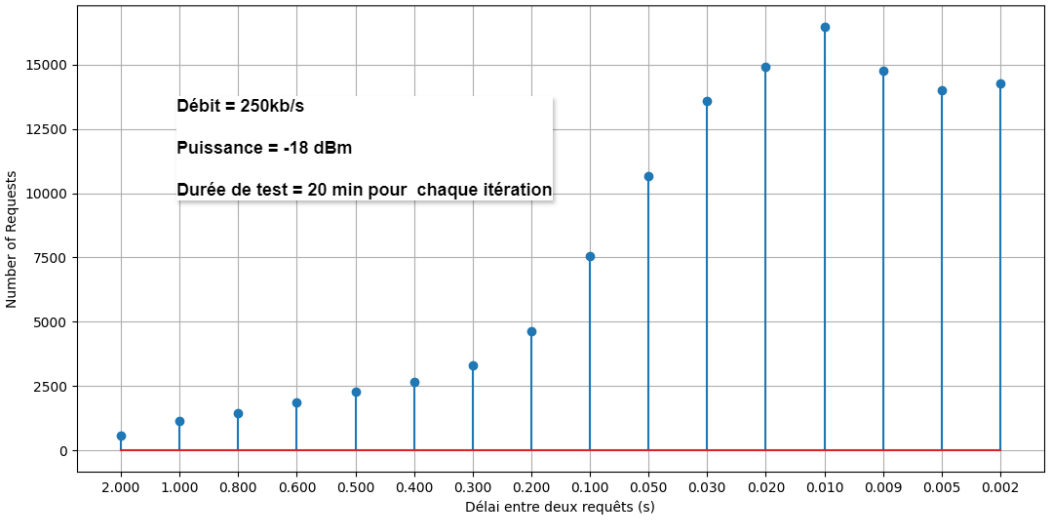

Figure 3. 11 : Total des requêtes envoyées par la passerelle après optimisation du code

De même que nous avons comparé le nombre réel de requêtes avec le seuil avant l'optimisation, nous faisons une nouvelle comparaison pour évaluer l'efficacité de notre passerelle IoT après l'optimisation. La figure 3.12 présente l'analyse comparative résultant de la soustraction du seuil du nombre réel de requêtes, en fonction du délai.

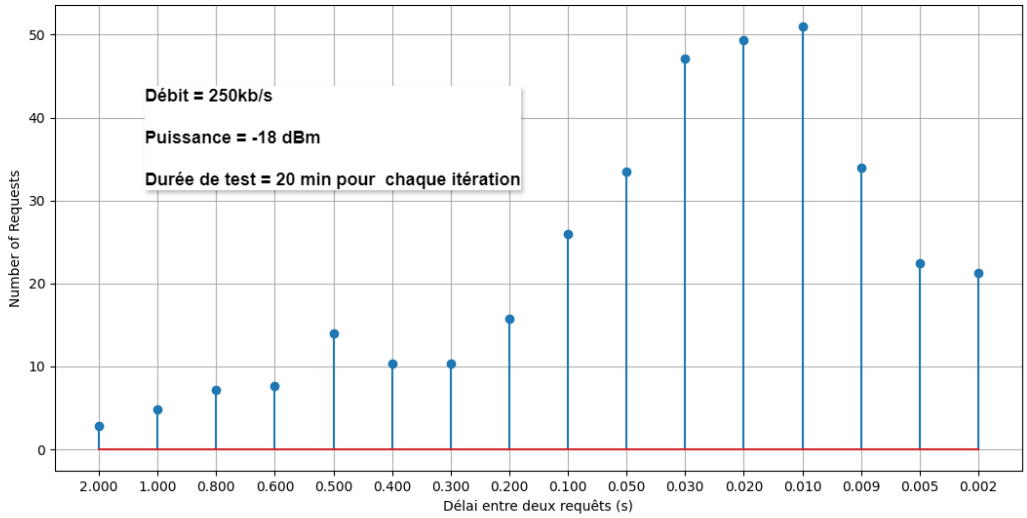

Figure 3. 12 : Résultats de la comparaison des requêtes après optimisation du code

Initialement, tous les résultats des comparaisons sont positifs, d'où, résultats favorables. Cependant, après la 13ème expérience, lorsque le délai est égal à 10 ms, nous constatons une diminution des performances. Cela suggère que le système entre dans une zone de divergence.

Nous estimons la capacité maximale de nœuds que notre passerelle peut supporter après l'optimisation du code pour mieux comprendre les limites du système. Nous avons fixé l'intervalle de temps entre les demandes du premier nœud à une minute, ce qui garantit des intervalles cohérents. La capacité maximale des nœuds a ensuite été calculée. La figure 3.13 illustre le nombre maximal de nœuds esclaves que le système peut gérer en une minute après l'optimisation du code, en fonction du délai entre les demandes.

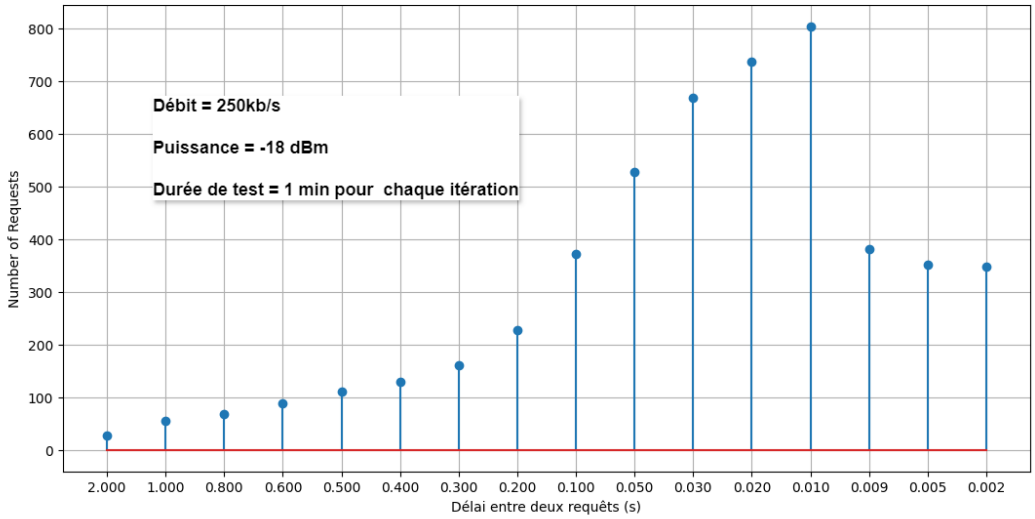

Figure 3. 13 : Nombre maximum des nœuds acceptés après optimisation du code

En examinant le graphique, nous constatons qu'au moment où le délai diminue, le nombre

de nœuds augmente dans la région de stabilité, en particulier lorsque le délai est supérieur ou égal à 10 ms. Cette tendance correspond au comportement que nous avions anticipé. Fait remarquable, le graphique montre que la capacité maximale des nœuds est atteinte avec un délai de 10 ms, ce qui permet au système de gérer efficacement jusqu'à 800 nœuds.

3.4 Comparaison des tests avant et après l'optimisation

Cette section se lance dans une comparaison complète entre les résultats des tests de performance obtenus par notre passerelle avant et après l'optimisation du code. En examinant méticuleusement ces deux ensembles de données. Notre objectif est de montrer l'impact réel de l'optimisation du code sur les performances globales de notre passerelle. Grâce à cette analyse comparative, nous nous efforçons de mieux comprendre l'efficacité des efforts d'optimisation et de vérifier les implications réelles pour la fonctionnalité de notre système.

— **Le taux de succès et le taux d'erreur :** sont deux mesures essentielles dans ce test, car ils sont inversement liés ; lorsque le taux de succès augmente, le taux d'erreur diminue. Les deux tests nous ont permis d'observer un impact significatif de l'optimisation de notre code sur le taux de succès. Après optimisation, le taux de succès se situe entre 99,8 % et 97,76 %, contre un taux de 92,97 % à 87,94 % avant optimisation. Ces changements se sont produits dans la zone de stabilité de notre passerelle, soulignant l'efficacité des optimisations.

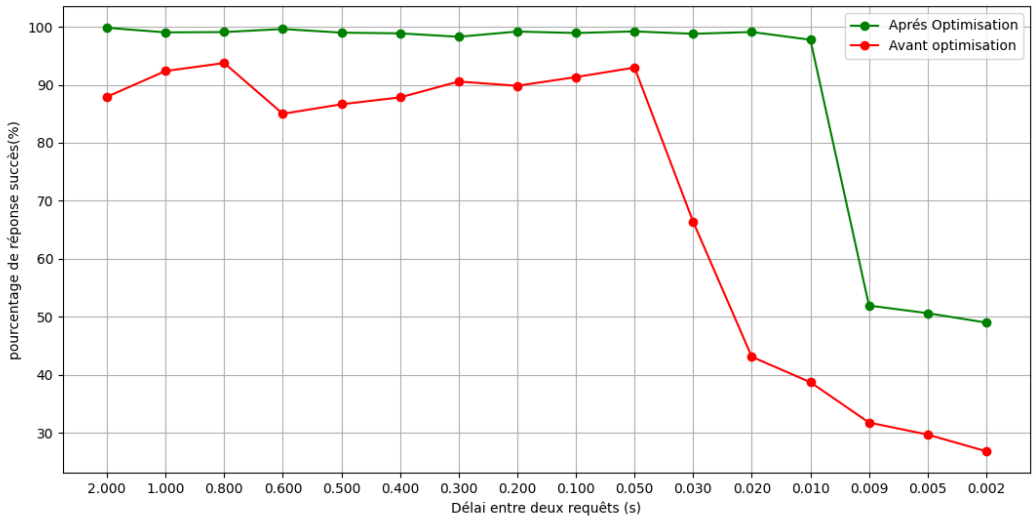

Figure 3. 14 : Comparaisons du taux de réussite avant et après optimisation

— **Le temps de réponse et la plage acceptée :** sont deux autres mesures importantes dans ce test, car elles sont inversement liées ; lorsque le temps de réponse diminue, la région de stabilité augmente, ce qui est le résultat que nous souhaitons obtenir. Notre analyse montre clairement un impact significatif de l'optimisation du code sur le temps de réponse, le réduisant de plus

de cinq fois par rapport à la limite avant la mise en œuvre de l'optimisation du code. Plus précisément, le temps de réponse limite est passé de 50 ms avant l'optimisation du code à 10 ms après l'optimisation, ce qui démontre une amélioration substantielle des performances de notre passerelle.

— **L'augmentation de la capacité maximale des nœuds de notre passerelle** : est l'objectif principal de ce test. Notre analyse révèle que l'optimisation du code a eu un impact significatif sur cette mesure. Après optimisation, la capacité maximale de nœuds est passée de 622 nœuds à 804 nœuds, ce qui indique que notre passerelle peut désormais gérer plus de nœuds en une minute. Cette amélioration souligne l'efficacité des optimisations mises en œuvre.

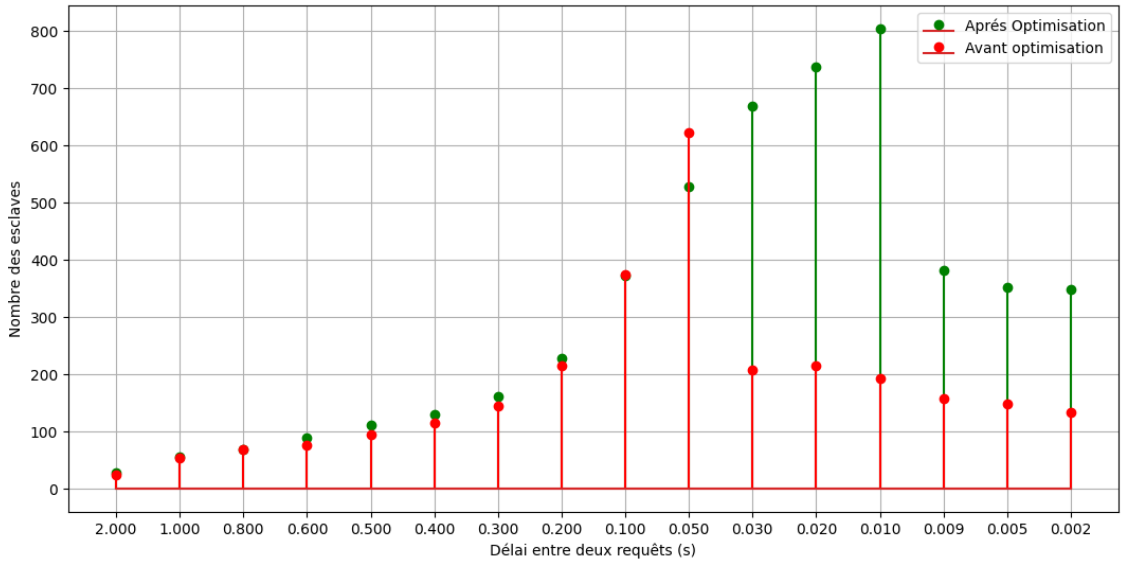

Figure 3. 15 : Comparaisons de la capacité maximal en nœuds avant et après optimisation

Le tableau ci-dessous illustre les résultats de la comparaison :

Tableau 3. 1 : Comparaison des mesures avant et après l'optimisation

Paramètres	Avant l'optimisation	Après optimisation
Taux de réussite	92.97% - 87.94%	99.8% - 97.76%
Délai minimal (ms)	50	10
Capacité maximale des nœuds	622 nodes	804 nodes

Ces comparaisons et résultats montrent clairement que l'optimisation du code a eu un impact positif sur notre passerelle, améliorant ainsi considérablement ses performances et les amenant à un très bon niveau.

3.5 Test à long-termes

Sur la base des tests de performance initiaux et de l'identification de la région de stabilité pour notre passerelle IIoT, nous entreprenons un test de suivi pour évaluer l'impact de la durée du test sur le taux de réussite. L'objectif de ce test est d'évaluer la stabilité et la fiabilité à long terme de la passerelle IIoT en observant le taux de réussite sur de longues périodes. Cette évaluation garantit que la passerelle peut maintenir des performances constantes dans la région de stabilité identifiée, ce qui est crucial pour les déploiements dans le monde réel.

L'objectif de ce test est d'évaluer la stabilité et la fiabilité à long terme de la passerelle IoT en observant le taux de réussite sur de longues périodes. Cette évaluation garantit que la passerelle peut maintenir des performances constantes dans la région de stabilité identifiée, ce qui est très important pour les déploiements dans le monde réel.

3.4.1 Méthodologie d'évaluation de la durabilité

Pour évaluer la durabilité de notre passerelle IoT, nous avons adopté une méthodologie qui suit. Tout d'abord, nous avons sélectionné un délai dans la région de stabilité déterminée précédemment. Ensuite, nous avons effectué un essai prolongé sur une période prolongée, tout en conservant le délai sélectionné. Pendant cette période, nous avons contrôlé et enregistré en permanence le taux de réussite de la passerelle. Finalement, nous avons analysé les données collectées pour identifier toute variation ou tendance du taux de réussite au fil du temps.

3.4.2 Résultats prévus et vises

— **Performances constantes :** Le taux de réussite doit rester stable dans la plage acceptable, ce qui indique un fonctionnement fiable à long terme.

— **Identification des problèmes :** Toute déviation ou baisse significative du taux de réussite sera identifiée, ce qui permettra de déceler d'éventuels problèmes de performance à long terme.

Ce test est essentiel pour valider les performances de la passerelle IoT dans des scénarios réels où un fonctionnement continu et fiable est nécessaire. Il garantit que la passerelle maintienne des taux de réussite élevés sur des périodes prolongées, confirmant son aptitude à fonctionner dans des environnements persistants et exigeants ou hostiles. En démontrant des performances durables, ce test confirme que la passerelle est prête à être déployée dans des applications pratiques, où la stabilité à long terme est essentielle.

Dans ce test, nous avons maintenu les mêmes conditions que le test de performance initial, à savoir un débit de données de 250 kbps et une puissance de transmission de -18 dBm. Nous

avons choisi un délai de 30 ms entre deux requêtes, en veillant à ce que ce délai se situe dans la zone de stabilité identifiée précédemment. La durée de ce test a été fixée à 1,5 heure.

Le graphique dans la figure 3.16 illustre la variation du taux de réussite des demandes de données dans le temps.

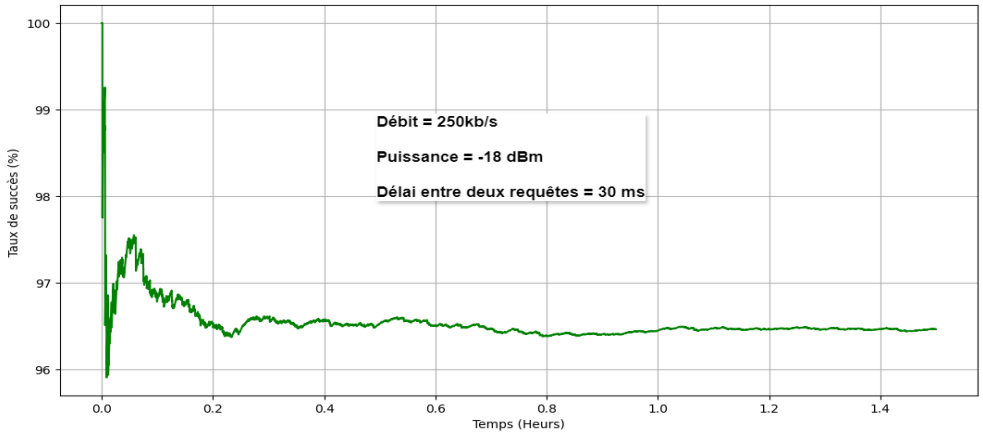

Figure 3. 16 : Pourcentage de réponses réussies

Dans notre analyse du graphique, nous avons observé deux régions temporelles distinctes. La première région, qui s'étend de 0 à 0,2 heure, présente une variation significative du taux de réussite, comprise entre 100 % et 95,9 %. Cependant, cette variation a cessé lorsque nous sommes entrés dans la deuxième région temporelle, qui s'étend de 0,2 heure à la fin du test. Au cours de cette période, nous avons observé un taux de réussite stable autour de 96,5 %. Cette stabilité durable indique la robustesse et la fiabilité de notre système.

Notre passerelle IoT fait preuve d'une stabilité et d'une fiabilité à long terme impressionnantes, en maintenant un taux de réussite constant d'environ 96,5 % pendant toute la durée du test. Cette performance valide son adéquation aux déploiements dans le monde réel, en soulignant sa robustesse à fournir un fonctionnement fiable sur des périodes prolongées.

3.6 Effet de l'ajout des nœuds de capteurs

Nous étudions l'impact des nœuds de capteurs multiples sur les performances globales du système. Nous analysons comment l'ajout de plusieurs nœuds affecte la latence, le débit, la fiabilité, les interférences, les collisions et les stratégies d'optimisation des performances du réseau.

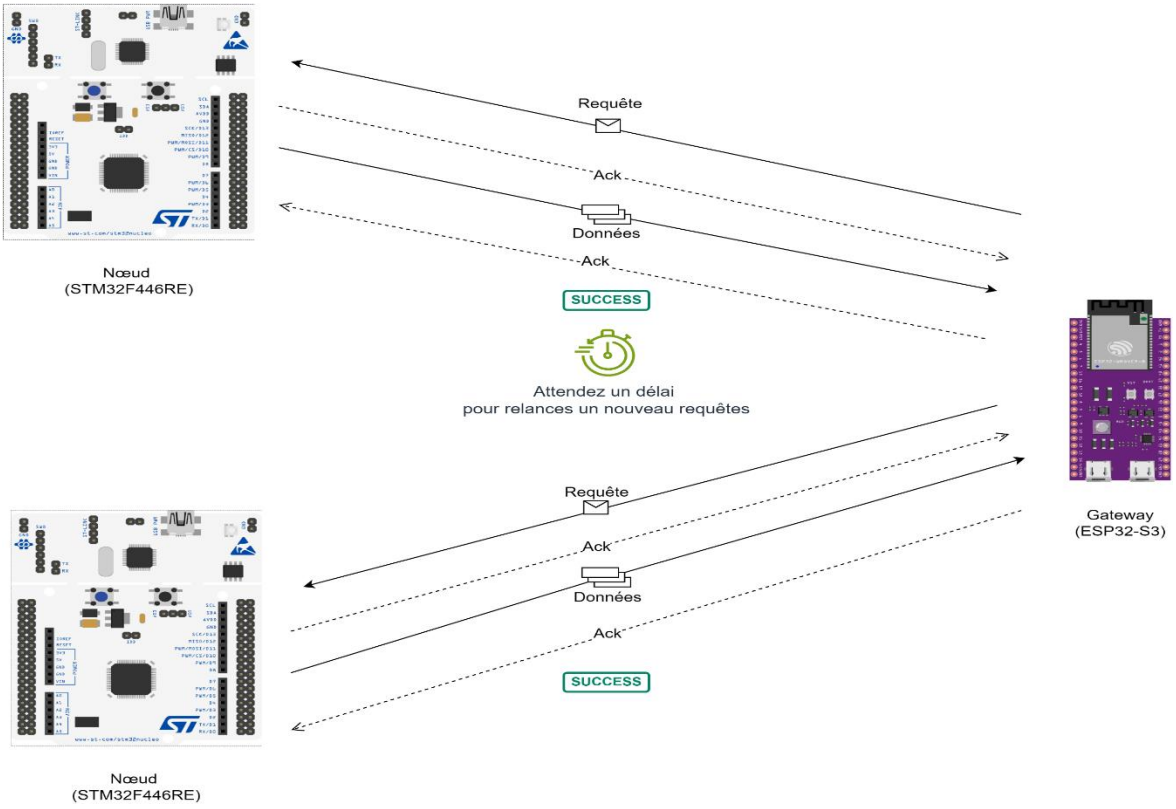

Figure 3. 17 : Schéma explicatif

Dans ce test, nous conservons les mêmes paramètres de configuration que ceux utilisés dans les tests précédents. Toutefois, nous introduisons un nœud supplémentaire dont la configuration est identique à celle du premier, comme le montre la figure 3.17. La durée du test s'étend sur une période de 2 heures, avec un délai de 1 seconde entre chaque cycle de transmission. Après avoir mené ce test, nous avons collecté et analysé les données obtenues pour évaluer précisément l'impact de l'ajout de ce nœud de capteur supplémentaire. La figure 3.18 illustre le résultat généré pendant le test.

```
//*_____________________________________________________*//
Test AckPayload !! : Ack by Node 0
Slave 0     = 90
slave 1     = 85
Data        = 175
Total       = 179
My Data     = Hello World 0 !!
//*_____________________________________________________*//
Test AckPayload !! : Ack by Node 1
Slave 0     = 90
slave 1     = 86
Data        = 176
Total       = 180
My Data     = Hello World 1 !!
//*_____________________________________________________*//
```

Figure 3. 18 : Résultat générée pendant le test

Le tableau 3.2 présente les résultats obtenus à l'issue de nos tests exhaustifs

Tableau 3. 2 : Résultat des différents tests obtenus à partir des expériences réalisées

Elément	Réponses aux requêtes	Pourcentage du taux de réussite
Nœud 1	3432	99.7 %
Nœud 2	3390	98.4 %
Requêtes de données	6822	99.02 %
Total	6884	-

L'analyse du tableau démontre que notre passerelle a lancé 6884 requêtes de données aux deux nœuds esclaves et a reçu 6822 réponses, ce qui représente un taux de succès impressionnant de 99,09 %. Les taux de succès des nœuds esclaves individuels sont remarquablement similaires, ne différant que de 1,3 %.

Sur la base de ces résultats, nous pouvons conclure que le système présente un niveau élevé de fiabilité et des performances équilibrées. Ces résultats constituent une preuve solide de l'efficacité et de la robustesse de la conception de notre passerelle pour gérer efficacement la communication avec plusieurs nœuds.

3.7 Conclusion

En conclusion, ce chapitre a mis en évidence l'importance de la validation des tests pour garantir la fiabilité, la précision et la performance de notre passerelle IIoT. Les tests réalisés ont permis de vérifier le bon fonctionnement de la passerelle et de valider sa capacité à gérer plusieurs nœuds de capteurs. Les résultats obtenus sont vraiment d’une valeur significative, car ils nous permettent de comprendre en profondeur le comportement de la passerelle dans le domaine de l'IIoT et d'identifier d'éventuels problèmes ou limitations. Cela nous permet également d'ajuster et d'optimiser les algorithmes et les paramètres pour améliorer les performances globales de la passerelle. En résumé, les tests et les résultats obtenus sont conformes à nos attentes, ce qui nous permet de fournir un outil fiable, précis et adapté aux besoins spécifiques de l'IIoT.

Conclusion générale

Conclusion générale

Ce projet nous a permis de concevoir et de réaliser une passerelle pour l'internet industriel des objets (IIoT) répondant aux exigences spécifiques des environnements industriels modernes.

Notre passerelle, basée sur des technologies de microcontrôleurs ESP32 pour la passerelle principale et STM32F4 pour les nœuds de capteurs, a été conçue pour assurer une communication fiable et efficace entre les différents éléments du réseau IIoT. Les nœuds de capteurs sont capables de collecter et de transmettre des données en temps réel, permettant ainsi une surveillance continue et une gestion optimisée des processus industriels.

La conception de la passerelle IIoT a été réalisée en plusieurs étapes clés, chacune marquée par des défis techniques spécifiques. Le choix des microcontrôleurs a été déterminant pour garantir une performance élevée et une faible consommation d'énergie. Les tests et les validations dans le développement de la passerelle, permettant d'identifier et de résoudre les problèmes de communication et d'intégration des nœuds de capteurs.

Les résultats expérimentaux ont démontré la robustesse et l'efficacité de notre passerelle. Les tests de performance ont confirmé la capacité de la passerelle à gérer plusieurs nœuds de capteurs simultanément, avec un taux de succès élevé relative au nombre de réponses aux requêtes envoyées, et une latence minimale. Les optimisations du code effectuées ont permis d'améliorer significativement la performance globale du système, réduisant significativement les temps de réponse et augmentant efficacement la capacité maximale des nœuds.

En outre, l'intégration de la passerelle avec des plateformes cloud via le protocole MQTT qui a constituée effectivement notre réseau global et réel IoT et ayant été réalisé avec grand succès, a été considéré comme étant le point clé d’aboutissement de notre projet répondant au cahier des charges de notre projet qui nous a été exigé dès le départ, permettant ainsi une transmission sécurisée et efficace des données collectées vers des services de stockage et d'analyse à distance.

Cette fonctionnalité est essentielle pour les applications de maintenance prédictive, de suivi des actifs et de surveillance à distance, qui nécessitent des données en temps réel pour une prise de décision rapide et éclairée.

Ce projet a, non seulement atteint ses objectifs initiaux, mais a également ouvert la voie à de futures améliorations et expansions. La passerelle IIoT conçue est un outil puissant et flexible, capable de répondre aux besoins variés des environnements industriels modernes.

Les connaissances acquises via les différentes technologies développées et utilisées dans le cadre de ce projet offrent une base solide pour de futurs travaux de recherche et de développement

dans le domaine de l'Internet industriel des objets. Les résultats obtenus témoignent de l'importance des passerelles IoT dans le cadre et le but de la transformation numérique des industries, offrant des solutions innovantes pour améliorer l'efficacité opérationnelle, réduisant ainsi les coûts et stimulant de l'innovation.

Annex 1

A.1 Installation du ESP-IDF :

Cette section a pour but de vous aider à configurer l'environnement de développement logiciel pour le matériel basé sur la puce ESP32 S3 d'Espressif. Ensuite, un exemple simple vous montrera comment utiliser ESP-IDF (Espressif IoT Development Framework) pour la configuration des menus, puis pour construire et flasher le micrologiciel sur une carte ESP32-S3.

Espressif fournit des ressources matérielles et logicielles de base pour aider les développeurs d'applications à concrétiser leurs idées en utilisant le matériel de la série ESP32-S3. Le cadre de développement logiciel d'Espressif est destiné au développement d'applications de l'Internet des objets (IoT) avec Wi-Fi, Bluetooth, gestion de l'alimentation et plusieurs autres fonctionnalités du système.

Pour commencer à utiliser l'ESP-IDF sur l'ESP32-S3, installez le logiciel suivant [22] :

— Toolchain pour compiler le code pour l'ESP32-S3
— Outils de construction - CMake et Ninja pour construire une application complète pour l'ESP32-S3
— ESP-IDF qui contient essentiellement l'API (bibliothèques logicielles et code source) pour l'ESP32-S3 et les scripts pour faire fonctionner la chaîne d'outils.

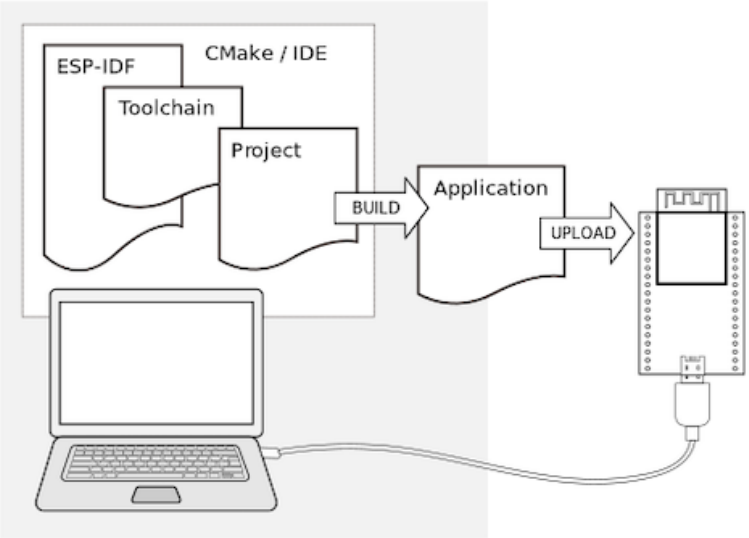

Figure A. 1 : Schéma du processus de développement ESP32-S3 [22]

Afin d'utiliser l'ESP-IDF avec l'ESP32-S3, vous devez installer certains logiciels en fonction de votre système d'exploitation. Ce guide d'installation vous aide à tout installer sur les systèmes basés sur Linux. Pour installer ESP-IDF sur Linux Ubuntu, suivez les étapes suivantes :

A.1.1 Installer les prérequis :

Ouvrez un terminal et exécutez la commande suivante pour installer les paquets nécessaires :

```
sudo apt-get install git wget flex bison gperf python3 python3-pip python3-venv cmake ninja-build ccache libffi-dev libssl-dev dfu-util libusb-1.0-0
```

A.1.2 Cloner le dépôt ESP-IDF :

Créez un répertoire pour ESP-IDF et clonez le dépôt avec Git :

```
mkdir -p ~/esp
cd ~/esp
git clone --recursive https://github.com/espressif/esp-
```

A.1.3 Installer les outils :

Exécutez le script d'installation pour installer les outils nécessaires :

```
cd ~/esp/esp-idf
./install.sh esp32s3
```

A.1.4 Configurer les variables d'environnement :

Exécutez le script d'exportation pour configurer les variables d'environnement :

```
.$HOME/esp/esp-idf/export.sh
```

A.1.5 Vérifier l'installation :

Pour vérifier que tout fonctionne correctement, exécutez la commande suivante :

```
idf.py --version
```

A.1.6 Configurer le projet :

Pour configurer le projet, exécutez la commande suivante :

```
idf.py set-target esp32s3

idf.py menuconfig
```

A.1.7 Bâtir et flasher le projet :

Pour bâtir et flasher le projet, exécutez la commande suivante :

```
idf.py build

idf.py -p PORT flash
```

Note : Assurez-vous de remplacer PORT par le nom du port série de votre ESP32-S3.

Bibliographie

[1] A. Yarali, M. Srinath, and R. G. Joyce, "A Study of Various Network Security Challenges in the Internet of Things (IoT)," 2018.

[2] C. Reichel, "Internet of Things (IoT): Emphasizing Its Applications and Emergence in Environmental Management—The Profound Cases," in Book Title, Editor(s), Edition ed. Publisher, City, State, Country, 2022, pp. 201-212.

[3] P. Nguyen-Hoang and P. Vo-Tan, "Development An Open-Source Industrial IoT Gateway," in 2019 19th International Symposium on Communications and Information Technologies (ISCIT), Ho Chi Minh City, Vietnam, 2019, pp. 201-204, doi: 10.1109/ISCIT.2019.8905157.

[4] Lekic, M., & Gardasevic, G. (2018). IoT sensor integration to Node-RED platform. 2018 7 th International Symposium INFOTEH-JAHORINA (INFOTEH). https://doi.org/10.1109/infoteh.2018.8345544

[5] K.K. Patel, S. M. Patel, and P. Scholar. Internet of things-iot : definition, characteristics, architecture, enabling technologies, application & future challenges. International Journal of Engineering Science and Computing, 6(5), 2016.

[6] H. SAADI, « cours de l'internet des objets » pour les 5eme année ingénieurs, ENSTICP, Algérie, 2023

[7] Dorsemaine, Bruno, et al. "Internet of things: a definition & taxonomy." 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies. IEEE, 2015.

[8] Yassine Haddab. Introduction a l'internet des objets (IdO – IoT). Universite de Montpellier.

[9] Architecture IoT : L'essentiel a savoir," IoT Industriel, 23 nov. 2022. [En ligne]. Disponible sur : https://iotindustriel.com/iot-iiot/architecture-iot-lessentiel-a-savoir/. [Consulte le 14 juin 2024].

[10] Sikder, A.K., Petracca, G., Aksu, H., Jaeger, T., & Uluagac, A.S. (2018). A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications. arXiv, abs/1802.02041.

[11] P. Nguyen-Hoang and P. Vo-Tan, "Development an open-source industrial IOT gateway," 2019 19th International Symposium on Communications and Information

TECHNOLOGIES (ISCIT), SEP. 2019. DOI :10.1109/ISCIT.2019.8905157

[12] LIU, X., ZHANG, T., HU, N., ZHANG, P., & ZHANG, Y. (2020). THE METHOD OF INTERNET OF THINGS ACCESS AND NETWORK COMMUNICATION BASED ON MQTT. COMPUTER COMMUNICATIONS, 153, 169-176.

[13] PATTI, G., LEONARDI, L., TESTA, G., & BELLO, L. L. (2024). PRIOMQTT: A PRIORITIZED VERSION OF THE MQTT PROTOCOL. COMPUTER COMMUNICATIONS, 220, 43-51.

[14] N. Kolban, *Kolban's Book on ESP32*, pp. 60-80, Septembre 2018.

[15] Espressif Systems. "ESP32-S3 Series Datasheet." Espressif Systems, 2021, https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

[16] ESPRESSIF SYSTEMS, "USER GUIDE FOR DEVKITC-1 WITH ESP32-S3," IN *ESP-IDF PROGRAMMING GUIDE*, 2023. [ONLINE]. AVAILABLE: HTTPS://DOCS.ESPRESSIF.COM/PROJECTS/ESP-IDF/EN/LATEST/ESP32S3/HW REFERENCE/ESP32S3/USER-GUIDE-DEVKITC-1.HTML. [ACCESSED: APR. 10, 2023].

[17] A step-by-step guide to the most complete ARM Cortex-M platform, using a free and powerful development environment based on Eclipse and GCC, page 21-

[18] S. T. Microelectronics. "Datasheet - STM32F446xC/E - Arm Cortex-M4 32-bit MCU+FPU, 225 MHz, 512 KB Flash, 128 KB RAM, 12-bit ADC 1.7 V to 3.6 V Operating Voltage." STMicroelectronics, 22 janv. 2021, https://www.st.com/resource/en/datasheet/stm32f446mc.pdf.

[19] Nordic Semiconductor. "nRF24L01+ Datasheet." Nordic Semiconductor, 2021, https://www.nordicsemi.com/eng/products/2.4ghz-rf/nrf24l01p.

[20] Ayati, A., & Naji, H. R. (2023). A security mechanism for Enhanced ShockBurst wireless communication protocol using nRF24L01.

[21] R. Barry, *Mastering the FreeRTOS™ Real Time Kernel: A Hands-On Tutorial Guide*, Pre-release 161204 Edition. Real Time Engineers Ltd., 2016.

[22] ESPRESSIF SYSTEMS. "GET STARTED - ESP32-S3." ESP-IDF PROGRAMMING GUIDE, 2023. [ONLINE]. AVAILABLE: HTTPS://DOCS.ESPRESSIF.COM/PROJECTS/ESP-IDF/EN/LATEST/ESP32S3/GET-STARTED/INDEX.HTML.