الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière : Électronique
Spécialité : Electronique des systèmes embarqués

Présenté par

Assaous Oussama

&

Brahimi Sidali

Etude et implémentation du protocole industriel Modbus TCP/IP pour les I/O devices , avec conception et réalisation d'un I/O device

Proposé par : Larfi Abdelmoumen & Naceur Djamila

Année Universitaire 2020-2021

Remerciements

Nos vifs remerciements vont dans un premier temps à notre encadreur D.Naceur, pour sa confiance, sa disponibilité et son suivi.

Nous remercions également M. LARFI Abdelmoumen et M. REBIKA Samir ainsi que toute l'équipe de DOOFAS pour leur disponibilité, leurs précieux conseils et pour nous avoir introduit au monde professionnel.

Nous remercions BOUCETTA Belaid technicien dans la maintenance des smartphones qui nous aide de faire le soudage des composants

ملخص: في غالبية المنشآت الصناعية، يوجد عدد كبير من أجهزة الاستشعار والمشغلات من أجل مراقبة الكميات المادية للعمليات والتحكم فيها، يتطلب نشر مستشعرات المشغل هذه كميات من بطاقات الاقتناء حساسة للتدخل (أو غيرها) وكابلات أطول. شبكة تناظرية.

الهدف من هذا العمل هو تقليل تكاليف وتعقيد هذه التركيبات من خلال تصميم جهاز إدخال / إخراج إيثرنت ثنائي المنافذ لإنشاء شبكة حلقة أو نجمة تتواصل باستخدام بروتوكول اتصال Modbus TCP / IP.

كلمات المفاتيح:

Résumé : Dans la majorité des installations industrielles, on trouve un nombre important de capteurs et d'actionneurs afin de surveiller et de contrôler les grandeurs physiques des procédés, le déploiement de ces capteurs actionneurs requièrent des quantités de cartes d'acquisition et des longueurs de câbles plus un réseau analogique (4-20mA ou autre) sensible au interférences.

Le but de ce travail est de minimiser les coûts et la complexité de ces installations par la conception et réalisation d'un I/O device dual port Ethernet pour réaliser un réseau en anneau ou en étoile communiquant à l'aide du protocole de communication Modbus TCP/IP.

Mots clés : Modbus TCP/IP ; Ethernet ; modules d'E/S

Abstract: In the majority of industrial installations, there is a large number of sensors and actuators in order to monitor and control the physical quantities of the processes, the deployment of these actuator sensors requires quantities of acquisition cards and longer length cables. an analog network (4-20mA or other) sensitive to interference.

The goal of this work is to minimize the costs and complexity of these installations by designing a dual port Ethernet I / O device to create a ring or star network communicating using the Modbus TCP / IP communication protocol.

Keywords : Modbus TCP/IP ; Ethernet ; I/O device

Listes des acronymes et abréviations

PLC : ou (API) Automate Programmable Industriel

BOOTP: Bootstrap Protocol

TCP: Transmission Control Protocol

IP: Internet Protocol

PoE: power over Ethernet

PCB: Printed circuit board

PCBA: Printed Circuit Board Assembly

PHY: physical layer of the Open Systems Interconnection

EMI: Electromagnetic interference

4ppoe: 4-Pair Power Over Ethernet

MCU: microcontroller unit

Table des matières

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Liste des tableaux

Chapitre 1

Chapitre 2

Introduction générale

Au cours des dernières années, le développement de la technologie industrielle implique un passage de la technologie analogique vers la technologie numérique, la transition vers le numérique a été la priorité des services publics, entreprises et industries, vu les avantages qu'offre la digitalisation de nos informations. Il a été donc nécessaire d'adapter les équipements et machines afin d'assurer la possibilité de traitement de données numériques et de communication à l'aide de divers protocoles qui nous permettent le transfert et l'acquisition d'informations. Ainsi l'introduction de capteurs et d'actionneurs afin de surveiller et de contrôler les grandeurs physiques selon les protocoles de communication permettant aux dispositifs électroniques, représentés sous forme de système embarqué d'être interconnectés aux différents périphériques dans un réseau de communication.

Ce mémoire présente une étude ainsi que l'implémentation et la conception d'un IO device communiquant à l'aide du protocole Modbus TCP/IP. Il présente également les tests effectués sur cet IO device et les résultats obtenus

Le mémoire est organisé comme suit :

Dans le chapitre 1, on introduit le protocole Modbus TCP/IP et sa fonctionnalité

Le chapitre 2, est consacré à l'implémentation Hardware et software du protocole industriel Modbus TCP/IP.

Le chapitre 3 est dédié à la conception et la réalisation du PCB.

Et enfin les tests Hardware et software ainsi que les résultats obtenus sont discutés au chapitre 4

Chapitre 1 Etude du protocole Modbus TCP/IP

1.1 Introduction

Le modbus TCP/IP est un Protocole de communication qui utilise le modèle TCP/IP qui assure la communication client/serveur entre les appareils connectés sur différents types de bus ou réseaux pour transmettre les messages modbus à travers Ethernet.

Le modbus est un Protocole de messagerie qui réside sur la couche application du modèle TCP/IP

1.2 Modbus

Le protocole Modbus a été développé en 1979 par Modicon, pour les systèmes d'automatisation industrielle et les automates programmables Modicon. Il est depuis devenu une méthode standard de l'industrie pour le transfert des informations d'E/S analogiques et des données d'enregistrement entre le contrôle industriel et les dispositifs de surveillance. Modbus est désormais un domaine public largement accepté, protocole ouvert qui nécessite une licence, mais n'exige pas le paiement en contrepartie de son utilisation à son propriétaire. [1]

Les appareils Modbus communiquent à l'aide d'un maître-esclave (client-serveur) technique dans lequel un seul appareil (le maître/client) peut initier les transactions (appelées requêtes). Les autres appareils (esclaves/serveurs) répondent au maître en lui fournissant les données demandées, ou en prenant l'action demandé dans la requête. Un esclave est tout périphérique (transducteur d'E/S, vanne, lecteur réseau ou autre appareil de mesure) qui traite les informations et envoie la sortie au maître via Modbus. Les modules d'E/S forment des périphériques esclaves/serveurs, tandis qu'un périphérique maître typique est un ordinateur hôte exécutant le logiciel

d'application approprié. D'autres appareils peuvent fonctionner à la fois comme clients (maîtres) et serveurs (esclaves). [1]

Les maîtres peuvent s'adresser à des esclaves individuels ou peuvent lancer un message de diffusion à tous les esclaves. Les esclaves renvoient une réponse à toutes les requêtes qui leur sont adressées individuellement, mais ne répondent pas aux requêtes de diffusion. Les esclaves n'initient pas des messages seuls, ils ne répondent qu'aux requêtes du maître. [1]

Une requête maîtresse consistera en une adresse esclave (ou adresse de diffusion), en un code de fonction définissant l'action demandée, toutes les données requises et en un champ de contrôle pour l'erreur. La réponse d'un esclave est constituée de champs confirmant l'action prise, toutes les données à renvoyer et un champ de vérification des erreurs. Notez que la requête et la réponse incluent toutes les deux, une adresse de périphérique, un code de fonction, les données applicables et un champ de vérification des erreurs. Si aucune erreur ne se produit, la réponse de l'esclave contient les données demandées. Si une erreur se produit dans la requête reçue, ou si l'esclave est incapable d'effectuer l'action demandée, l'esclave renverra un message d'exception comme réponse (voir Exceptions Modbus [2]). Le champ de contrôle d'erreur du télégramme de l'esclave permet au maître de confirmer que le contenu du message est valide. Pour le Modbus traditionnel, les messages sont transmis en série et le contrôle de parité est également appliqué à chaque caractère transmis dans sa trame de données.

À ce stade, il est important de faire la distinction que Modbus lui-même est un protocole d'application, car il définit des règles d'organisation et d'interprétation des données, mais reste simplement une structure de messagerie, indépendante du sous-jacente couche physique. Comme il est facile à comprendre, disponible gratuitement et accessible à tous, il est ainsi largement soutenu par de nombreux constructeurs. [1]

Le Modbus est actuellement implémenté sur :

- TCP/IP à travers Ethernet
- Transmission série asynchrone sur une variété de supports

- MODBUS PLUS à grande vitesse de transmission

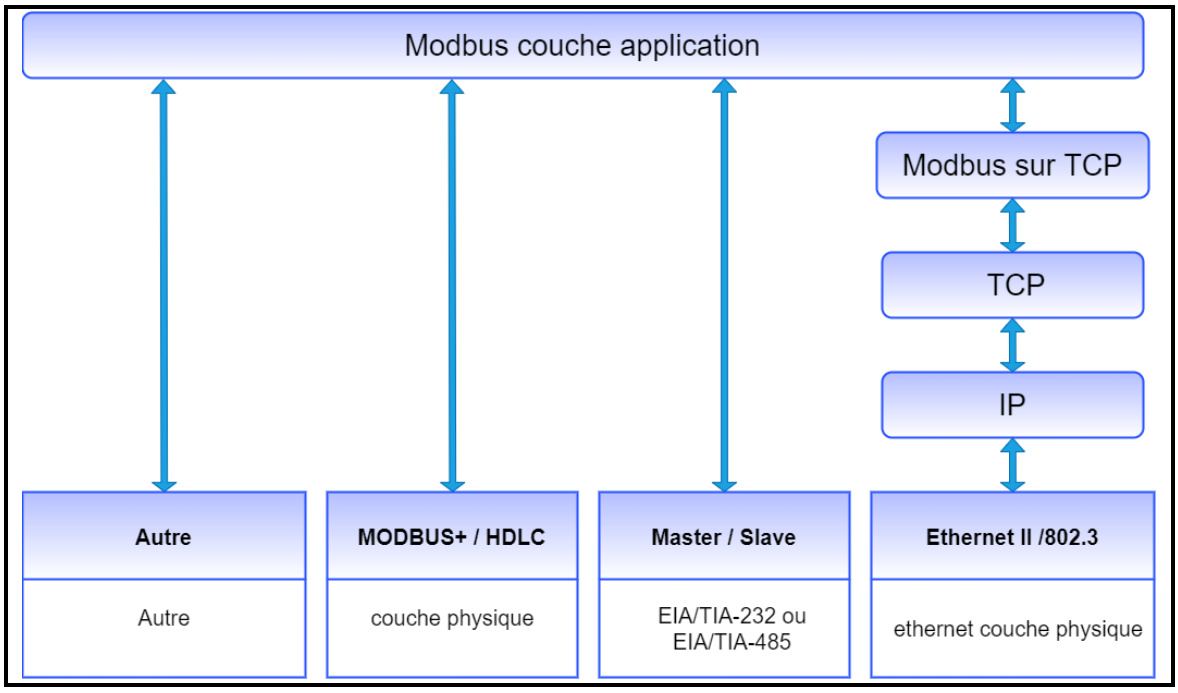

Figure 1.1 : Protocole modbus

1.3 Modbus TCP/IP

Modbus TCP/IP (également Modbus-TCP) est simplement le protocole Modbus RTU avec une interface TCP qui fonctionne sur Ethernet.

La structure de messagerie Modbus est le protocole d'application qui définit les règles d'organisation et d'interprétation des données indépendamment des données support de transmission.

TCP/IP fait référence au protocole de contrôle de transmission et au protocole Internet, qui fournit le support de transmission pour la messagerie Modbus TCP/IP.

En termes simples, TCP/IP permet d'échanger des blocs de données binaires entre les ordinateurs. C'est aussi une norme mondiale qui sert de fondation au World Wide Web. La fonction principale de TCP est de s'assurer que tous les paquets de données sont reçus correctement, tandis que IP s'assure que les messages sont correctement adressés et acheminés. Notez que le TCP/IP combinaison est simplement un protocole de transport, et ne définit pas ce que le data signifie ou comment les données doivent être interprétées (c'est le travail du protocole d'application, Modbus dans ce cas).

En résumé, Modbus TCP/IP utilise TCP/IP et Ethernet pour transporter les données de la structure du message Modbus entre les appareils compatibles. Modbus TCP/IP combine un réseau physique (Ethernet), avec un réseau standard (TCP/IP) et une méthode standard de représentation des données (Modbus comme le protocole d'application). Essentiellement, le message Modbus TCP/IP est simplement une communication Modbus encapsulée dans un Ethernet TCP/IP empaqueté.

En pratique, Modbus TCP embarque une trame de données Modbus standard dans un trame TCP, sans la somme de contrôle Modbus.

1.4 Réseaux modbus

Il existe plusieurs types de modbus dans l'industrie, Modbus TCP/IP, Modbus RS485, Modbus RS232 et Modbus plus (MB+), il est possible de brancher sur tous ces réseaux **figure 1.2** pour mettre une flexibilité sur la communication entre tous les types des dispositifs industriels (PLC, interface HMI, les capteur…)

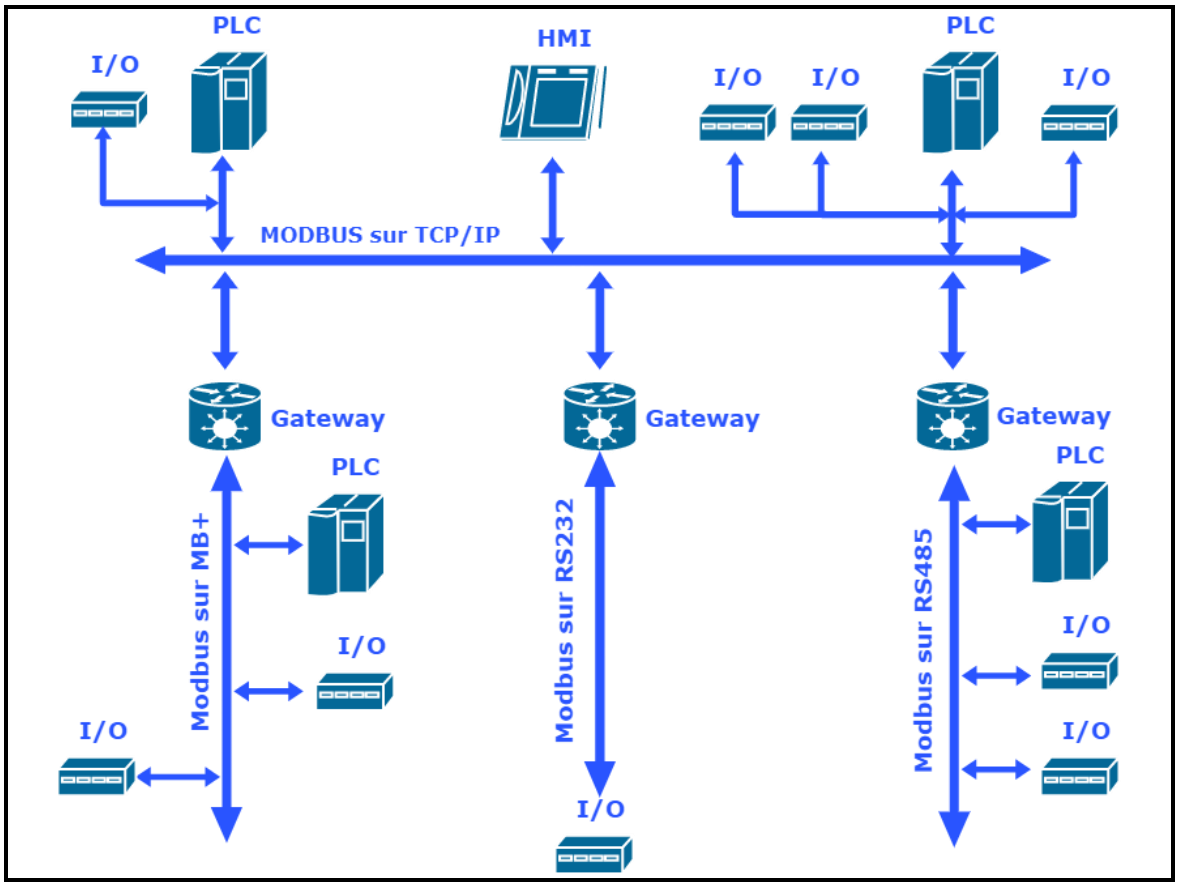

Figure 1.2 : Réseaux modbus

1.5 La communication sur modbus TCP/IP

La communication sur Modbus RS485, Modbus RS232 est basée sur la notion MASTER/SLAVE (maitre/esclave), tous les dispositifs de la partie commande se comportent comme maitre et tous les dispositifs de la partie opérationnelle se comportent comme des esclaves, le maitre envoie une commande (exp : un PLC) pour lire une donnée à partir d'un esclave (exp : un capteur), ce dernier doit répondre par une donnée

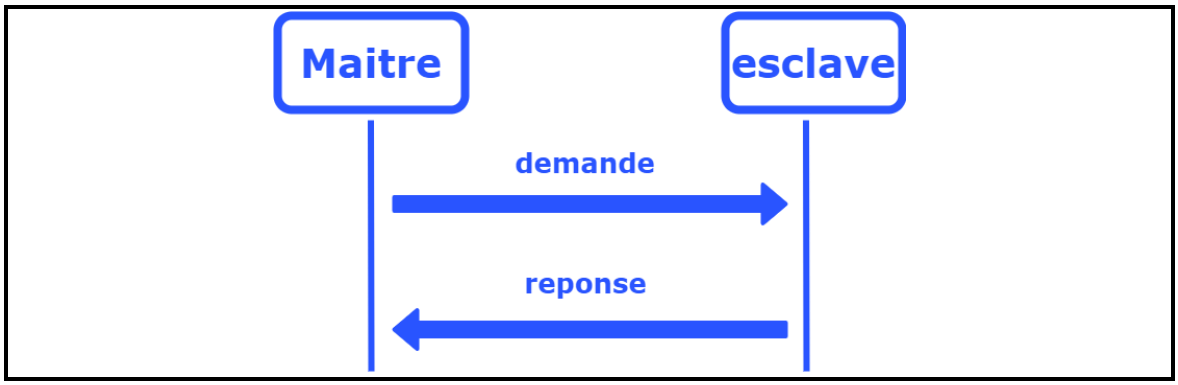

Figure 1.3 : Communication modbus maitre/esclave

Le modbus TCP/IP **figure 1.4** est basé sur la notion CLIENT/SERVER (client /serveur), tous les dispositifs de la partie commande se comportent comme des clients ou client/serveur en même temps et tous les dispositifs de la partie opérationnelle se comportent comme des serveurs

Pour lancer une communication entre le client et le serveur il faut que le client doit être connecté avec le serveur

Pour cela le client envoie une trame SYN que le serveur doit accuser la réception et réponds par l'envoie d'une trame SYN_ACK, ensuite le client accuse la réception et réponds par une trame ACK, ainsi la connexion est établie et par conséquence le client peut communiquer avec le serveur.

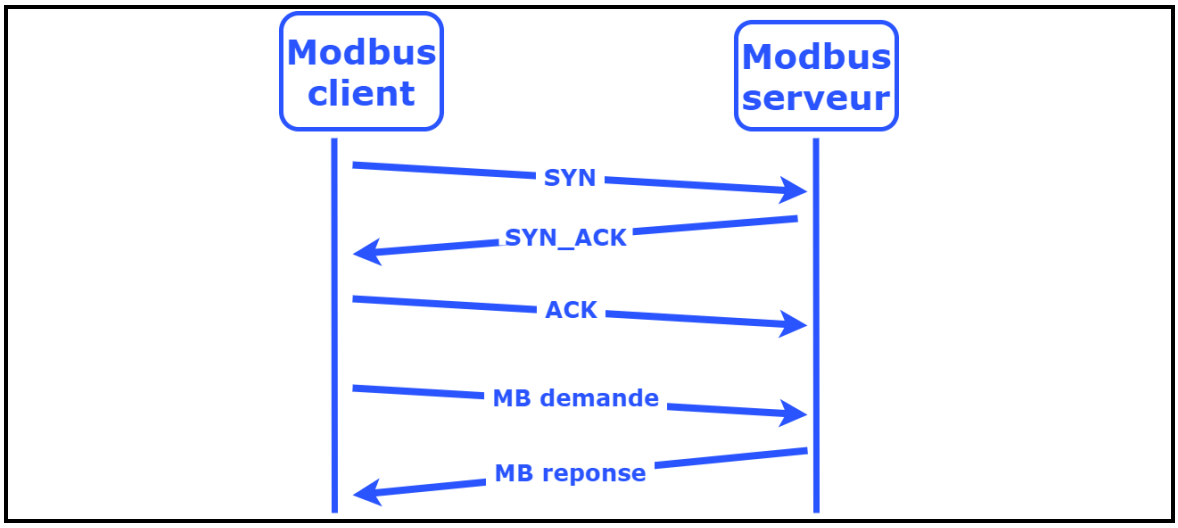

Figure 1.4 : Communication modbus TCP/IP

1.5.1 Modèle de données MODBUS

Le modbus utilise quatre zone de mémoire pour représente quatre type de donnes comme suit sur le **tableau 1.1**

	Type d'objet	**Type de**	**commentaires**
Entrée discrète (Discretes Input)	1 bit	Lecture seulement	Ce type de données peut être fourni par un système d'E/S.
Bobines (Coils)	1 bit	Lecture seulement	Ce type de données peut être altérable par une application programme.
Registres d'entrée (Input Registers)	16 bit	Lecture seulement	Ce type de données peut être fourni par un système d'E/S
Registres de détention (Holding Registers)	16 bit	Lire /écrire	Ce type de données peut être altérable par une application programme.

Tableau 1.1 : Type des donne modbus [2]

Les entrées discrètes (Discretes Input) ce sont des entrées a 1 bit par exemple un capteur de position il a deux état possible 1 et 0

Les bobines (Coils) se sont les sorties a 1 bit par exemple un relier il a deux état possible 1 et 0

Les Registres d'entrée (Input Registers) se sont des entrer a 16 bit

Registres de détention (Holding Registers) se sont des registres a 16 bit

Les distinctions entre entrées et sorties, et entre adressable par bit et adressable par mot éléments de données, n'impliquent aucun comportement de l'application. C'est parfaitement acceptable et très courant, de considérer les quatre zones de mémoire comme se superposant, si c'est le plus naturel interprétation sur la machine cible en question.

Pour chacune des zones de mémoire primaires, le protocole permet une sélection individuelle de 65536 éléments de données, et les opérations de lecture ou d'écriture de ces éléments sont conçues pour s'étendre sur plusieurs éléments de données jusqu'à une limite de taille de données qui dépend du code de fonction de transaction.

Il est évident que toutes les données manipulées via MODBUS (bits, registres) doivent se trouver dans l'appareil mémoire d'application. Mais l'adresse physique en mémoire ne doit pas être confondue avec les données référence. La seule exigence est de lier la référence de données à l'adresse physique. Les numéros de référence logique MODBUS, qui sont utilisés dans les fonctions MODBUS, ne sont pas signés indices entiers commençant à zéro. [2]

1.5.2 Trame modbus TCP/IP

La trame modbus TCP/IP ADU (Application Data Unit) **figure 1.5** est constituée de trois parties, l'entête de la trame pour l'identifier (MBAP), le code de fonction pour identifier l'action demander à faire en cas d'une requête Modbus créer par le client ou en cas pour identifier une réponse Modbus créer par le serveur et la partie (DATA) contenant les données transmises par le client ou le serveur

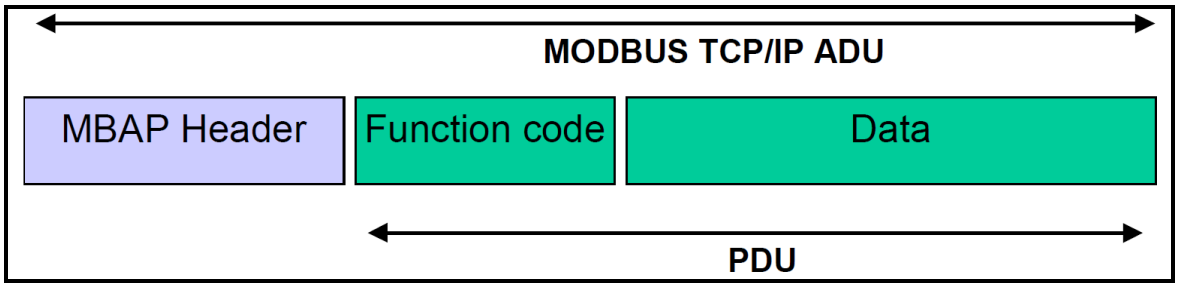

Figure 1.5 : Modbus requête/réponse à travers TCP/IP [2]

- **L'en-tête MBAP**

L'en-tête MBAP contient les champs suivants :

Identifiant de la transaction (2 octet) : pour l'identification d'une transaction Requête/Réponse MODBUS, Initialisé par le client et Recopié par le serveur à partir de la requête reçue

Identificateur de protocole (2 octet) : met à zéro pour le Protocole Modbus, Initialisé par le client et Recopié par le serveur à partir de la requête reçue

Longueur (2 octet) : Nombre d'octets suivants ce champ, Initialisé par le client (requête) et Initialisé par le serveur (Réponse)

Identificateur d'unité (1 octet) : Identification d'un esclave distant connecté sur une ligne série ou sur d'autres bus, Initialisé par le client et Recopié par le serveur à partir de la requête reçue

- **Code des fonctions**

Il existe trois catégories de codes Fonctions MODBUS **tableau 1.2.** Elles sont :

Codes de fonction publique :

- Sont des codes de fonction bien définis,
- Garantie d'être unique,
- Validé par la communauté MODBUS.org,
- Documenté publiquement
- Disposer d'un test de conformité,

- Comprend à la fois des codes de fonction attribués publics définis ainsi que des codes de fonction non attribués réservés pour une utilisation future.

Codes de fonction définis par l'utilisateur :

- Il existe deux plages de codes de fonction définis par l'utilisateur, à savoir 65 à 72 et de 100 à 110 décimal.
- L'utilisateur peut sélectionner et implémenter un code de fonction qui n'est pas pris en charge par la spécification.
- Il n'y a aucune garantie que l'utilisation du code de fonction sélectionné sera unique
- Si l'utilisateur souhaite repositionner la fonctionnalité en tant que code de fonction publique, il doit initier un RFC pour introduire le changement dans la catégorie publique et se voir attribuer un nouveau code de fonction publique.
- L'Organisation MODBUS, se réserve expressément le droit de développer le RFC proposé.

Codes de fonction réservés :

- Codes de fonction actuellement utilisés par certaines entreprises pour les produits hérités et qui ne sont pas disponibles pour un usage public.

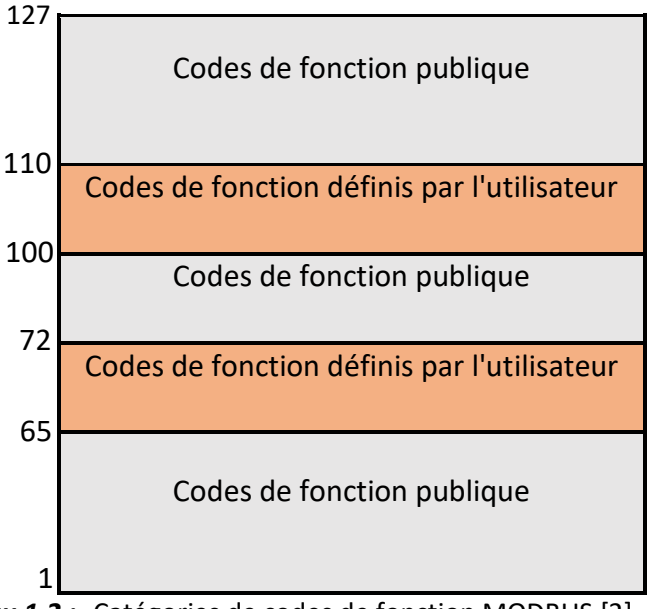

Tableau 1.2 : Catégories de codes de fonction MODBUS [2]

Pour plus de détail sur les codes des fonctions voir [2]

La trame modbus TCP/IP ADU est encapsulée sur TCP a la couche transport, ensuit la trame TCP est encapsulée sur la trame IP. Enfin la trame IP est encapsulée sur la trame Ethernet **figure 1.6**

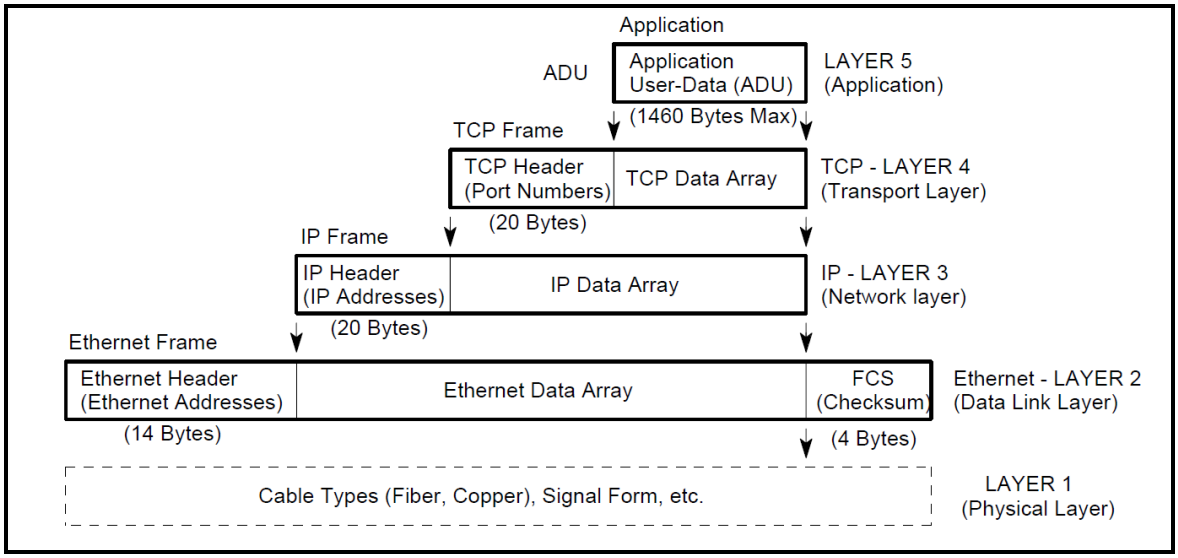

Figure 1.6 : Construction d'un paquet de donnée TCP/IP-ETHERNET [1]

L'unité de données d'application (ADU) MODBUS est construite par le client qui initie une transaction MODBUS.

Le code de la fonction indique au serveur quel type d'action effectuer. L'application MODBUS Le protocole établit le format d'une requête initiée par un client.

Le champ de code de fonction d'une unité de données MODBUS est codé sur un octet. Les codes valides sont dans le plage de 1 ... 255 décimal (la plage 128 - 255 est réservée et utilisée pour les exceptions réponses). Lorsqu'un message est envoyé d'un client à un périphérique serveur, le champ du code de fonction indique au serveur quel type d'action effectuer. Le code de fonction "0" n'est pas valide.

Des codes de sous-fonction sont ajoutés à certains codes de fonction pour définir plusieurs actions. Le champ de données des messages envoyés d'un client aux périphériques serveur contient des informations supplémentaires que le serveur utilise pour effectuer l'action définie par le code de fonction. Cela peut inclure des éléments comme les adresses discrètes et de registre, la quantité d'articles à traiter et le nombre de octets de données réels dans le champ.

Le champ de données peut être inexistant (de longueur nulle) dans certains types de requêtes, dans ce cas le serveur ne nécessite aucune information supplémentaire. Le code de fonction à lui seul spécifie l'action.

Si aucune erreur ne se produit liée à la fonction MODBUS demandée dans un MODBUS correctement reçu ADU le champ de données d'une réponse d'un serveur à un client contient les données demandées. Si une erreur liée à la fonction MODBUS demandée se produit, le champ contient un code d'exception que l'application serveur peut utiliser pour déterminer la prochaine action à entreprendre.

Par exemple un client peut lire les états ON/OFF d'un groupe de sorties ou d'entrées TOR ou il peut lire/écrire le contenu des données d'un groupe de registres.

Lorsque le serveur répond au client, il utilise le champ du code fonction pour indiquer soit un réponse normale (sans erreur) ou qu'une erreur s'est produite (appelée exception réponse). Pour une réponse normale, le serveur renvoie simplement à la requête l'original code de fonction.

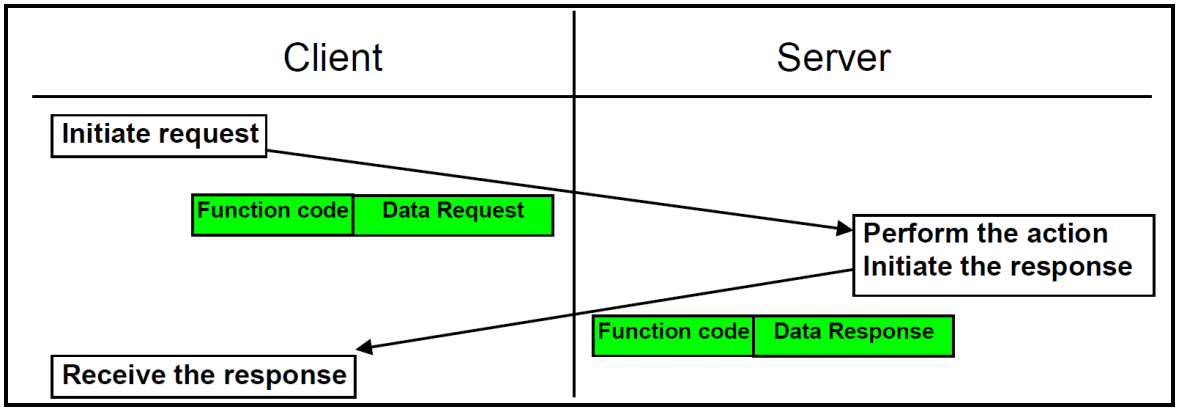

Figure 1.7 : Transaction MODBUS (sans erreur) [2]

Pour une réponse d'exception, le serveur renvoie un code équivalent à la fonction d'origine code de la PDU de demande avec son bit de poids fort défini sur logique 1.

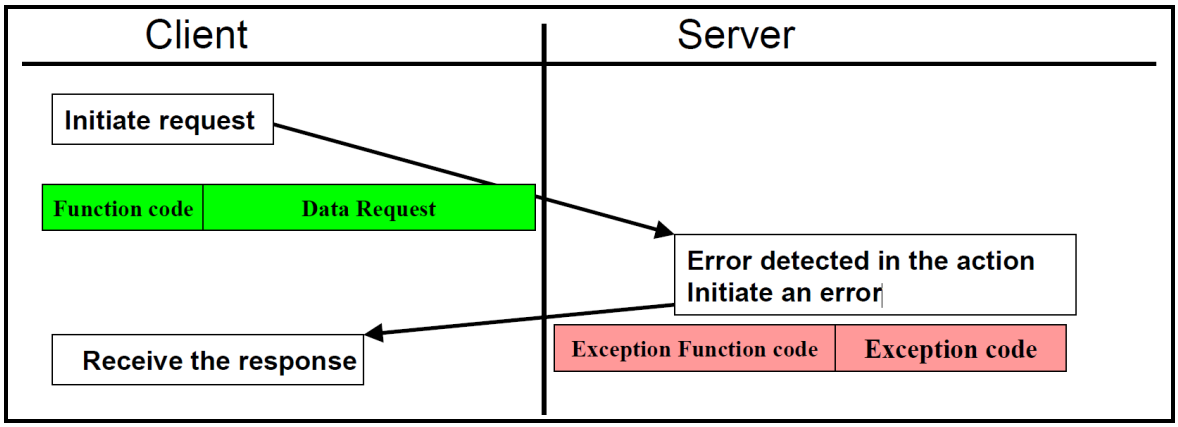

Figure 1.8 : Transaction MODBUS (réponse d'exception) [2]

1.6 Conclusion

Nous avons vu dans ce chapitre la constitution d'un protocole de communication modbus TCP/IP, notre choix s'est porté sur la conception d'un IO device qui sera apte à communiquer avec un PLC à l'aide de ce protocole.

Chapitre 2 Implémentation

2.1 Introduction

Dans la majorité des installations industrielles, on trouve un nombre important de capteurs et d'actionneurs afin de surveiller et de contrôler les grandeurs physiques des procédés, le déploiement de ces capteurs actionneurs requièrent des quantités de cartes d'acquisition et des longueurs de câbles plus un réseau analogique (4-20mA ou autre) sensible au interférences.

Afin de minimiser les coûts et complexité de ces installations nous avons conçu une solution hardware et software concrétisée par la réalisation d'un I/O Device communiquant via le protocole industriel MODBUS TCP/IP et un algorithme de gestion de la communication pour des topologies étoiles ou anaux.

Dans ce chapitre nous allons présenter comment implémenter le modbus avec notre pure software et hardware.

2.2 HARDWARE

La **figure 2.1** montre la vue globale du HARDWARE de notre projet.

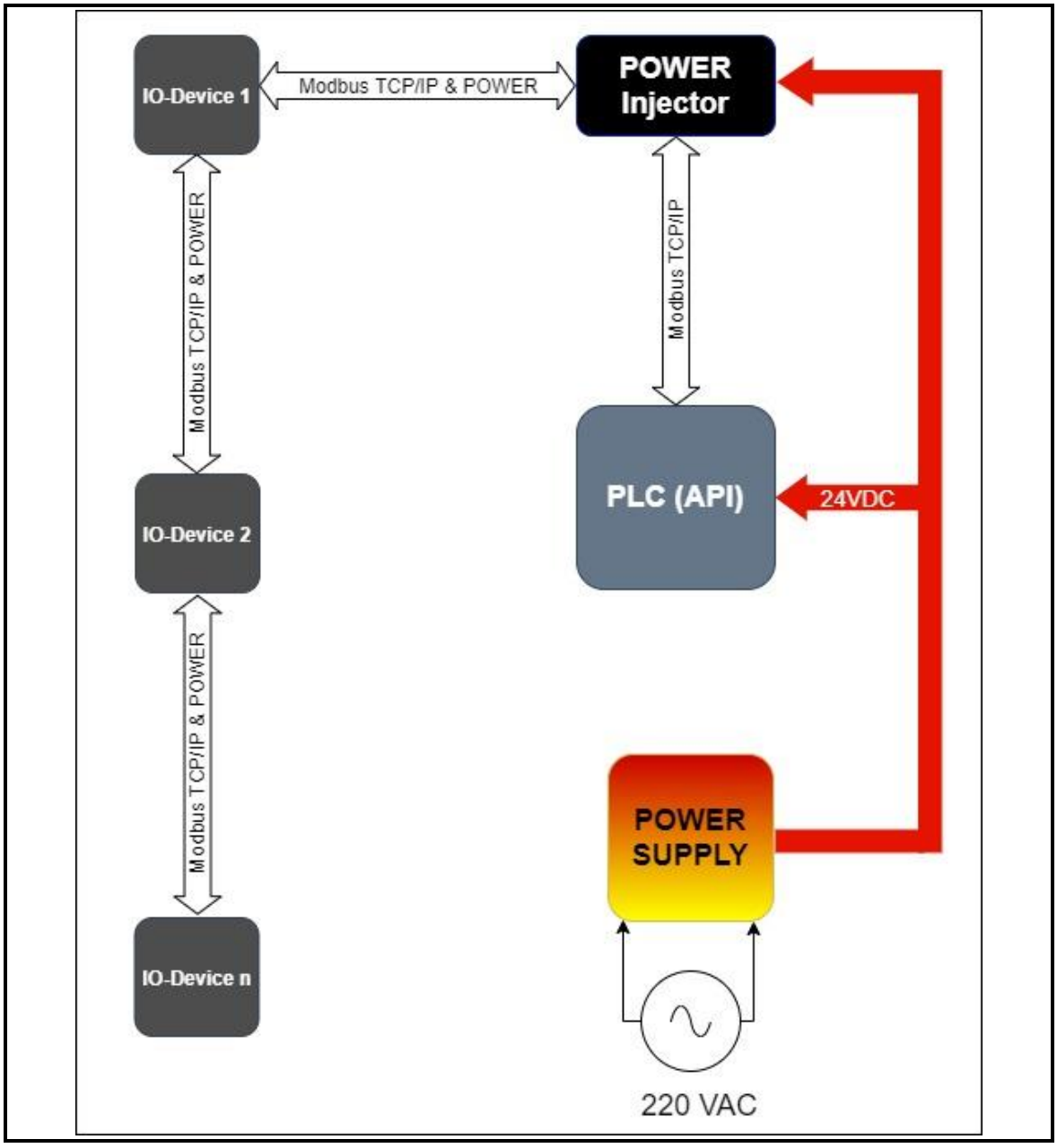

Figure 2.1 : Schéma synoptique globale

Remarque : le nombre des IO-Devices est limité par deux facteurs

- L'énergie absorbée par le io-device.
- Le courant maximum délivré par l'injecteur de puissance (POWER Injector).

Nous avons fabriqué notre propre équipement en suivant le cahier des charges.

2.2.1 Cahier des charges

L’implémentation du protocole industriel modbus nécessite un cahier des charges bien précis à suivre.

Pour la partie HARDWARE, notre but est de construire un IO-Device qui possède les caractéristiques suivantes :

- Prise en charge de la technologie PoE.
- Prise en charge de deux connecteurs RJ45 pour pouvoir se connecter en anneau.
- Deux émetteur-récepteur d’Ethernet (dual-PHY on chip).
- Une tension de marche 24V qui répond aux besoins de l'industrie.
- Un port i2c pour avoir connecté un capteur externe.
- Un microcontrôleur avec une RAM d'au moins 128 Ko, qui prend en charge la connectivité Ethernet.
- Une carte PCB de petite taille pour réduire les coûts avec un design qui respecte la compatibilité électromagnétique.
- Un étage Buck 24V/5V ou 24V/3.3V pour avoir faire fonctionner tous les composants de l'équipement.
- Deux Crystal (oscillateurs) ou plus, un de 50Mhz branché avec le PHY pour le MODE RMII et les autres selon le datasheet du MCU pour assurer le bon fonctionnement.

Avec une réalisation d'injecteur de puissance (POWER Injector) adaptée à notre IO-Device.

La **figure 2.2** illustre le schéma synoptique interne du IO-Device et de sa connexion avec le réseau.

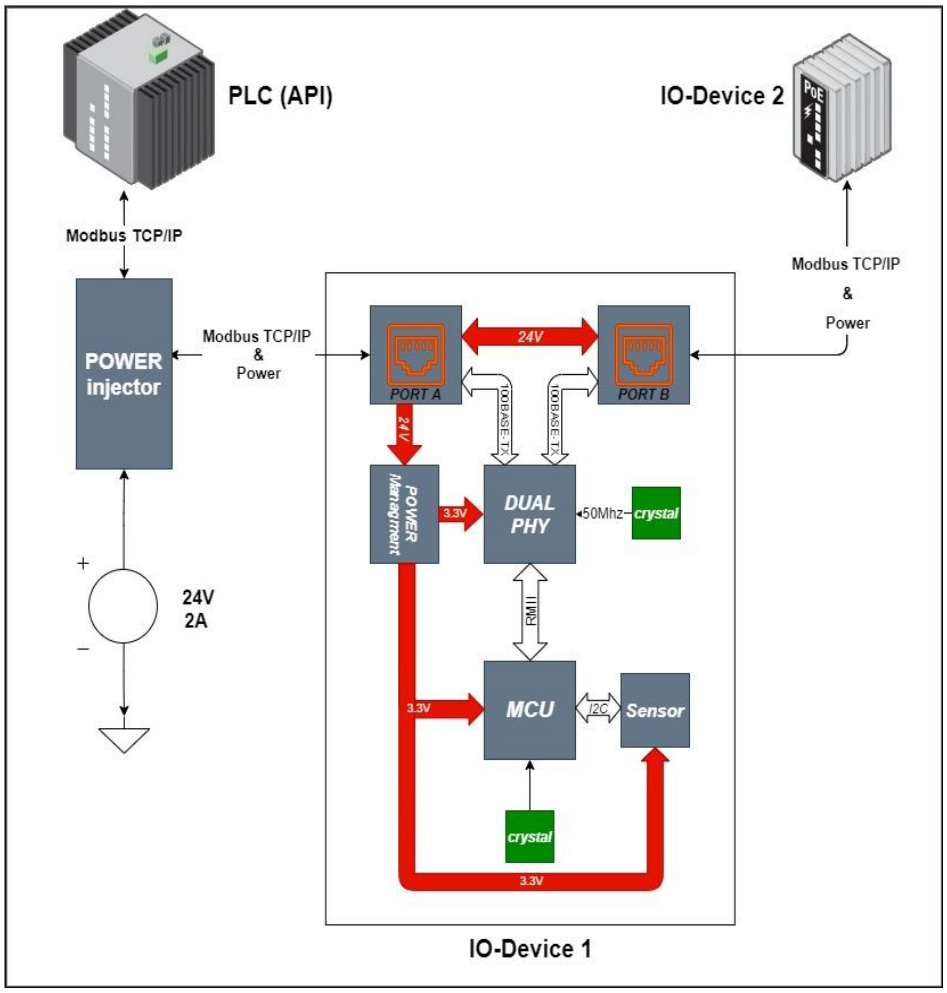

Figure 2.2 : Structure interne du IO-Device et sa connexion au réseau

2.2.2 Power over Ethernet (PoE)

a) Définition

L'alimentation électrique par câble Ethernet (*Power over Ethernet* ou PoE en anglais), permet de faire passer une alimentation électrique (initialement un courant continu d'une puissance maximale de 15,4 watts avec une tension d'environ 48 V), en plus des données à 100 Mbit/s ou 1 Gbit/s. Cette technologie est définie par la norme IEEE 802.3af, appartenant au standard IEEE 802.3 (Ethernet) ratifiée le 11 juin 2003 et publiée le 11 juillet 2003 [3].

Cette technologie alloue deux paires (ou plus) sur les quatre paires que contient un câble UTP ou STP afin d'alimenter certains appareils d'un réseau Ethernet tels que des téléphones IP, des disques durs réseaux, des caméras IP ou des points d'accès Wi-

Fi. La norme IEEE 802.3at, également appelée PoE+, ainsi que la norme IEEE 802.3bt appelée 4PPoe, en étendent les caractéristiques techniques [3].

b) PoE standards

Il existe quatre normes du PoE, le tableau suivant explique la différence entre ces normes

PoE[Power Over Ethernet] STANDARDS				
Standards IEEE	802.3at (type1) "PoE"	802.3at(type2) "PoE+"	802.3bt(type3) "4PPoE"	802.3bt(type4) "4PPoE"
puissance disponible à PD	12.95 W	25.50 W	51 W	71 W
puissance maximale délivrée par PSE	15.40 W	30 W	60 W	100 W
courant maximal	350 mA	600 mA	600 mA par paire	960 mA par paire

Tableau 2.1 : PoE standards

Ce qui nous intéresse dans ce tableau est la quatrième colonne en vert qui contient la norme IEEE 802.3bt qui s'appelle aussi 4PPoE, on a choisi cette norme car nous avons besoin d'un courant électrique maximal pour connecter autant d'IO-Devices que possible dans les réseaux.

c) Le standard IEEE 802.3bt type 4

Le standard IEEE 802.3bt, également dénommé 4PPoE (*4-Pair Power over Ethernet*), ou encore PoE++ par extension d'appellation PoE+ du standard IEEE 802.3at, introduit deux niveaux supplémentaires de puissance maximale délivrée par le switch, type 3 avec 55 W (avec une intensité maximale par paire de 600 mA) et type 4 avec 90W (avec une intensité maximale par paire de 960 mA) répartie sur les 4 paires du câble Ethernet.

Compte-tenu des pertes dans le câble, la puissance réellement disponible sur l'équipement est de 71 W sur l'équipement alimenté en type 4.

La norme IEEE 802.3bt-2018 (Draft Standard for Ethernet Amendement : Physical Layer and Management Parameters for DTE Power via MDI over 4-Pair) est approuvée le 27 septembre 2018.

Par rapport aux normes PoE et PoE+ qui utilisent deux paires parmi quatre, à puissance délivrée égale, le 4PPoE améliore l'efficacité énergétique globale car les pertes sont diminuées de moitié. Pour une intensité constante, la puissance effectivement délivrée sur quatre paires est plus élevée que celle disponible sur deux paires [3].

La **figure 2.3** montre la transmission d'énergie et de données dans un câble cat5 par le standard 4ppoe.

Pin	Signal Name	Description	cable wire color	Name	Pin
1	TX+_D1 & VCC	Transmit Data+	White with green strip	TX+_D1	1
2	TX-_D1 & VCC	Transmit Data-	Green with white stripe or solid green	TX-_D1	2
3	RX+_D2 & GND	Receive Data+	White with orange stripe	RX+_D2	3
4	VCC		Blue with white stripe or solid blue		4
5	VCC		White with blue stripe		5
6	RX-_D2 & GND	Receive Data-	Orange with white stripe or solid orange	RX-_D2	6
7	GND		White with brown strip		7
8	GND		Brown with white stripe or solid brown		8

Figure 2.3 : Transmission (data & power) dans le standard 4ppoe

2.2.3 Power injector

C'est un appareil qui combine l'énergie et les données, puis l'envoie par le port Ethernet à un équipement qui prend en charge la technologie PoE.

Pour simplifier les choses, nous avons développé un Schéma synoptique **figure 2.4** qui explique comment cela fonctionne.

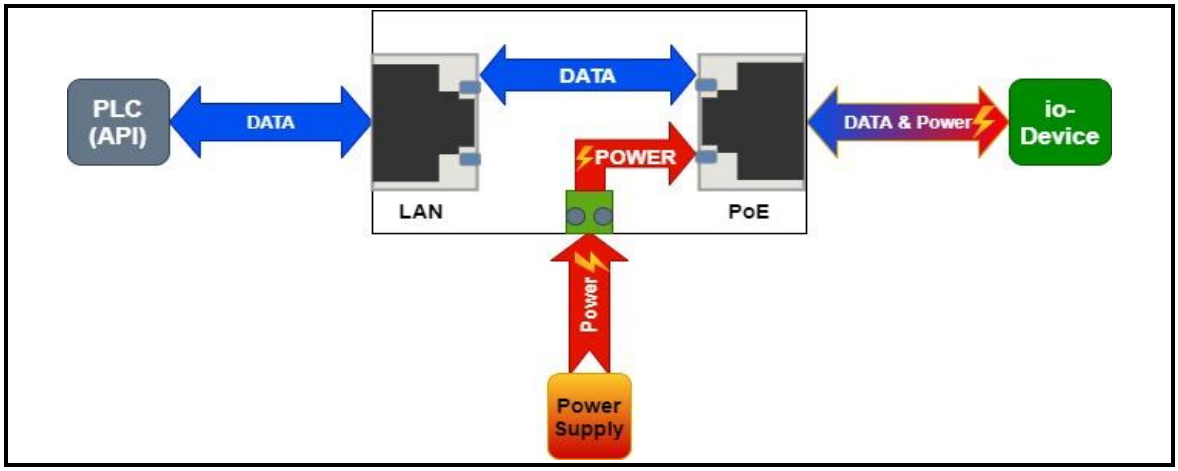

Figure 2.4 : Schéma synoptique de l'injecteur de puissance.

a) Choix des composants

Les composants que nous avons besoin pour avoir réalisé l'injecteur de puissance (Power Injector) sont :

- Un Connecteur rj45 simple.
- Un Connecteur rj45 qui supporte la technologie 4PPoE.
- Un connecteur pour brancher l'alimentation.

Après des recherches approfondies, nous avons choisi les composants suivants :

	Connecteur rj45 avec 4PPoE	Connecteur rj45 simple	Connecteur d'alimentation
Image			
MPN	0826-1X1T-KL-F	HR911105A	TB004-508-02BE

Figure 2.5 : Composants utilisés dans l'injecteur de puissance [4]

b) Schéma électrique

La **figure 2.6** nous donne le schéma électrique interne de l'injecteur de puissance.

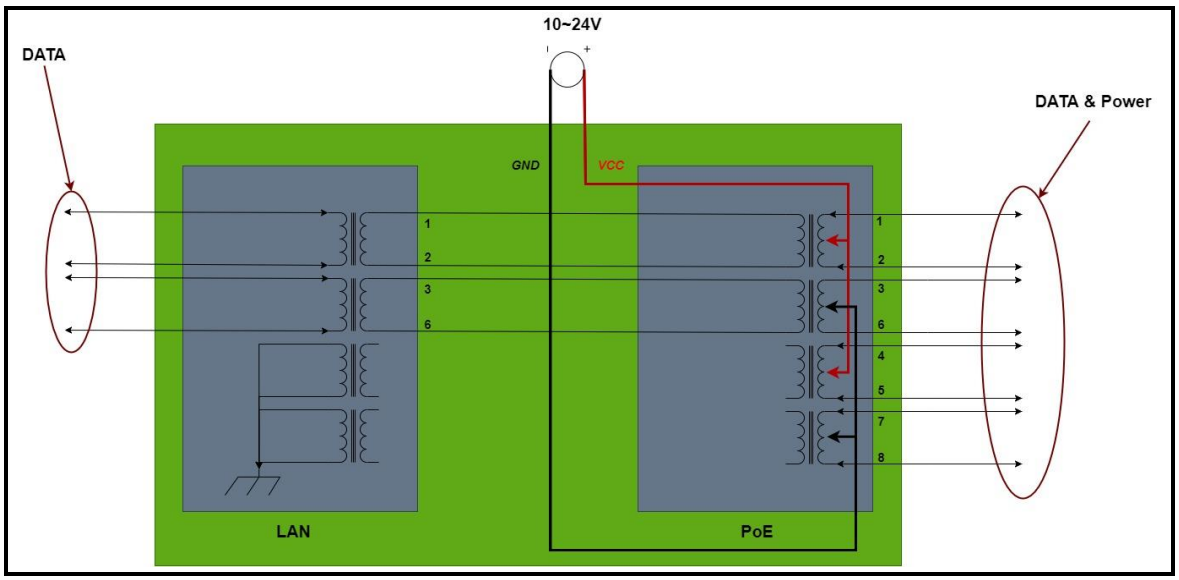

Figure 2.6 : Schéma électrique de l'injecteur de puissance.

c) Circuit utilisé sur le logiciel

Pour ceux qui n'ont jamais utilisé de tels programmes auparavant, nous expliquerons d'abord les méthodes de connexion des fils entre les composants.

La **figure 2.7** montre deux manières différentes de schématiser les connexions d'un circuit électrique.

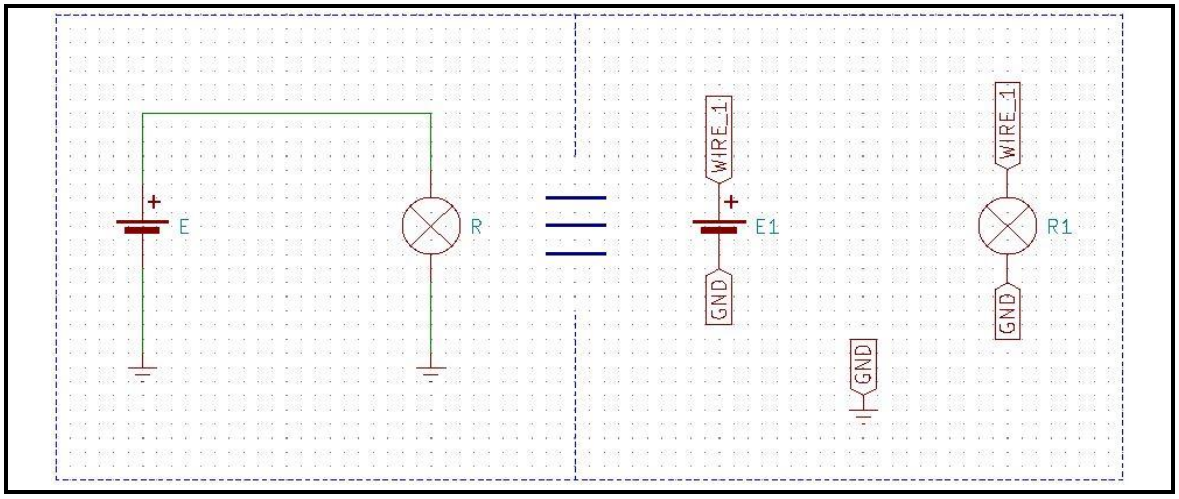

Figure 2.7 : Différents schémas de connexion des composants.

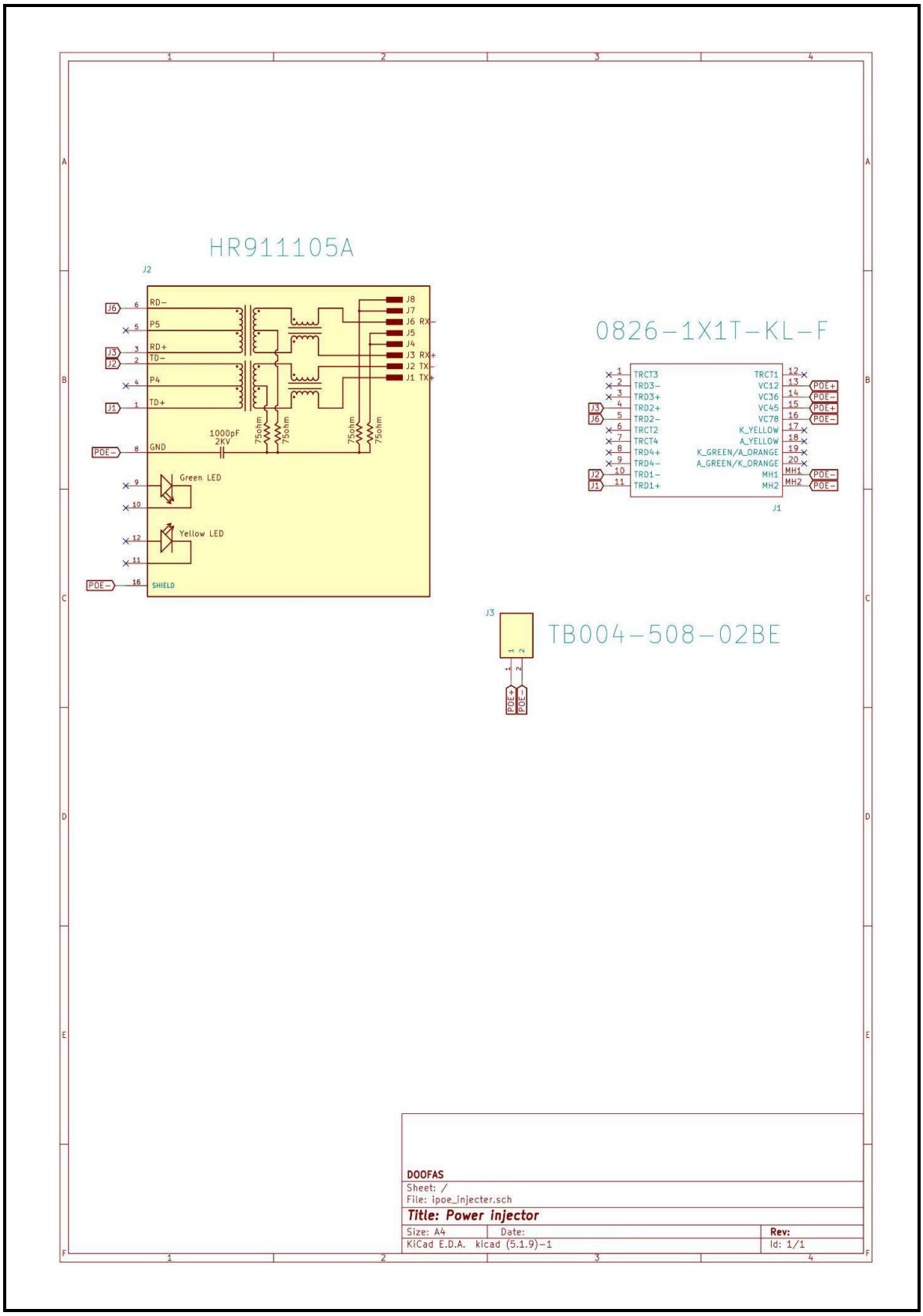

Figure 2.8 : Circuit électrique utilisé pour le power injector

2.2.4 IO-Device

a) Définition

Également appelé périphérique IO, ce périphérique d'entrée/sortie est tout matériel utilisé par un opérateur humain ou d'autres systèmes pour communiquer avec un ordinateur. Comme son nom l'indique, les périphériques d'entrée/sortie sont capables d'envoyer des données (sortie) à un ordinateur et de recevoir des données d'un ordinateur (entrée).

b) Choix et caractéristique des composant principaux

Le choix des composants dépend principalement du cahier des charges et du schéma synoptique de la **figure 2.2**.

Nous avons choisi les composants suivants à cause de leurs compatibilités, leurs disponibilités, leurs prix et leurs tailles.

- **MCU : STM32F207IGH6**

Dans le logiciel de STMicroelectronics (STM32CubeMX), nous pouvons choisir le microcontrôleur en fonction des caractéristiques qu'il contient, la **figure 2.9** montre comment nous avons choisi le MCU.

Nous avons tout d'abord choisi le périphérique Ethernet, une liste des microcontrôleurs apparaît dans l'ordre croissant en termes de spécifications.

Puis nous avons choisi la RAM requise dans le cahier des charges.

Le choix du boîtier (package) est alors défini, il doit être le plus petit.

Enfin, le bon microcontrôleur est listé.

Nous avons finalement changé le MCU stm32f207ich6 par stm32f207igh6 en raison de sa disponibilité et de la crise actuelle de fabrication des semi-conducteurs, la différence entre ces deux MCU's est la mémoire flash, le stm32f207ich6 a une mémoire de 256 KB alors que pour le stm32f207igh6, elle est à1024 KB.

Figure 2.9 : Étapes de sélection du MCU.

Le stm32f207igh6 est un microcontrôleur 32 bits basé sur l'architecture ARM, le **tableau 2.2** montre ses caractéristiques.

Attribut de produit	Valeur d'attribut
Fabricant:	STMicroelectronics
Catégorie du produit:	Microcontrôleurs ARM - MCU
Série:	STM32F207IG
Style de montage:	SMD/SMT
Package/Boîte:	BGA-176

Coeur:	ARM Cortex M3
Taille de la mémoire du programme:	1 MB
Largeur du bus de données:	32 bit
Résolution CAN:	12 bit
Fréquence de l'horloge max.:	120 MHz
Nombre d'E/S:	140 I/O
Taille de la RAM de données:	128 kB
Tension d'alimentation de fonctionnement:	1.8 V to 3.6 V
Température de fonctionnement min.:	- 40 C
Température de fonctionnement max.:	+ 85 C
Type de la mémoire programme:	Flash
Type de Ram de données:	SRAM
Taille de la ROM de données:	512 B

Tableau 2.2 : Caractéristiques du stm32f207 [5].

- **Dual-phy on chip: DP83849IFVS/NOPB**

- **Description**

Le nombre d'applications nécessitant une connectivité Ethernet continue d'augmenter. Parallèlement à cette augmentation, la demande du marché est un changement dans les exigences d'application. Là où l'Ethernet monocanal était

Suffisant, de nombreuses applications telles que les stations de base commandent sans fil et les réseaux industriels nécessitent désormais une fonctionnalité DUAL Port pour la redondance ou la gestion du système

Le DP83849IF est un appareil très fiable et riche en fonctionnalités parfaitement adapté aux applications industrielles permettant Ethernet dans l'usine. Le DP83849IF dispose de deux ports 10/100 entièrement indépendants pour le multi-port applications. La capacité de commutation de port unique de Texas Instruments permet également de configurer les deux ports pour fournir une extension de plage entièrement intégrée, une conversion de média, un basculement matériel et un port surveillance.

Le **tableau 2.3** montre les caractéristiques du DP83849IFVS/NOPB.

Attribut de produit	Valeur d'attribut
Fabricant:	Texas Instruments
Catégorie du produit:	CI Ethernet
Style de montage:	SMD/SMT
Package/Boîte:	TQFP-80
Produit:	Ethernet Transceivers
Standard:	0BASE-T, 100BASE-FX, 100BASE-TX
Nombre d'émetteurs-récepteurs:	2 Transceiver
Débit de données:	10 Mb/s, 100 Mb/s
Type d'interface:	MII, RMII, SNI
Tension d'alimentation de fonctionnement:	3.3 V
Tension d’alimentation - Max.:	3.6 V
Tension d’alimentation - Min.:	3 V
Température de fonctionnement min.:	- 40 C

Température de fonctionnement max.:	+ 85 C
Courant d'alimentation maximal:	180 mA
Pd - Dissipation d'énergie :	594 mW

Tableau 2.3 : Caractéristiques du DP83849IFVS/NOPB [6].

- **Circuit Buck: MP2451DT-LF-Z**

Le MP2451DT-LF-Z est un régulateur de tension de commutation Le **tableau 2.4** montre les caractéristiques du circuit Buck MP2451DT-LF-Z.

Attribut de produit	Valeur d'attribut
Fabricant:	Monolithic Power Systems (MPS)
Catégorie du produit:	Régulateurs de tension de commutation
Style de montage:	SMD/SMT
Package/Boîte:	SOT-23-6
Topologie:	Buck
Tension de sortie:	+0.8V to 0.8Vin
Courant de sortie:	600 mA
Tension d'entrée max.:	36 V
Fréquence de commutation:	2 MHz
Tension d'entrée min:	3.3 V

Tableau 2.4 : Caractéristiques du circuit Buck MP2451DT-LF-Z [7].

- **Connecteur RJ45 : 0826-1X1T-KL-F**

Le **tableau 2.5** illustre les caractéristiques du connecteur 0826-1X1T-KL-F

Attribut de produit	Valeur d'attribut
Fabricant:	Bel
Plage de températures de fonctionnement:	- 40 C to + 100 C
courant de ligne cc équilibré:	1A Max
technologie prise en charge:	4PPoE / 100W

Tableau 2.5 : Caractéristiques du connecteur 0826-1X1T-KL-F [8].

c) Configuration hardware

La configuration hardware est liée à la fiche technique, nous montrerons ici comment configurer chaque composant à partir de sa fiche technique.

- **Le microcontrôleur STM32F207IGH6**

- **Circuit de l'alimentation :**

La **figure 2.10** représente le circuit de l'alimentation du stm32f207igh6.

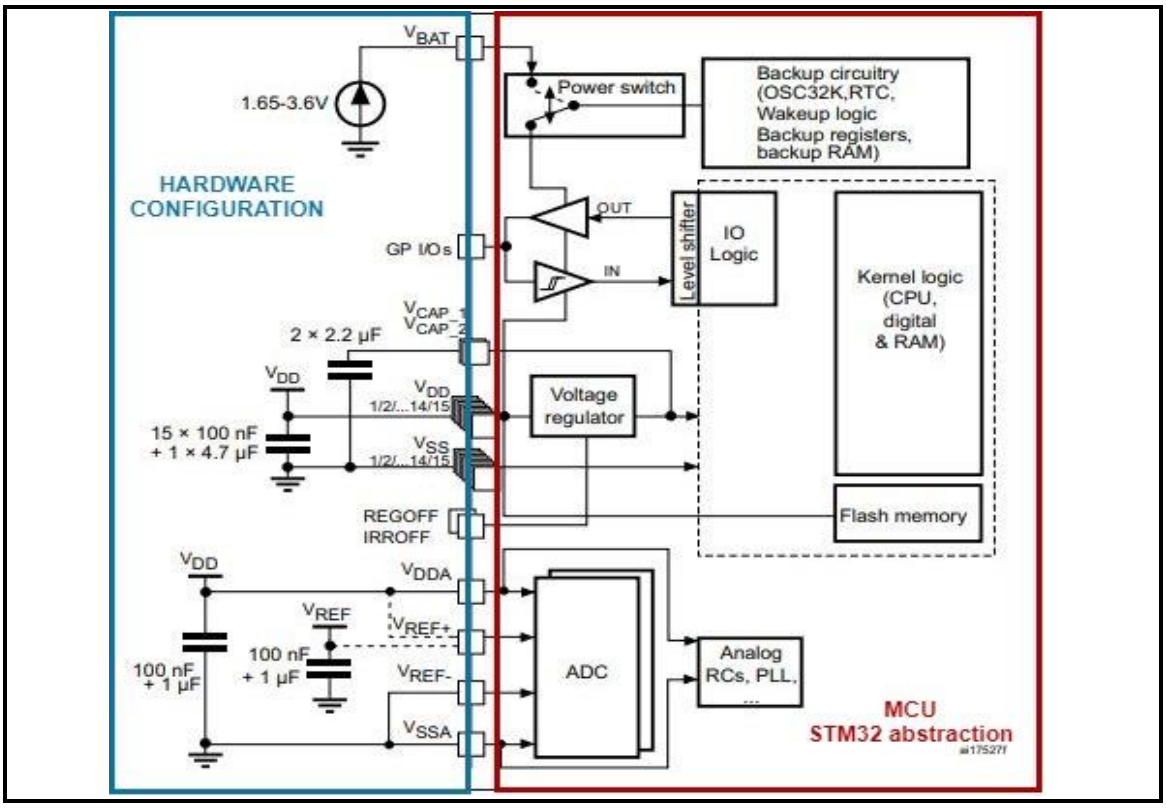

Figure 2.10 : Schéma d'alimentation [9].

-Le condensateur céramique de 4,7 µF doit être connecté à l'une des broches VDD

-Chaque paire d'alimentation (VDD/VSS, VDDA/VSSA...) doit être découplée avec condensateurs céramique filtrante comme indiqué ci-dessous. Ces condensateurs doivent être placés le plus près possible de, ou en-dessous, des broches appropriées sur la face inférieure du PCB, pour assurer un bon fonctionnement de l'appareil. Ce n'est pas recommandé de retirer les condensateurs de filtrage pour réduire la taille ou le coût du PCB. Cela pourrait provoquer un mauvais fonctionnement de l'appareil [9].

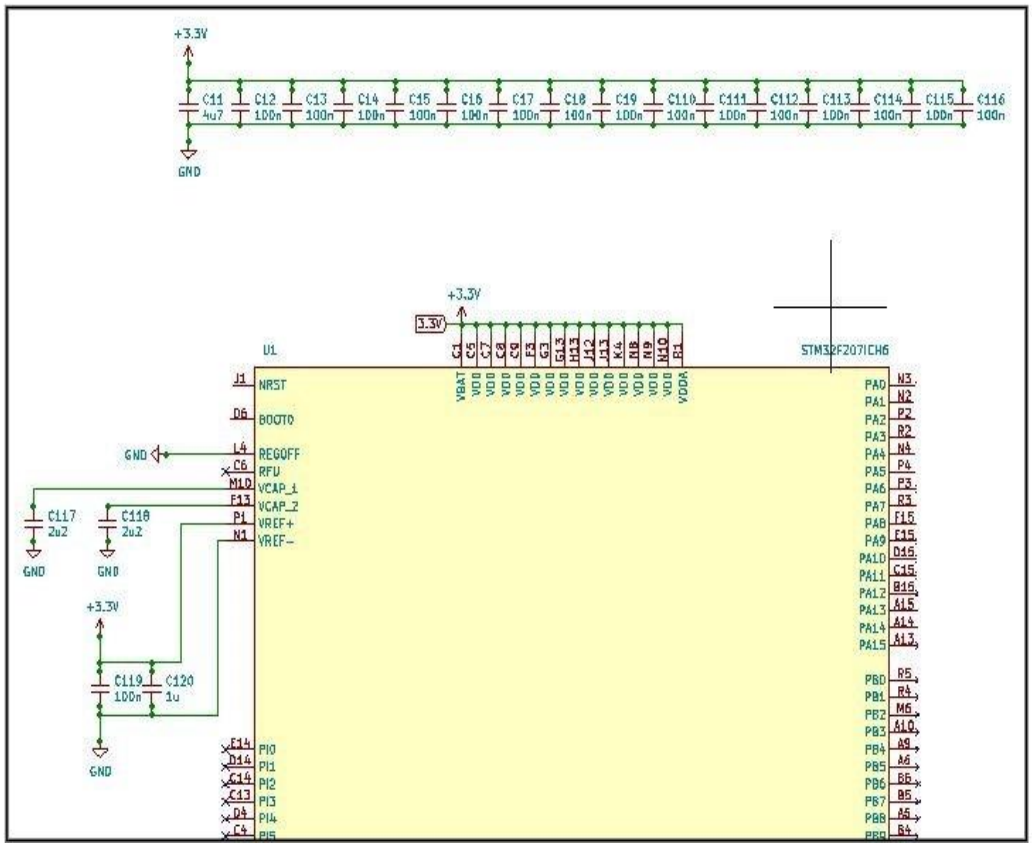

Figure 2.11 : Schéma d'alimentation sur le logiciel

- **Configuration de l'horloge (Crystal résonateur)**

La configuration de l'horloge de référence se fait à travers deux circuits résonateurs, le premier (HSE) garantie la référence de 16 MHz pour les timers internes, le deuxième (LSE) pour fournir la référence 32,768KHz qui permet de donner la date et l'heure.

- De la fiche technique du STM l'horloge externe haute vitesse (HSE) peut être fournie avec un oscillateur cristal/céramique de 4 à 26 MHz [9].

- Après avoir lu la fiche technique du STM, nous avons choisi les deux oscillateurs HSE (X322516MLB4SI) et LSE (Q13FC1350000400).

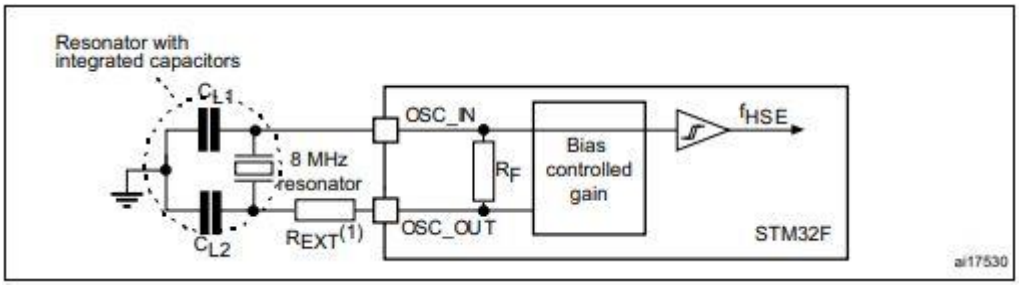

Figure 2.12 : Application typique avec un HSE cristal de 8 MHz [9].

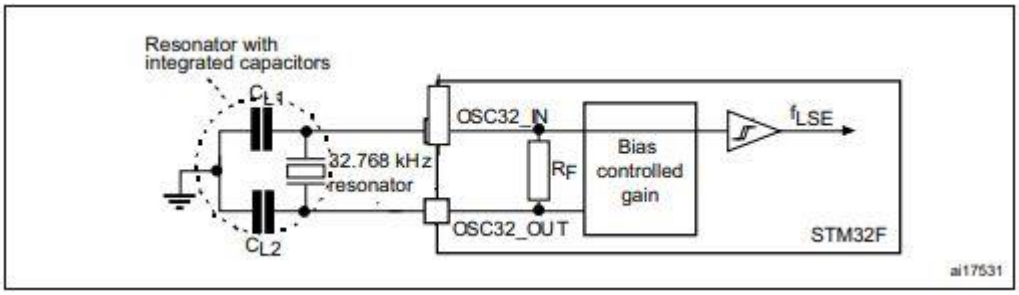

Figure 2.13 : Application typique avec un LSE cristal de 32,768 kHz [9].

Pour CL1 et CL2, il est recommandé d'utiliser des condensateurs céramiques externes de haute qualité dans la Plage de 5 pF à 25 pF (typ.), conçue pour les applications à haute fréquence et sélectionnée pour correspondre les exigences du cristal ou du résonateur (voir **figure 2.12**). CL1 et CL2 sont généralement de même taille. Le fabricant de cristal spécifie généralement une capacité de charge CL qui est la combinaison en série de CL1 et CL2. La capacité des broches du PCB et du MCU Cs doit être incluse (3 pF peut être utilisé comme une estimation approximative de la capacité combinée de la broche et de la carte) lors du dimensionnement CL1 et CL2 [9].

L'équation suivante donne l'expression de CL :

$$C_L = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_s \qquad [10]$$

La capacité de charge du l'oscillateur HSE (X322516MLB4SI) que nous avons utilisé est de 9pF

Donc CL1=CL2= 12pF.

La **figure 2.14** et la **figure 2.15** montrent la configuration des cristaux HSE et LSE.

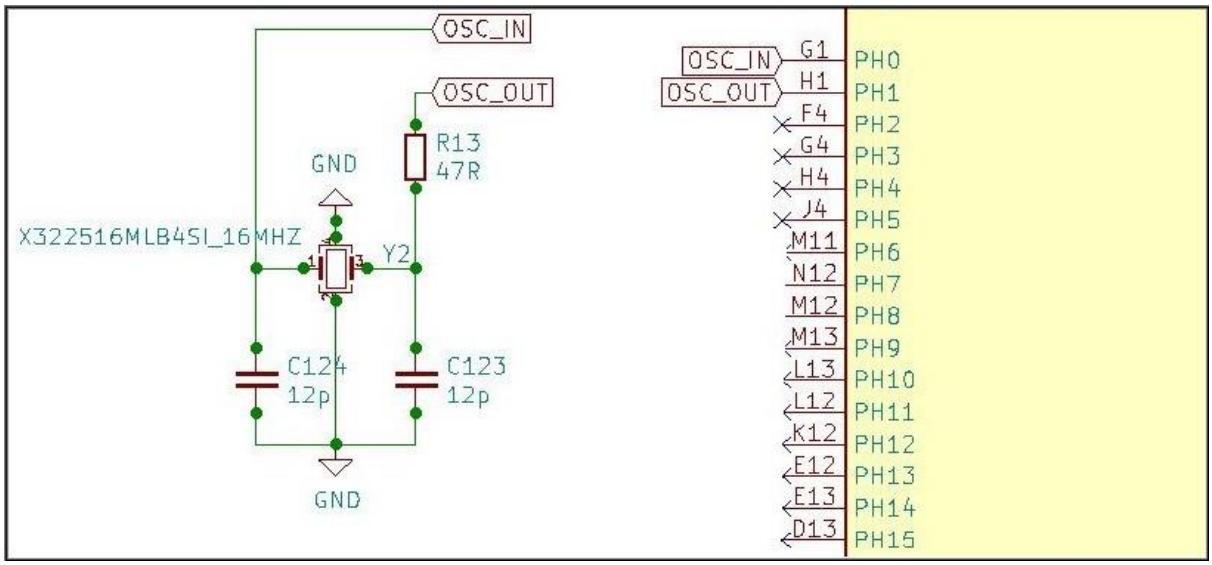

Figure 2.14 : Configuration du cristal HSE

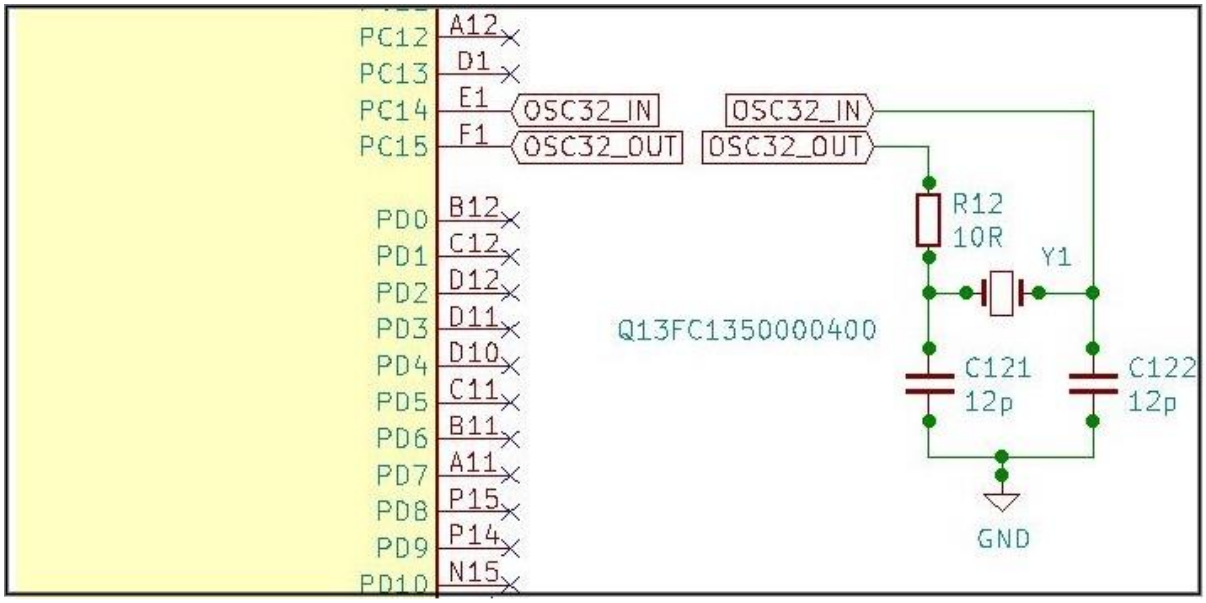

Figure 2.15 : Configuration du cristal LSE

- **Boot configuration**

Stm32f207igh6 a trois modes de démarrage :

En raison de sa mémoire fixe, la zone de code commence à partir de l'adresse 0x0000 0000 (accessible via les bus ICode/DCode) tandis que la zone de données (SRAM) commence à partir de l'adresse 0x2000 0000 (accessible via le bus système). Le

processeur Cortex®-M3 récupère toujours le vecteur de réinitialisation sur le bus ICode, ce qui implique d'avoir l'espace de démarrage disponible uniquement dans la zone de code (généralement, mémoire Flash). Les Microcontrôleurs STM32F20x et STM32F21x implémentent un mécanisme spécial pour pouvoir démarrer à partir d'autres mémoires (comme la mémoire interne SRAM).

Dans le STM32F20x et le STM32F21x, trois modes de démarrage différents peuvent être sélectionnés via les Broches BOOT[1:0] comme indiqué dans le **tableau 2.6** [11].

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Tableau 2.6 : Boot modes [11].

Dans notre système nous avons utilisé seulement la pin boot0 pour le boot configuration donc on a deux modes de démarrage, la **figure 2.16** montre le boot configuration dans le logiciel

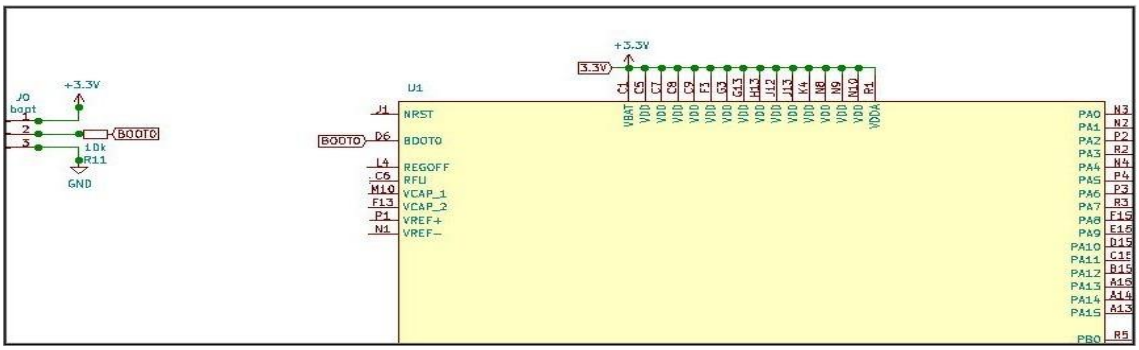

Figure 2.16 : Boot configuration.

- Démarrage à partir de la mémoire système : exécution des instructions à partir de l'adresse mémoire qui pointe sur System-Memory, Ce mode est activé en mettant la pin Boot0 à l'état haut.

- Démarrage à partir de la mémoire flash : exécution des instructions à partir de l'adresse mémoire qui pointe sur le début du programme que contient la mémoire. Ce mode est activé en mettant la pin Boot0 à la masse.

- **Serial wire debug (SWD)**

Le transfert du code et le débogage se fait sur cinq pins, deux pins de SWD, reset et les pins d'alimentation comme indiqué dans la **figure 2.17**.

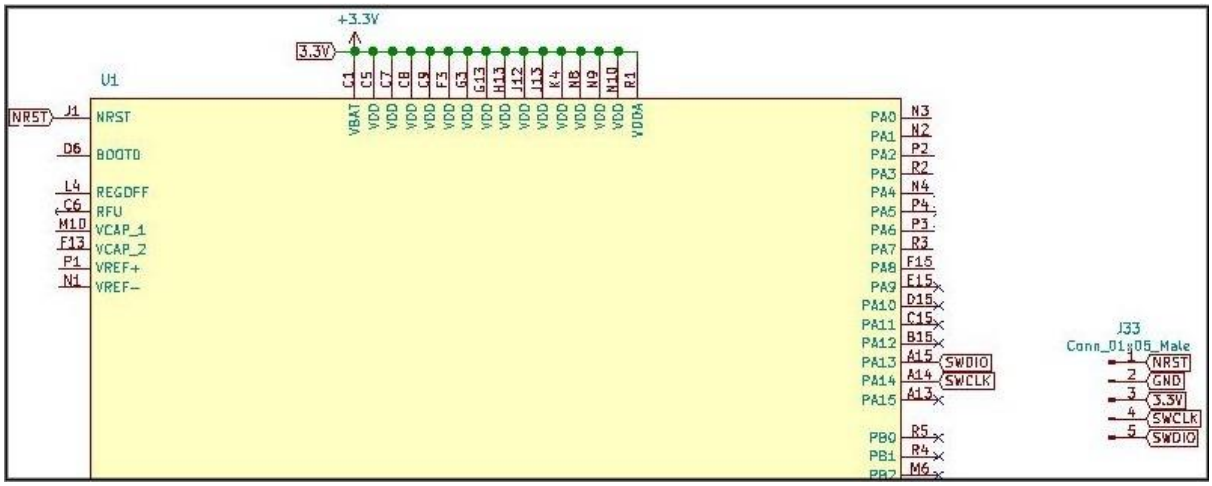

Figure 2.17 : SWD

Et pour programmer le microcontrôleur nous avons besoin du Débogueur/programmeur en circuit ST-LINK/V2 pour STM8 et STM32.

Figure 2.18 : Débogueur/programmeur en circuit ST-LINK/V2

La **figure 2.19** illustre la connexion des cinq pins du stm32 avec le ST-LINK/V2.

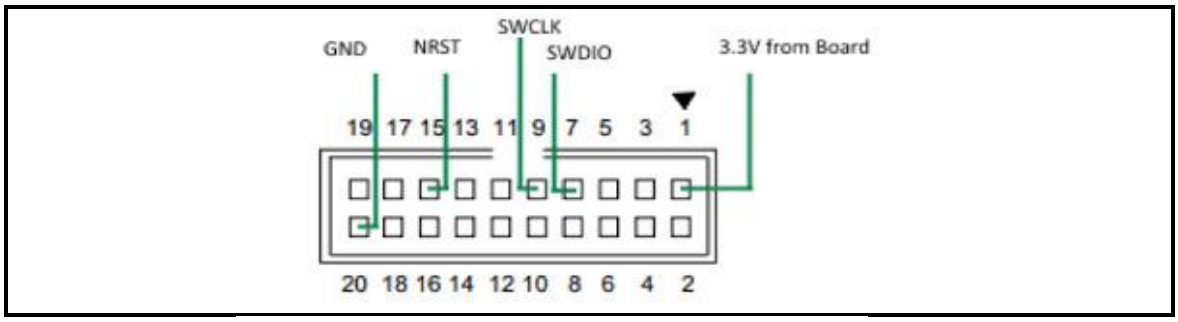

Figure 2.19 : Connexion du MCU avec le ST-LINK/V2

- **Communication io-device/capteur**

La communication entre le capteur et le io-device se fait à l'aide d'un bus à circuits inter-intégrés (I2C) qui utilise deux lignes, les données série (SDA) et l'horloge série (SCL), pour transférer les informations entre les appareils connectés au bus. Les appareils I2C ont des sorties à drain ouvert. Lorsqu'un périphérique I2C passe à un niveau bas, la sortie des périphériques met le bus à la terre. Lorsqu'un périphérique I2C passe à l'état haut, la sortie de l'appareil passe dans un état Z élevé où le bus est tiré jusqu'à VDD par une résistance de rappel connectée entre le bus et VDD. Les résistances de rappel ainsi que la capacité du câblage ou du bus créent une constante de temps RC de charge. Si vous utilisez un câblage externe ou des bus très longs, la capacité totale du bus augmente, ce qui augmente le temps de montée du signal et réduit la fréquence de fonctionnement maximale. La capacité totale du bus augmente également avec le nombre d'appareils connectés au bus. La **figure 2.20** illustre les connexions de deux dispositifs I2C de base à un bus I2C [12].

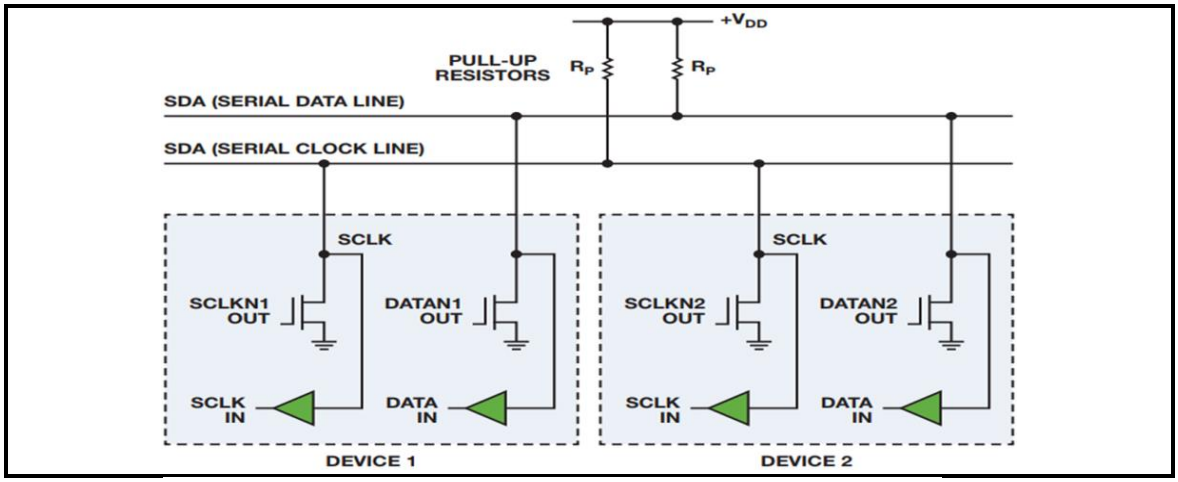

Figure 2.20 : Deux appareils I2C de base connectés à un bus I2C.

Nous avons ajouté aussi les broches d'alimentation pour alimenter le capteur, la **figure 2.21** illustre les connexions entre le capteur et le microcontrôleur dans le logiciel.

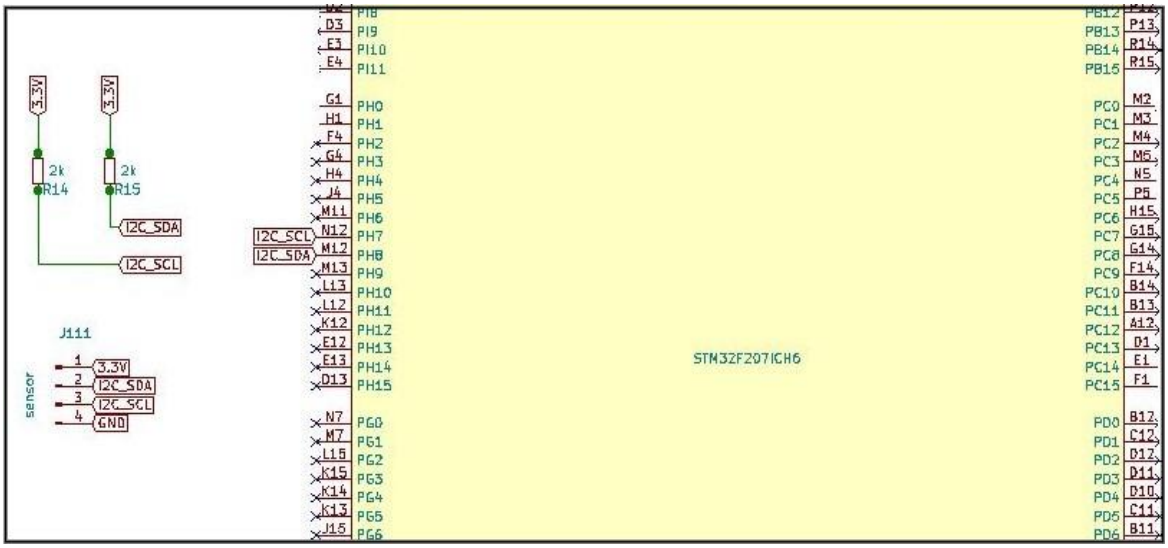

Figure 2.21 : Connexion capteur/io-device.

- **MP2451DT-LF-Z**

Avant de procéder à configurer le MP2451DT-LF-Z, nous devons d'abord suivre la fiche technique du circuit intégré, la **figure 2.22** illustre les conditions de fonctionnement recommandées et maximales du circuit [13].

ABSOLUTE MAXIMUM RATINGS (1)

Supply Voltage (V_{IN}) ..... -0.3V to 40V
Switch Voltage (V_{SW}) ..... -0.3V to $V_{IN\ (MAX)}$ + 0.3V
BST to SW ..... -0.3 to 6.0V
Enable (V_{EN}) 8V
Enable Sink Current (I_{EN}) 100µA
All Other Pins ..... -0.3V to 5.0V
Continuous Power Dissipation (T_A=+25°C) (2) 0.57W
Junction Temperature ..... 150°C
Lead Temperature ..... 260°C
Storage Temperature ..... -65°C to 150°C

Recommended Operating Conditions (3)

Supply Voltage V_{IN} ..... 3.3V to 36V
Output Voltage V_{OUT} ..... +0.8V to 0.8*V_{IN}
Operating Junction Temp. (T_J). -40°C to +125°C

Figure 2.22 : Conditions de fonctionnement.

Par contre, la **figure 2.23** illustre le rendement en fonction du courant lorsque Vout est égale à 3,3V [13].

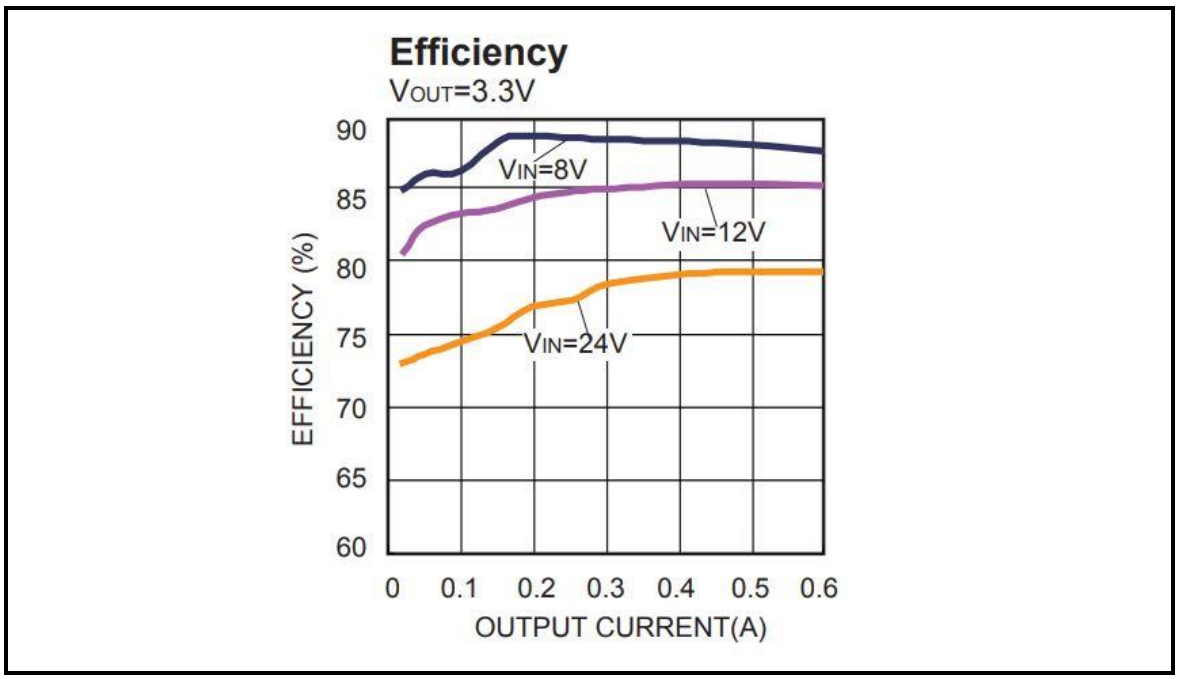

Figure 2.23 : CARACTÉRISTIQUES DE PERFORMANCES TYPIQUES.

- **CIRCUITS D'APPLICATION TYPIQUES**

Nous avons configuré le circuit MP2451DT-LF-Z comme l'indique la **figure 2.24**, un circuit d'application typique lorsque VOUT est égale à 3.3V et VIN de 6V à 24V [13].

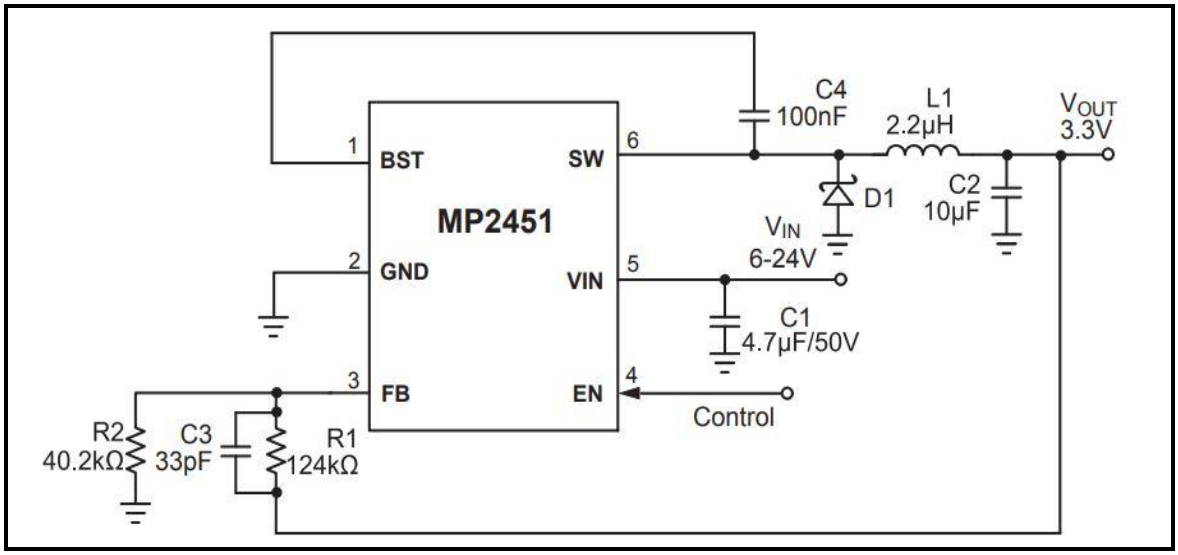

Figure 2.24 : Schéma d'application typique de la sortie 3,3 V.

Le rôle de la 4 ième broche EN (enable) est d'activer ou désactiver le CI, pour 0V, le CI est désactivé alors que pour de 1.55V à 7.5V, le CI est activé.

Nous avons ajouté un circuit simple composé de deux résistances et nous avons appliqué la loi du diviseur de tension pour que le CI soit activé comme l'indique la **figure 2.25**.

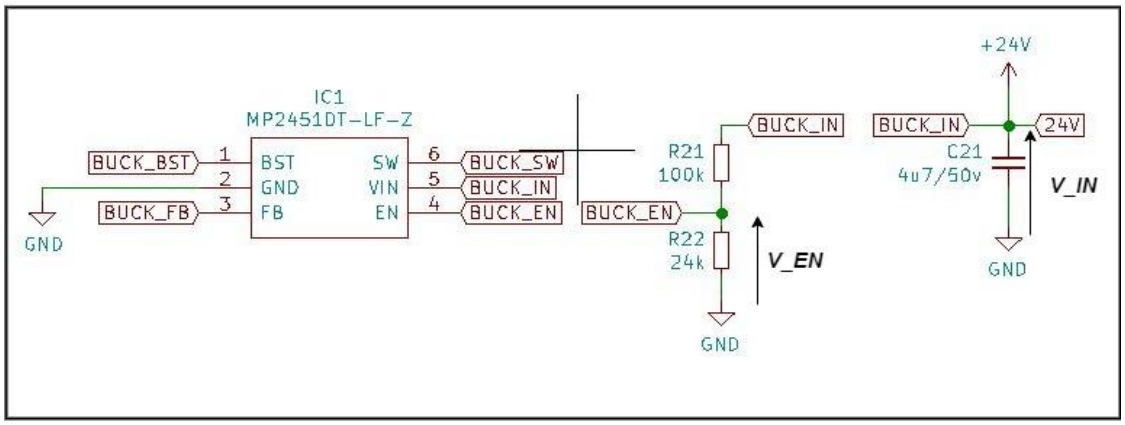

Figure 2.25 : Configuration de la broche ENABLE.

$$V\_EN = \frac{R22}{R22 + R21} x\ V\_IN$$

V_EN = 0.193 x V_IN

Pour que le CI soit activé il faut que V_EN soit supérieur ou égal à 1.55V et inférieur à 7.5V.

- **<u>Calcul des valeurs minimale et maximale du V_IN :</u>**

$1.55 \leq 0.193$x V_IN < 7.5

$$\frac{1.55}{0.193} \leq V\_IN < \frac{7.5}{0.193}$$

8.03V ≤V_IN < 38.86V

Donc

V_IN_minimale = 8.03V

V_IN_maximale = 24V car 24V est la tension maximale qui alimente notre circuit.

- **Schéma utilisé dans le logiciel**

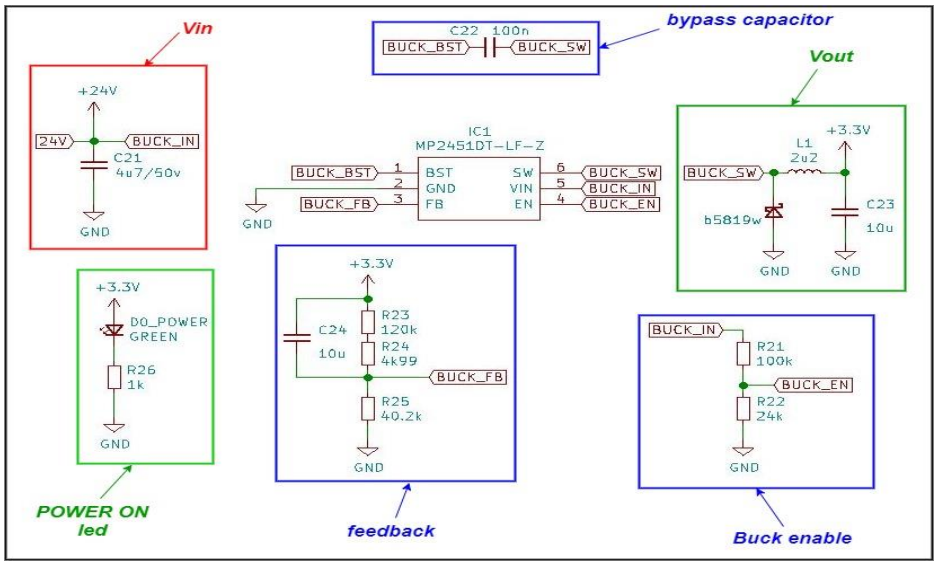

Figure 2.26 : Schéma électrique du bloc power management.

- **DP83849IFVS/NOPB**

- Diagramme système :

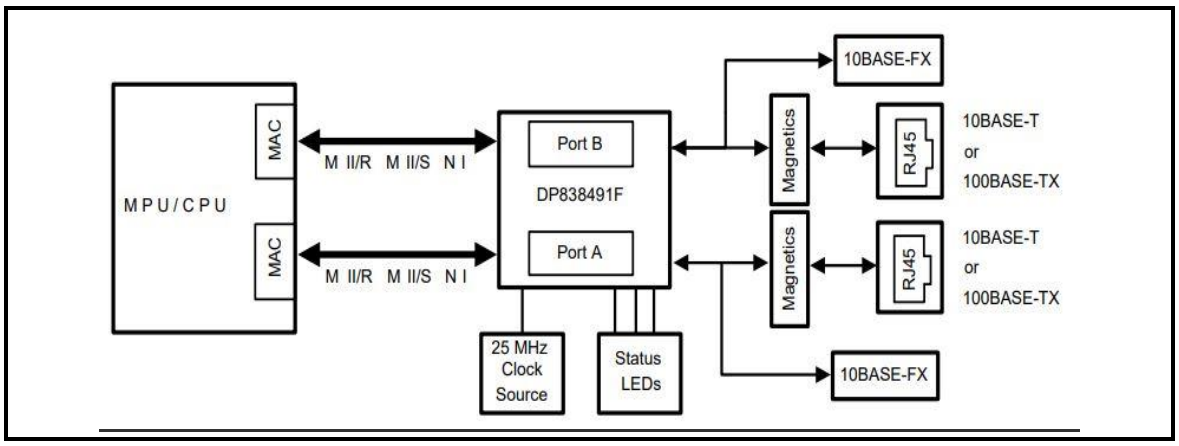

Figure 2.27 : Application typique du dp83849if [14].

- **AFFECTATION FLEXIBLE DE PORT**

Le DP83849IF prend en charge un schéma d'affectation flexible pour chacun des canaux à l'interface MII/RMII, L'un ou l'autre des ports MII peut être affecté aux canaux internes A/B. Ces valeurs sont contrôlées par le RMII et registre de contournement (RBR), adresse 17h. Les affectations de transmission et les affectations de réception peuvent être faites séparément pour permettre encore plus de flexibilité

(c'est-à-dire que les deux canaux pourraient transmettre de MII A tout en restant permettant des chemins de réception séparés pour les canaux).

De plus, le canal de réception opposé peut être utilisé comme source d'émission pour chaque canal. Comme montré dans la **figure 2.28** Les données de réception du canal A peuvent être utilisées comme source de données de transmission du canal B tandis que les données de réception du canal B peuvent être utilisées comme source de données de transmission du canal A. Pour une synchronisation correcte de l'horloge, cette fonction nécessite que l'appareil soit en mode RMII ou en mode de fonctionnement Single Clock MII. Une sangle de configuration est fournie sur la broche 56, RXD2_B/EXTENDER_EN pour activer ce mode.

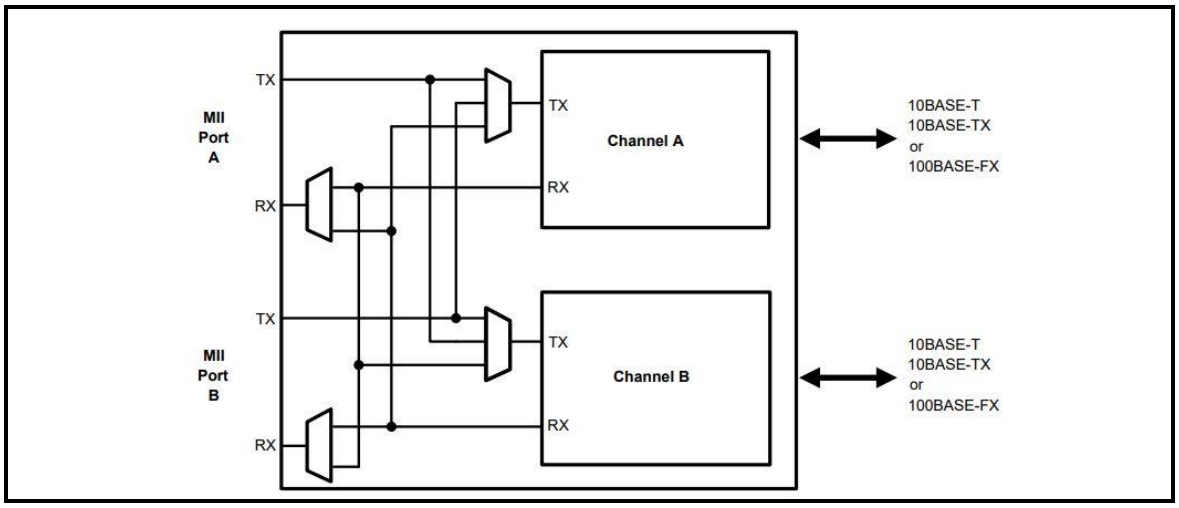

Figure 2.28 : Port switching [14].

Nous avons seulement utilisé un seul MAC pour notre application comme l'indique la **figure 2.29**

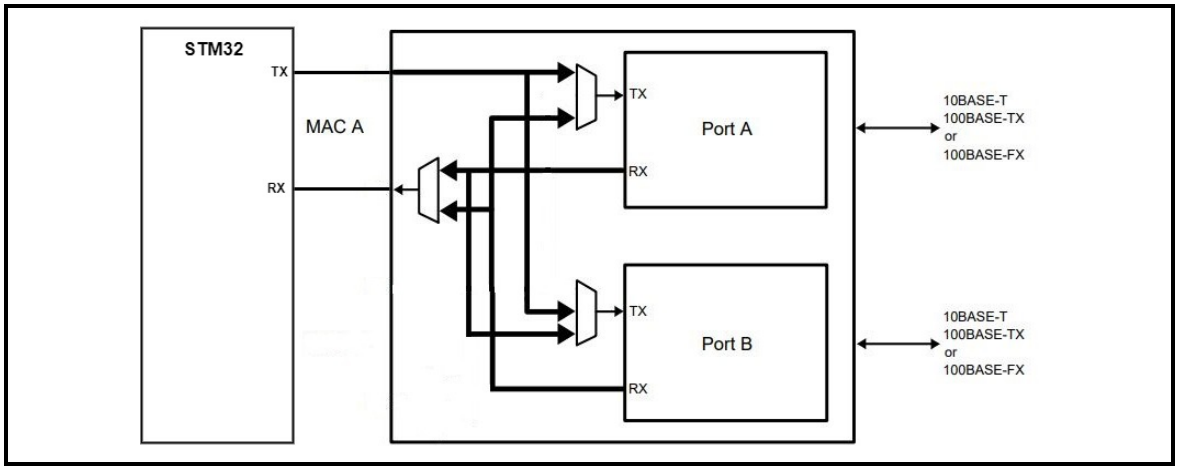

Figure 2.29 : Commutation MAC/ports.

- **Description des broches :**

Les broches du DP83849IF sont classées dans les catégories d'interface suivantes (chaque interface est décrite sous forme de tableau dans les sections qui suivent) [14] :

- Interface de gestion série
- Interface MAC Data
- Interface d'horloge
- Interface LED
- Interface JTAG
- Réinitialiser et éteindre
- Option sangle
- Interface PMD 10/100 Mb/s
- Broches de connexion spéciales
- Broches d'alimentation et de terre

Toutes les broches de signal du DP83849IF sont des cellules d'E/S, quelle que soit l'utilisation particulière. Les définitions ci-dessous définissent la fonctionnalité des cellules d'E/S pour chaque broche.

Type: I Input

Type: O Output

Type: I/O Input/Output

Type: OD Open Drain

Type: PD, PU Internal pulldown/pullup

Type: S Strapping pin (All strap pins have weak internal pull-ups or pull-downs. If the default strap value is to be changed then an external 2.2 kΩ resistor should be used).

La **figure 1 annexe B** illustre le Diagramme de connexion du DP83849IF

- **Reduced media-independent interface (RMII)**

La spécification RMII (reduced media-independent interface) réduit le nombre de broches entre le périphérique Ethernet du microcontrôleur et l'Ethernet externe en 10/100 Mbit/s.

Selon la norme IEEE 802.3u, un MII contient 16 broches pour les données et le contrôle. La spécification RMII est dédiée à réduire le nombre de broches à 7 broches (une diminution de 62,5% du nombre de broches).

Le RMII est instancié entre le MAC et le PHY. Cela aide à la traduction des MAC MII vers le RMII. Le bloc RMII a les caractéristiques suivantes :

• Il prend en charge les taux de fonctionnement de 10 Mbit/s et 100 Mbit/s

• La référence d'horloge doit être doublée à 50 MHz

• La même référence d'horloge doit provenir de l'extérieur à la fois du MAC et de l'extérieur le bloc PHY Ethernet Il fournit des chemins de transmission et de réception de données indépendants sur 2 bits (débit)

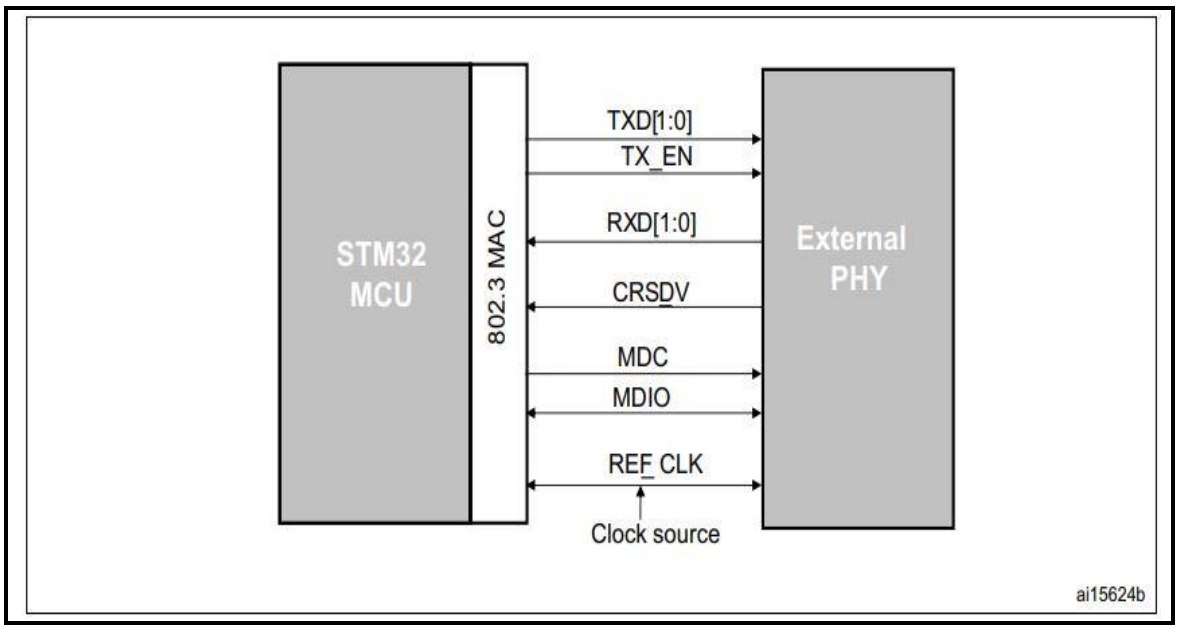

Figure 2.30 : Reduced media-independent interface signals [11].

Schéma utilisé dans le logiciel

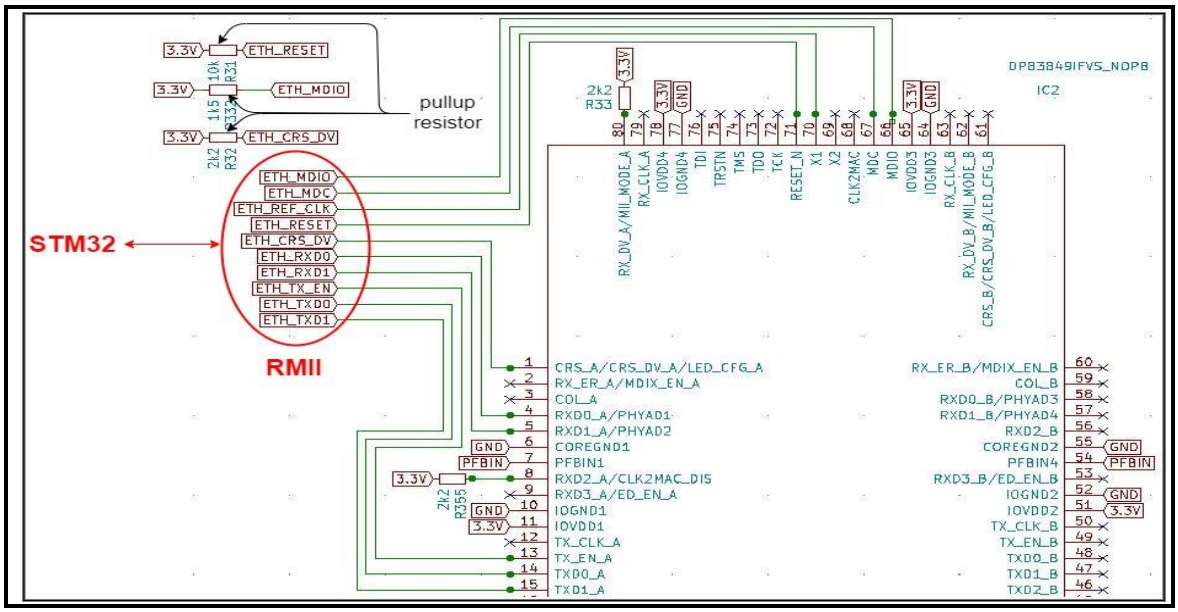

Figure 2.31 : Configuration RMII PHY-MAC.

La description de l'interface horloge se trouve à l'annexe B **Tableau 3**.

- **Sources d'horloge RMII**

Comme décrit dans la **figure 2.32**, les Sources d'horloge RMII, les STM32F20x et STM32F21x, STM32F107xx pourraient fournir le signal d'horloge de 50 MHz sur la broche de sortie MCO et on doit ensuite configurer cette valeur de sortie via la configuration PLL [11].

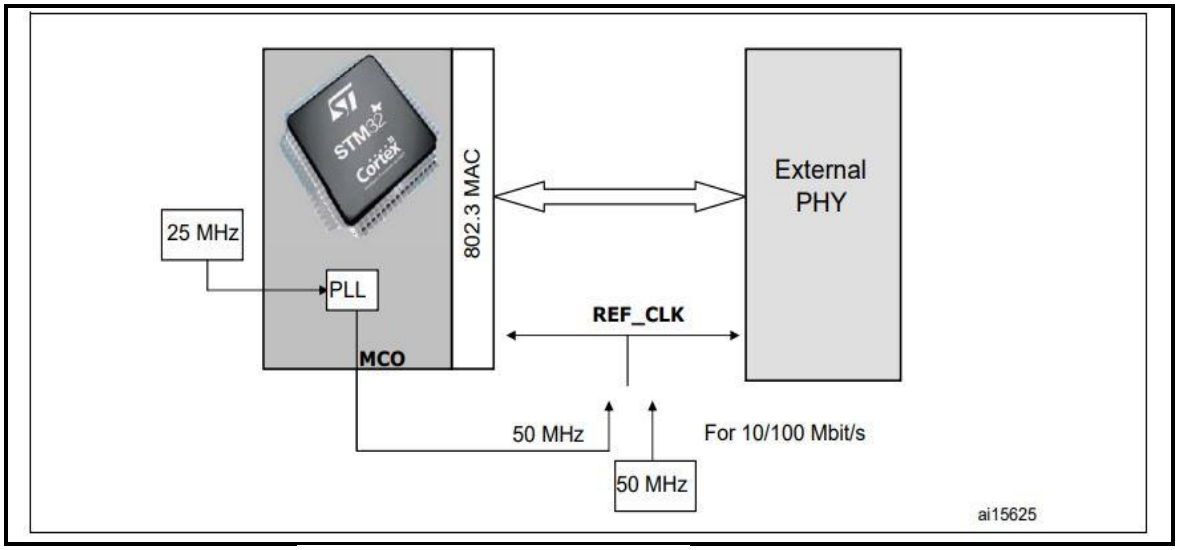

Figure 2.32 : Sources d'horloge RMII [11].

En utilisant cette méthode, voir l'annexe B **Tableau 2** nous économisons la taille et le coût du I/O device.

- Interface LED

Le DP83849IF prend en charge trois broches LED configurables. Les LED prennent en charge deux modes de fonctionnement qui sont sélectionnés par la sangle de mode LED et un troisième mode de fonctionnement qui est configurable par registre.

Les définitions des LED pour chaque mode sont détaillées dans l'annexe B **Tableau 7** et **4** Étant donné que les LED sont également utilisées comme options de sangle, la polarité de la sortie LED dépend du fait que la broche est tirée vers le haut ou vers le bas [14].

- **Interface JTAG (voir Annexe B, Tableau 5)**
- **Réinitialiser et éteindre (voir Annexe B, Tableau 6)**
- **Option sangle**

Le DP83849IF utilise de nombreuses broches fonctionnelles comme options de sangle. Les valeurs de ces broches sont échantillonnées pendant la réinitialisation et utilisées pour attacher l'appareil à des modes de fonctionnement spécifiques. La broche option sangle est donné par les affectations définies dans l'annexe B **tableau 10**. Le nom de la broche fonctionnelle est indiqué entre parenthèses.

Une résistance de 2,2 kΩ doit être utilisée pour le pull-down ou pull-up pour changer l'option de sangle par défaut. Si la valeur par défaut l'option est requise, alors il n'y a pas besoin de résistances pull-up ou pull down externes. Étant donné que ces broches peuvent avoir des fonctions alternatives une fois la réinitialisation désactivée, ils ne doivent pas être connectés directement à VCC ou GND.

Le DP83849IF prend en charge trois broches de diodes électroluminescentes (LED) configurables pour chaque port.

Plusieurs fonctions peuvent être multiplexées sur les trois LED en utilisant trois modes de fonctionnement différents. Le mode de fonctionnement LED peut être

sélectionné en écrivant dans les bits du registre LED_CFG[1:0] dans le contrôle PHY Registre (PHYCR) à l'adresse 19h, bits [6:5]. De plus, LED_CFG[0] pour chaque port peut être défini par un strap sur les broches CRS_A et CRS_B. LED_CFG[1] n'est contrôlable que par l'accès au registre et ne peut pas être défini comme broche de sangle

Voir le **tableau 4** annexe B pour la sélection du LED mode.

Étant donné que les options de sangle d'auto-négociation (AN) partagent les broches de sortie LED, les composants externes requis pour le cerclage et l'utilisation des LED doivent être pris en compte afin d'éviter les conflits.

Plus précisément, lorsque les sorties LED sont utilisées pour piloter directement les LED, l'état actif de chaque pilote de sortie est fonction du niveau logique échantillonné par l'entrée AN correspondante lors de la mise sous tension/réinitialisation. Par exemple, si une entrée AN donnée est abaissée de manière résistive, la sortie correspondante sera configurée comme une entrée active conducteur élevé. Inversement, si une entrée AN donnée est élevée de manière résistive, alors la sortie correspondante sera configurée comme un pilote faible actif.

On se reporte à la Figure 2.33 pour un exemple de connexions AN aux composants externes au port A. Dans cet exemple, le cerclage AN entraîne la désactivation de l'auto-négociation avec 100 Full-Duplex forcé.

La nature adaptative des sorties LED permet de simplifier les problèmes potentiels de mise en œuvre de ces doubles épingles à usage.

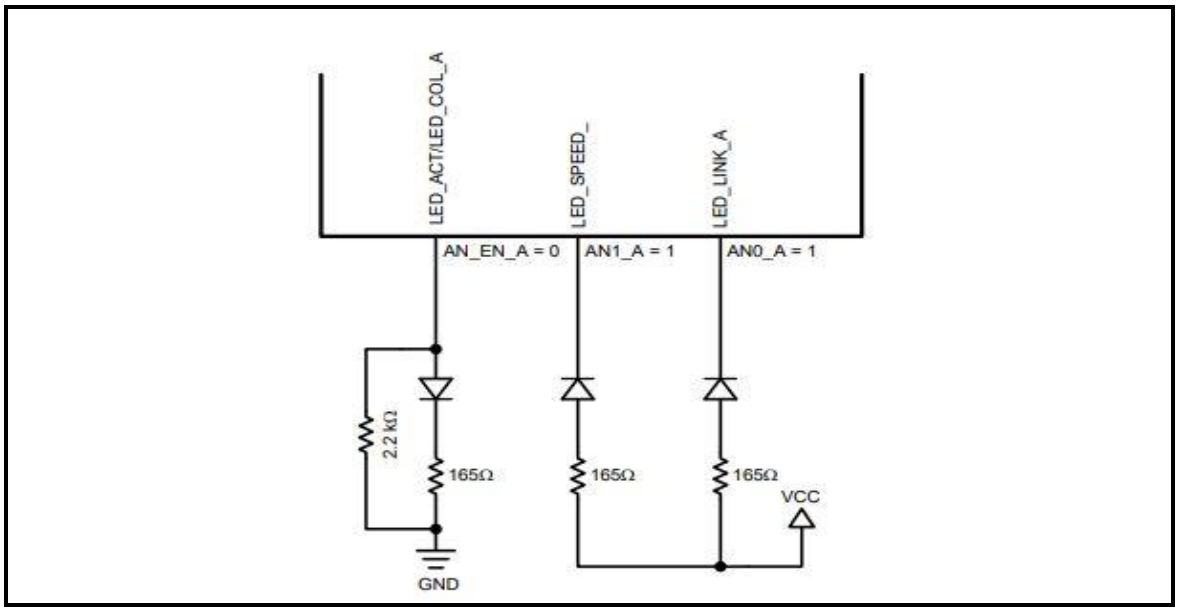

Figure 2.33 : Exemple de cerclage AN et de chargement de LED.

- **Interface PMD 10/100 Mb/s (**voir Annexe B, **Tableau 8**)
- **Directives de conception**

La **figure 2.34** montre le circuit recommandé pour un 10/100 Interface paire torsadée Mb/s. [15]

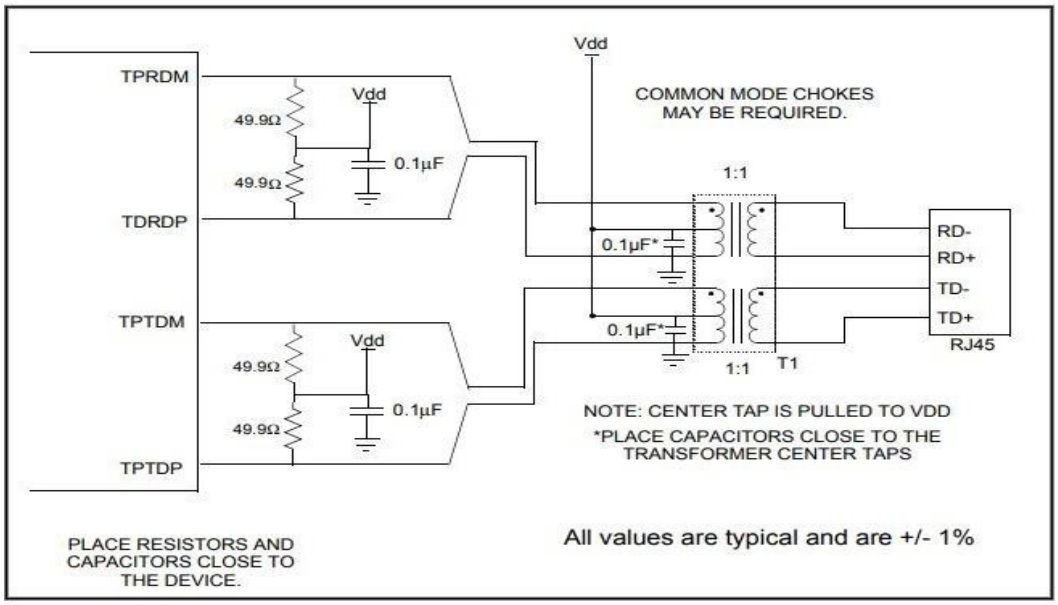

Figure 2.34 : Circuit réseau.

Circuit utilisé dans le logiciel

La **figure 2.35** illustre le schéma du connecteur rj45 utilisé

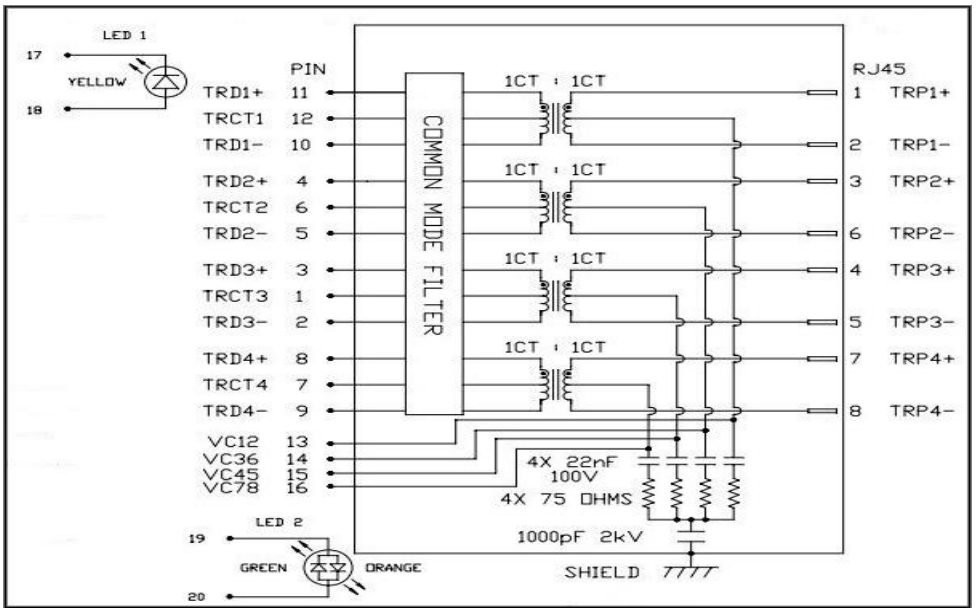

Figure 2.35 : Schéma du connecteur RJ45 0826-1X1T-KL-F [16].

La **figure 2.36** et la **figure 2.37** illustre la configuration du dp83849if avec les connecteur rj45 du port A et port B

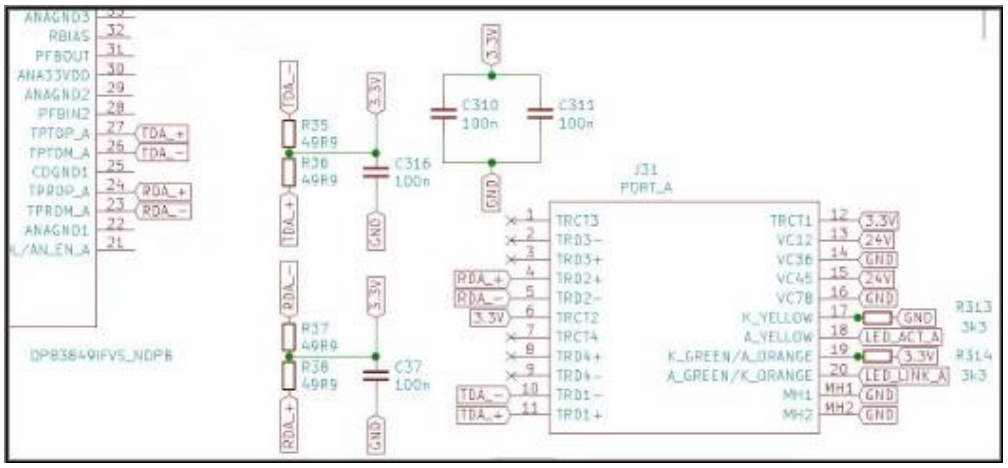

Figure 2.36 : Branchement du PHY avec le port_a

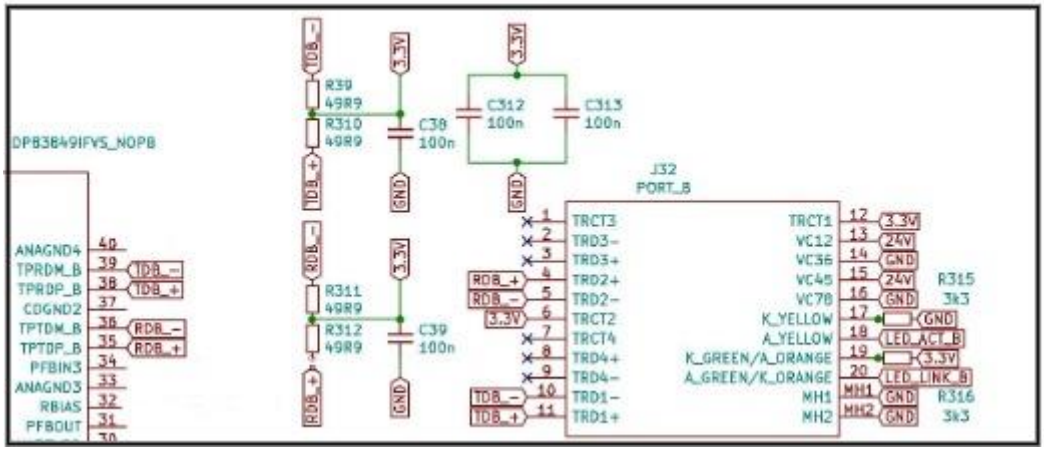

Figure 2.37 : Branchement du PHY avec le port_b

- **Broches de connexion spéciales** (voir Annexe B, **Tableau 9**)
- **POWER FEEDBACK CIRCUIT**

Pour garantir un fonctionnement correct du DP83849IF, des condensateurs parallèles avec des valeurs de 10 µF et 0,1 µF doivent être placés à proximité de la broche 31 (PFBOUT) de l'appareil. La broche 7 (PFBIN1), la broche 28 (PFBIN2), la broche 34 (PFBIN3) et la broche 54 (PFBIN4) doit être connectées à la broche 31 (PFBOUT), chaque broche nécessite un petit condensateur (0,1 µF). Voir la **figure 2.38** pour les connexions appropriées [14].

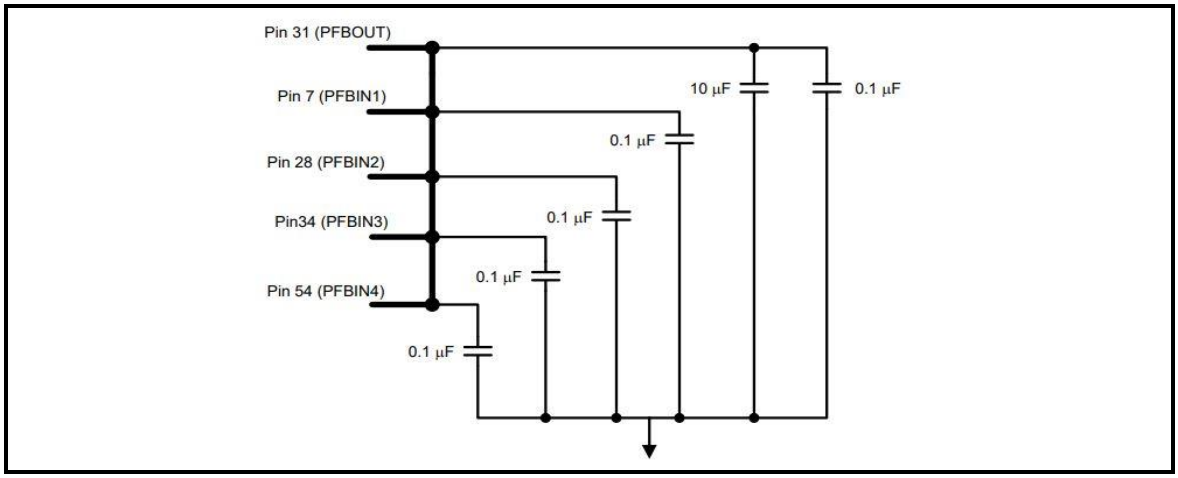

Figure 2.38 : Power feedback connection.

La **figure 2.39** montre le schéma utilisé dans le logiciel pour les broches de connexion spéciales

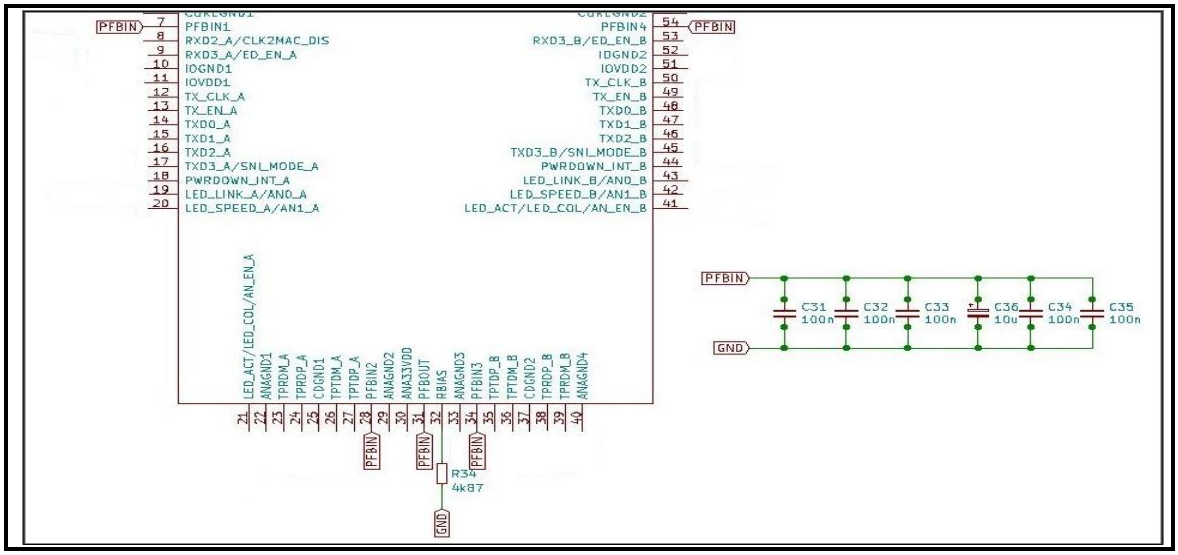

Figure 2.39 : Circuit des broches de connexion spéciales.

- **Broches d'alimentation et de terre**

Signal name	Pin#	Description
IOVDD1, IOVDD2, IOVDD3, IOVDD4	11,51,65,78	I/O 3.3V Supply
IOGND1, IOGND2, IOGND3, IOGND4	10,52,64,77	I/O Ground
COREGND1, COREGND2	6,55	Core Ground
CDGND1, CDGND2	25,37	CD Ground
ANA33VDD	30	Analog 3.3V Supply
ANAGND1, ANAGND2, ANAGND3, ANAGND4	22,29,33,40	Analog Ground

Tableau 2.7 : Broches d'alimentation et de terre.

- **Schéma synoptique final**

Après le choix des composants et de lire leurs descriptions nous avons optimisé le circuit de notre I/O Device en enlevant le quartz de 50 Mhz et en le remplaçant par une source d'horloge fourni par le STM32, voir la **figure 2.32**. Le schéma synoptique final devient comme l'indique la **figure 2.40**

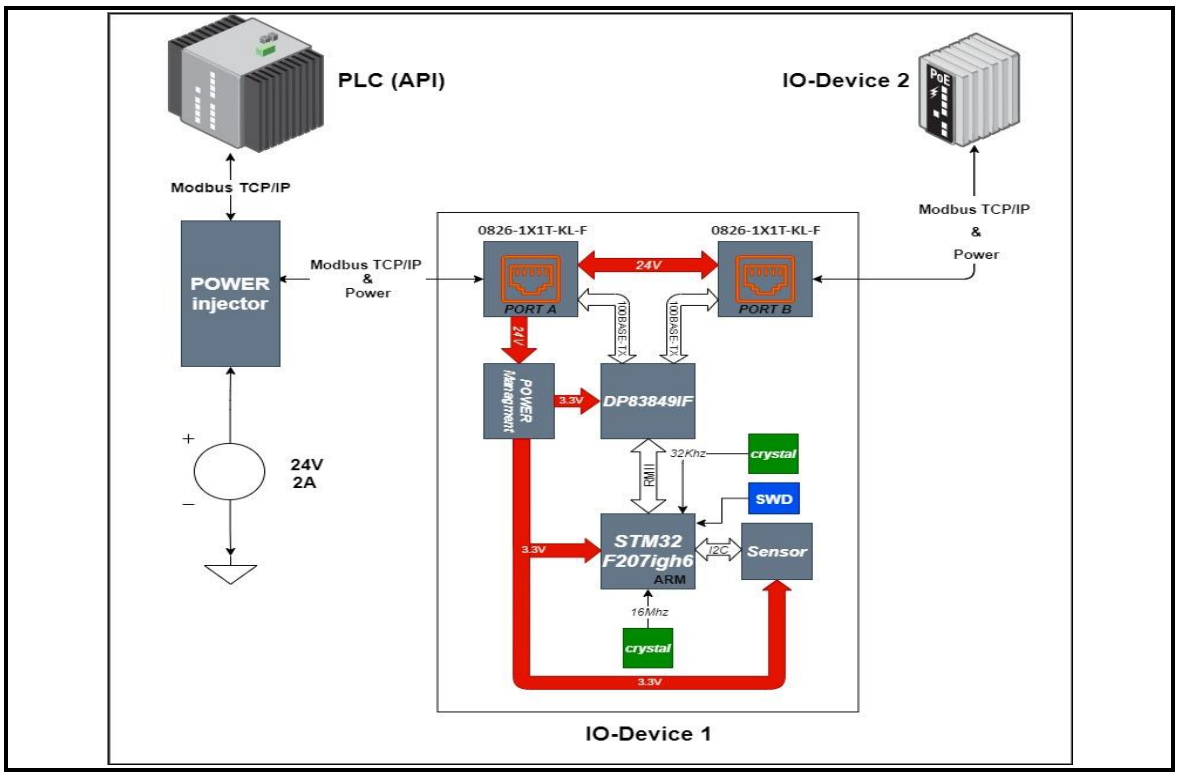

Figure 2.40 : Schéma synoptique final.

2.3 Implémentation software

Les **figure 2.42** et **figure 2.43** Montre la vue globale du côté software du projet.

Le côté software que nous avons réalisé est un programme qui fonctionne sur le microcontrôleur STM32F207IGH6J

Ce programme est constitué d'un système d'exploitation FREERTOS. Ce système permet, une fois intégré dans un microcontrôleur, l'utilisation de plusieurs taches en parallèle

Nous avons créé deux TASK (tâche), une tache pour les serveurs **figure 2.42** et la deuxième pour le capteur **figure 2.43**

Nous avons créé aussi trois serveurs un serveur pour le protocole Bootp, le deuxième protocole Modbus et un autre pour le Protocole ENIP

Comme notre projet est basé sur une communication TCP/IP, on a choisi d'utiliser la bibliothèque LWIP qui va nous permettre de travailler avec des sockets pour simplifier le programme au lieu de travailler avec des trames octet par octet

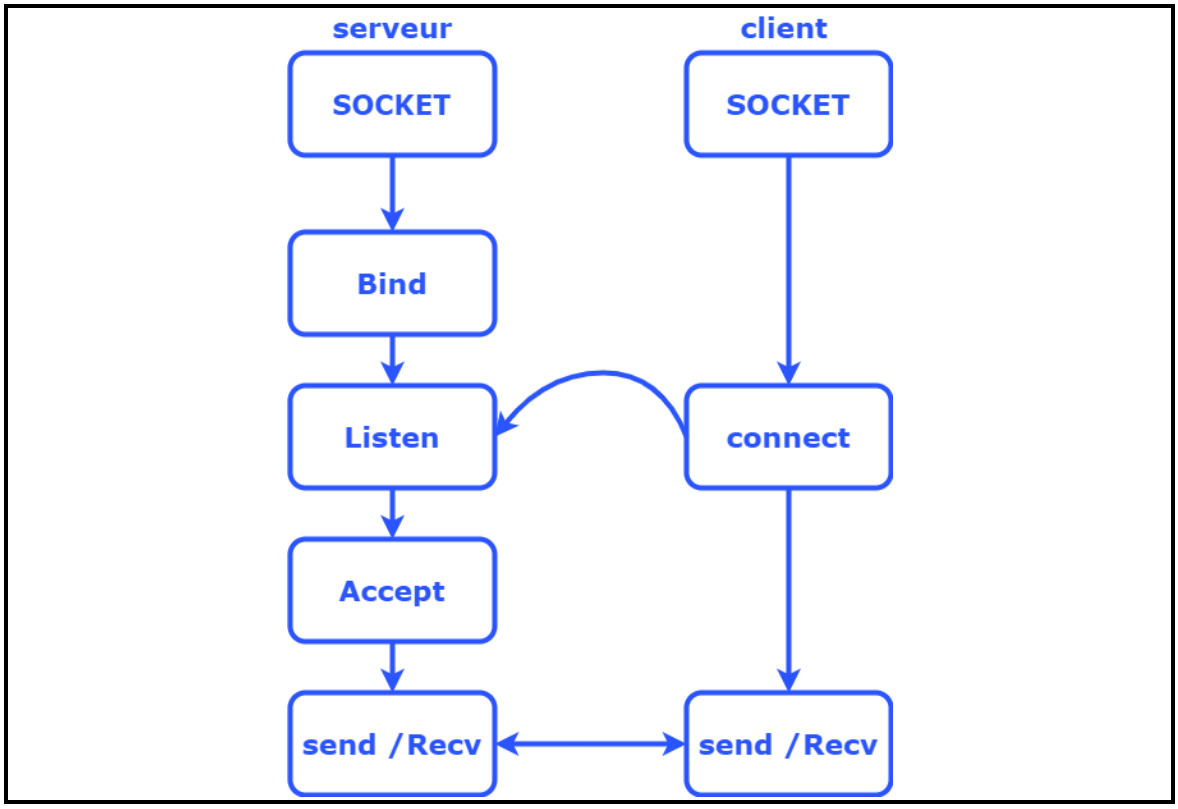

Figure 2.41 : Communication entre client et serveur

Le serveur crée un socket **Figure 2.41** et le lie avec une adresse IP et un Port, puis le serveur entre en mode écoute pour les demandes de connexion des clients. D'un autre côté le client crée un socket et il va essayer de se connecter avec le serveur qui est en mode écoute. En finalité le serveur accepte la connexion et la communication est établie.

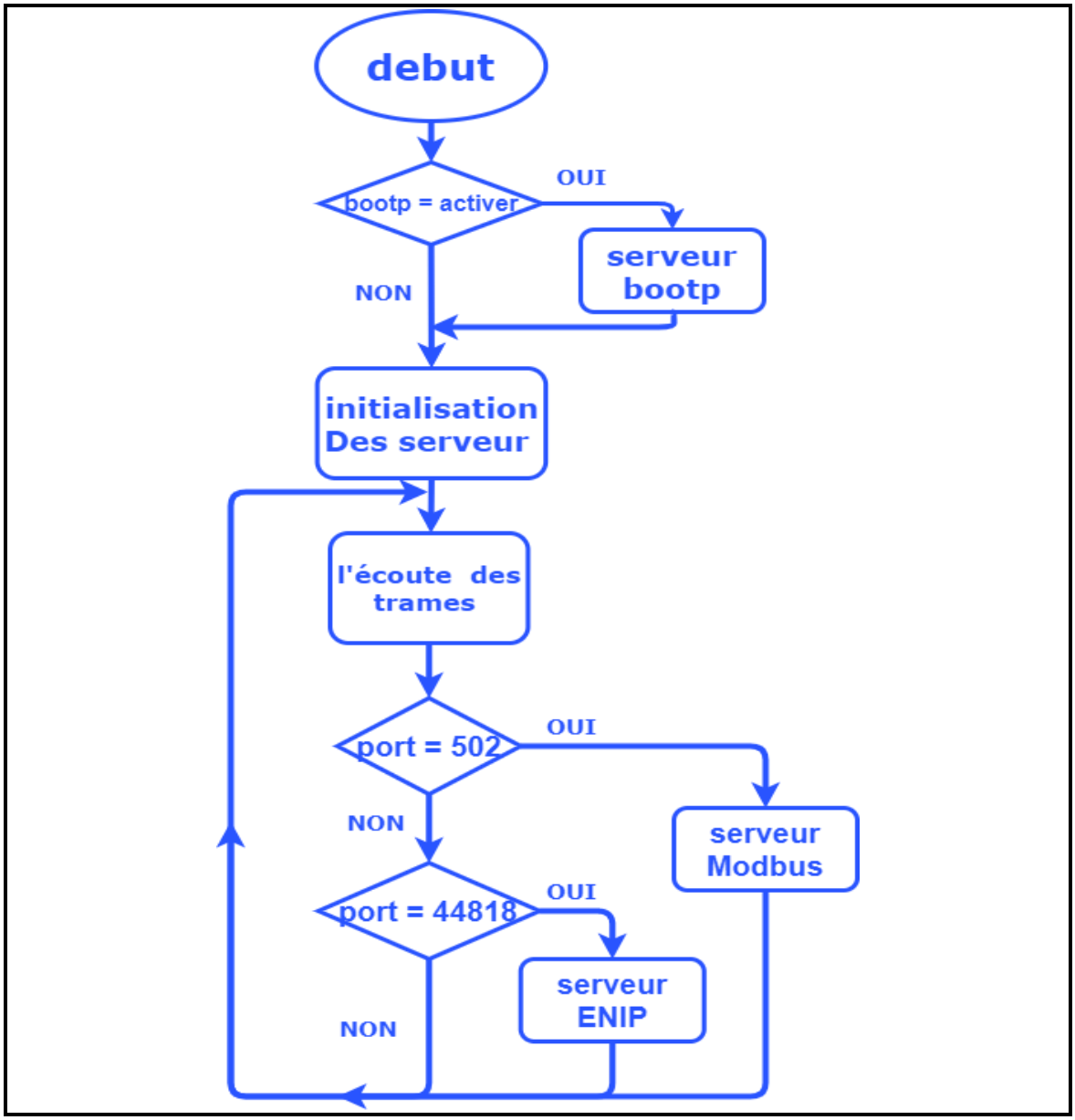

Figure 2.42 : Vue globale du programme

La première tâche, **figure 2.42** Consiste par la vérification si le serveur bootp est activé, il procède à la configuration de l'adresse IP, en envoyant un bootp requête pour la recherche de l'adresse IP, s'il n'est pas activé l'initialisation des serveurs commence avec l'adresse IP enregistré

Apres l'initialisation des serveurs, le dispositif va écouter les trames reçues à partir de l'Ethernet, si la trame reçue contient l'adresse IP du dispositif ,il passe à une autre étape pour vérifier à quelle port la trame a été transmise, si l'adresse IP n'est pas l'adresse du dispositif le programme reste en mode écoute des trames, si la trame est transmise vers le port 502 (port de modbus TCP/IP) il passe vers le serveur modbus qui va traiter cette trame, le serveur modbus gère la communication modbus TCP/IP et répond aux demandes du client (exp : un PLC) , si la trame est transmise vers le port 44818 (port de ENIP) il passe vers le serveur ENIP qui va traiter cette trame et active ou désactive le bootp

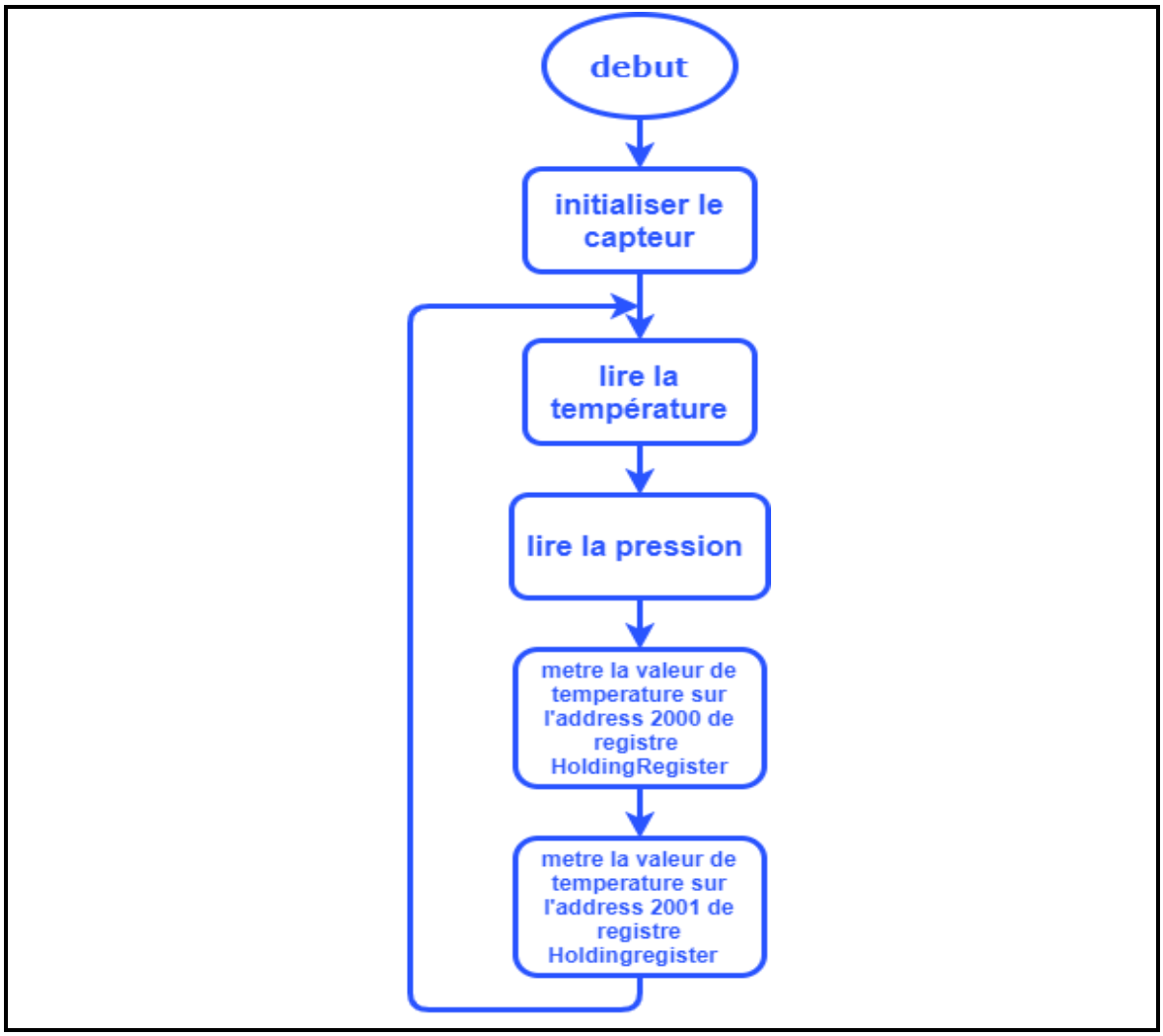

Figure 2.43 : Organigramme du programme de capteur

La deuxième tache **figure 2.43** débute par l'initialisation du pilote de l'interface I2C où le capteur est connecté afin de lire la température et la pression et enregistre chaque donné à sa propre adresse sur un vecteur des variables HlodingRegister

2.3.1 Configuration du STM32F207IGH6

Pour configurer le code de l'initialisation du stm32f207IGH6 on utilise le logicielle **STM32FCubeIDE** de **STMicroelectronics**

a) RCC

Pour configure le RCC il faut :

- Activer le HSE par la sélection de **Crystal/Ceramic Resonator**
- Activer le LSE par la sélection de **Crystal/Ceramic Resonator**
- Activer Mster Clock Outpout pour l'utiliser comme l'horloge d'Ethernet

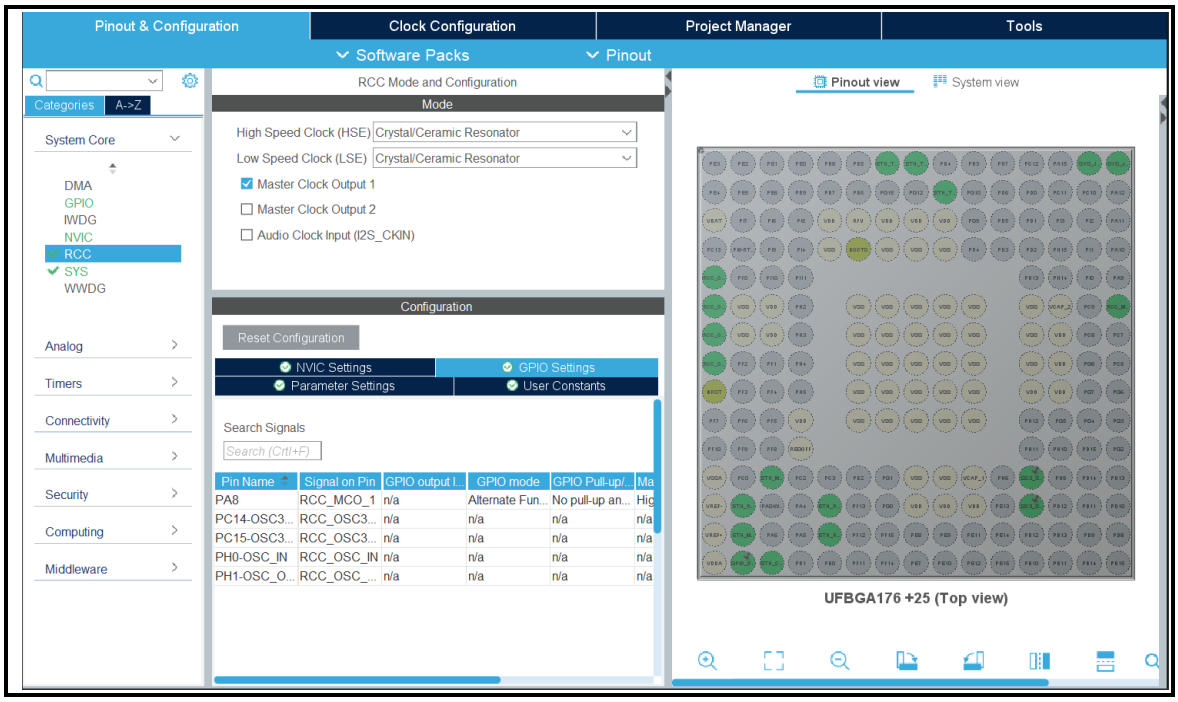

Figure 2.44 : Configuration du RCC

b) L'Horloge

Pour configurer l'horloge il faut :

- Configurer la fréquence d'entrée à 16 Mhz de HSE et 32Khz pour LSE
- Sélectionner la source d'horloge de PLL à partir de HSE
- Sélectionner l'horloge du système 100Mhz à partir PLLCLK

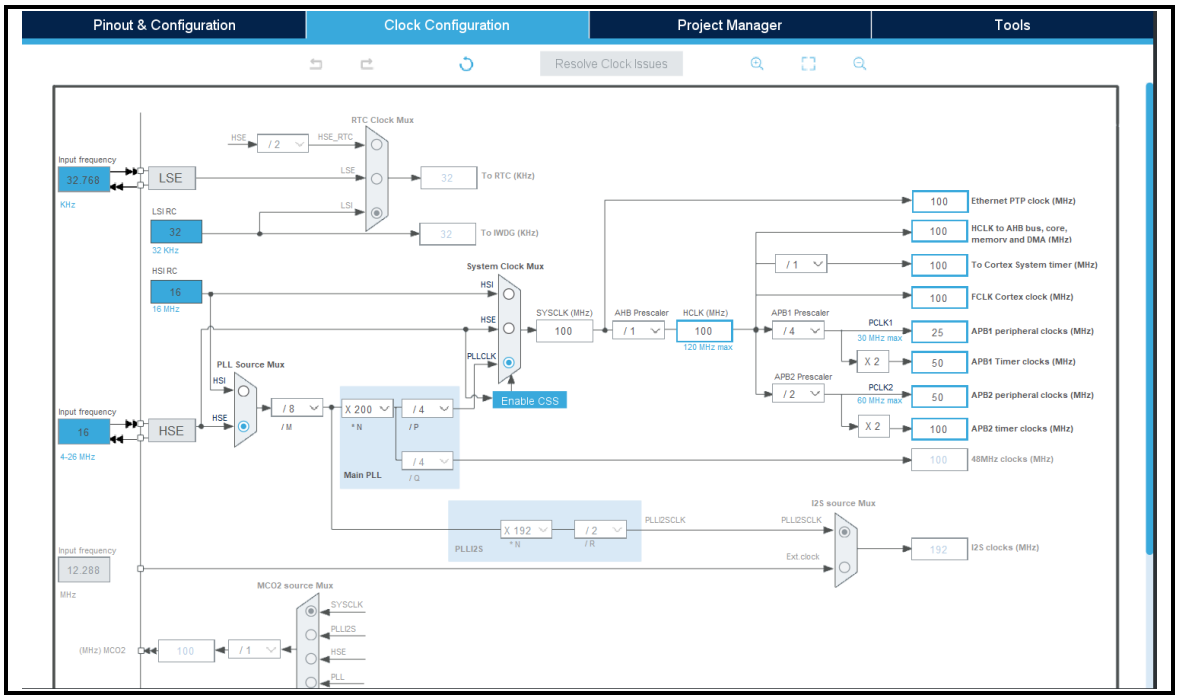

Figure 2.45 : Configuration d'horloge

c) FREERTOS

Pour configurer le FREERTOS il faut :

- Sélectionner l'interface **CMSIS_V1**
- Créer deux taches, la première tâche appelé **server** et la deuxième **sensor**

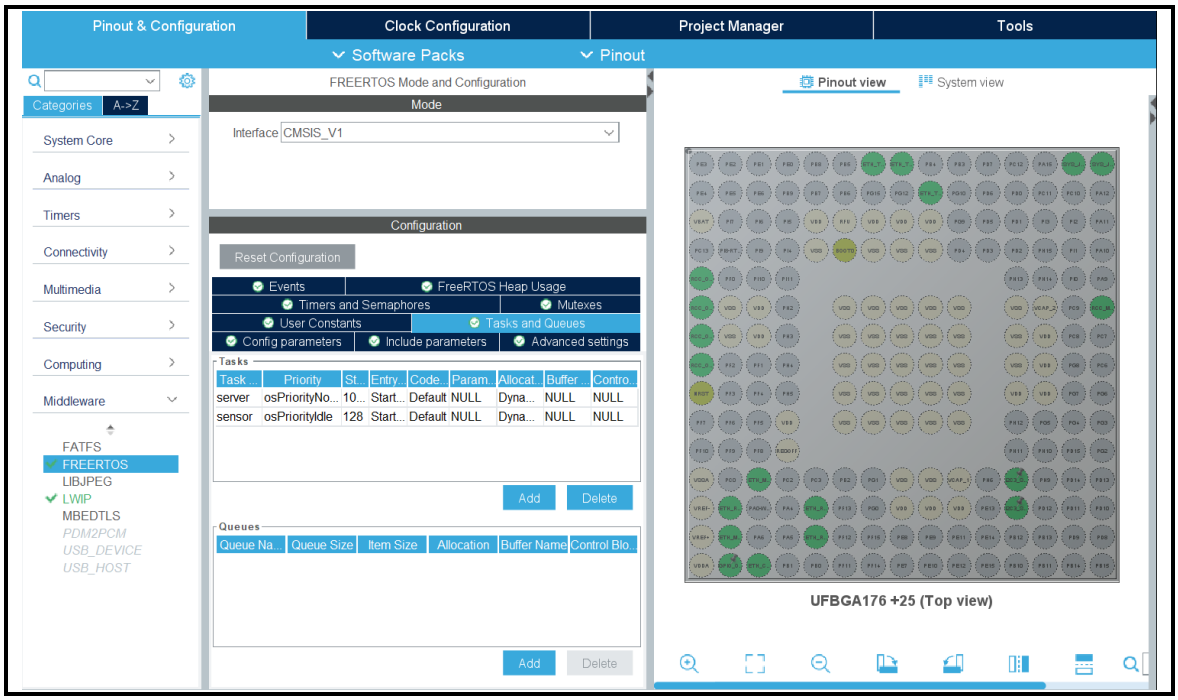

Figure 2.46 : Configuration de FREERTOS

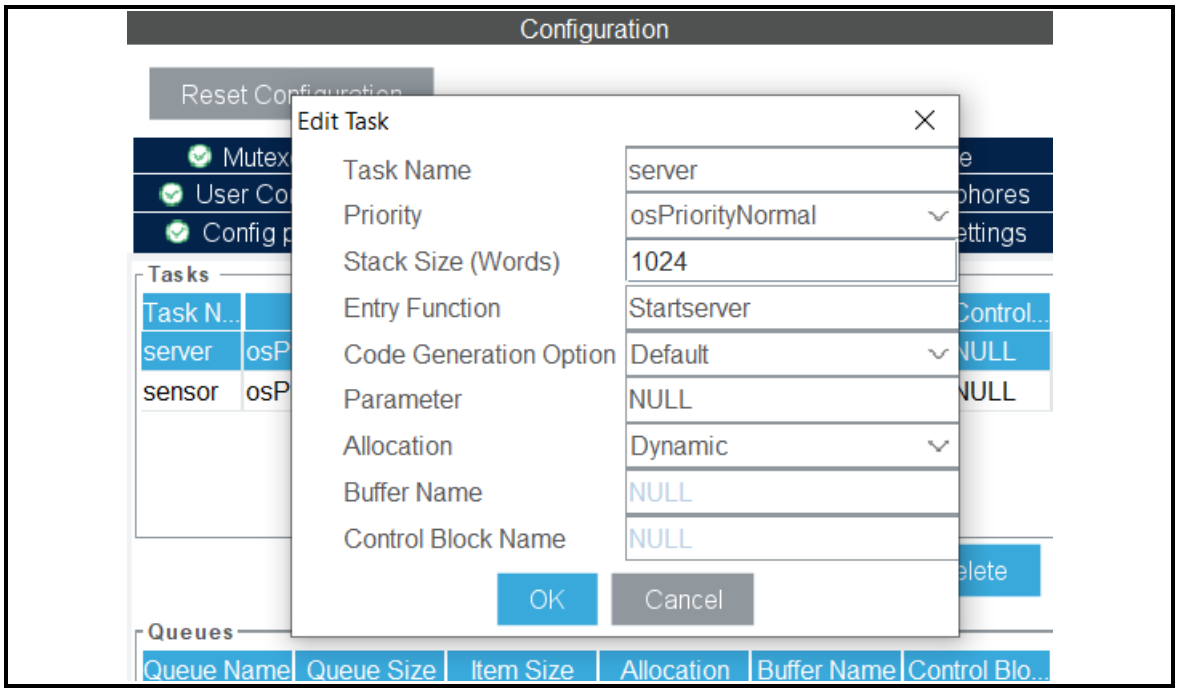

Figure 2.47 : Création du premiers 'task'

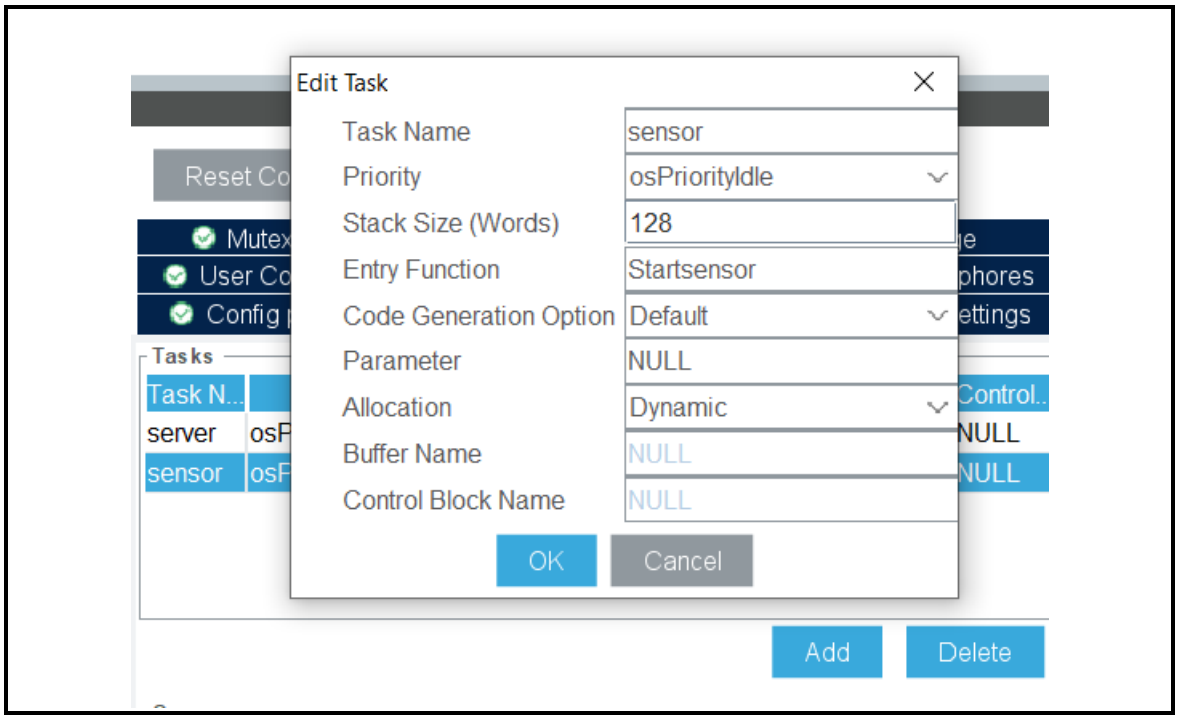

Figure 2.48 : Création du deuxième 'task'

d) Bibliothèque LWIP

Pour configurer le LWIP il faut :

- Activer LWIP
- Configurer l'adresse IP, l'adresse de passerelle et le masque

- Activer ICMP

Comme le montrent les figures **2.49**, **2.50** et **2.51**

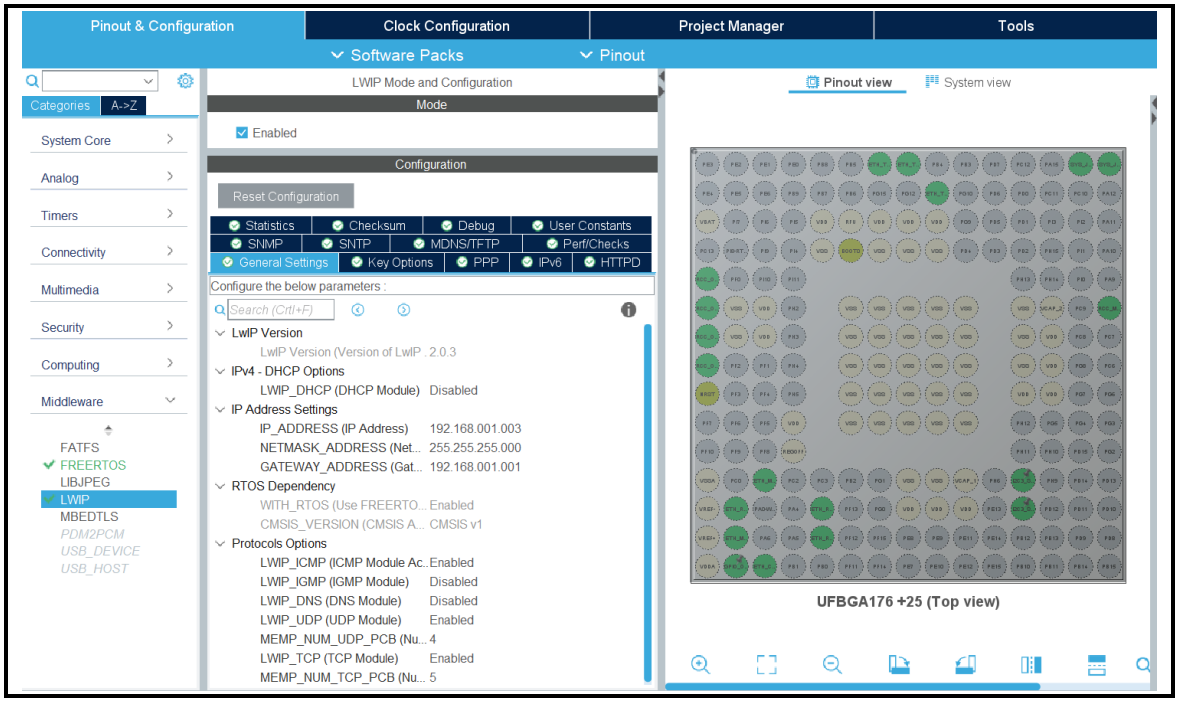

Figure 2.49 : Configuration de LWIP

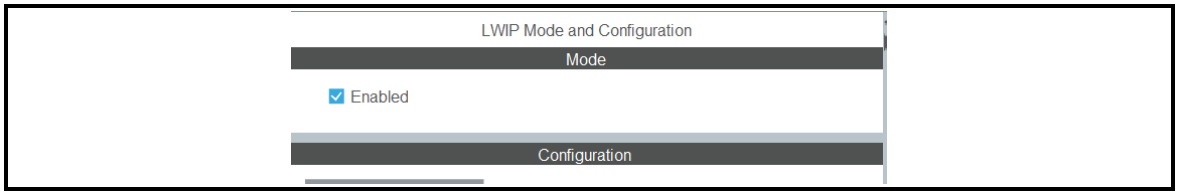

Figure 2.50 : Activation de LWIP

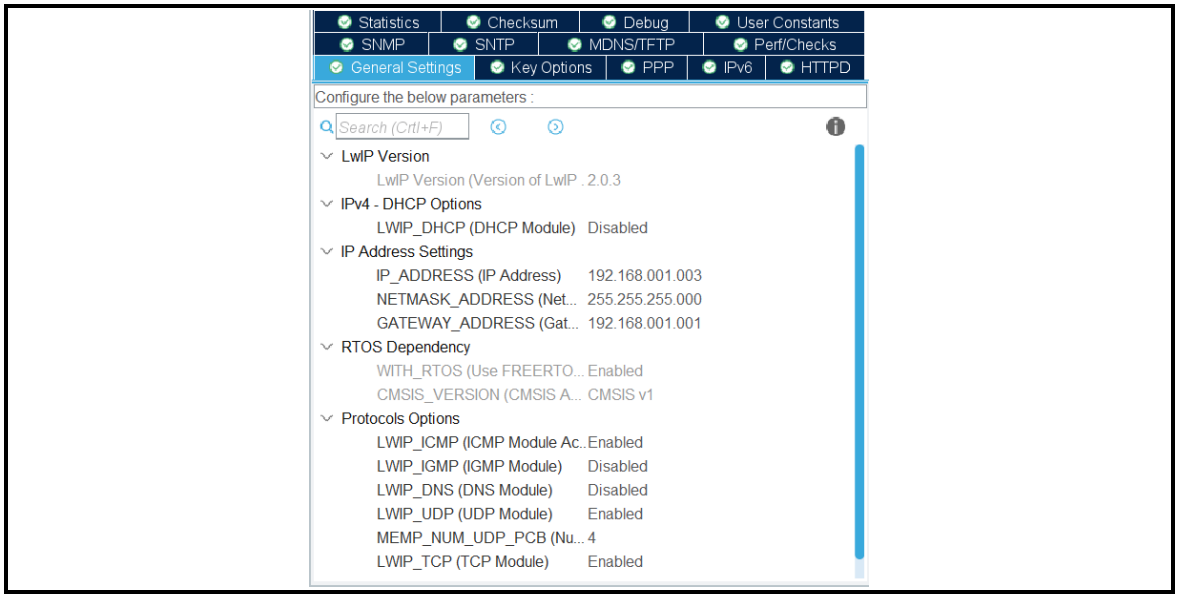

Figure 2.51 : Configuration de l'adresse IP,masque et passerelle

e) SYS

Pour configurer le SYS il faut sélectionner le mode de débogage on Serial Wire **figure 2.52**

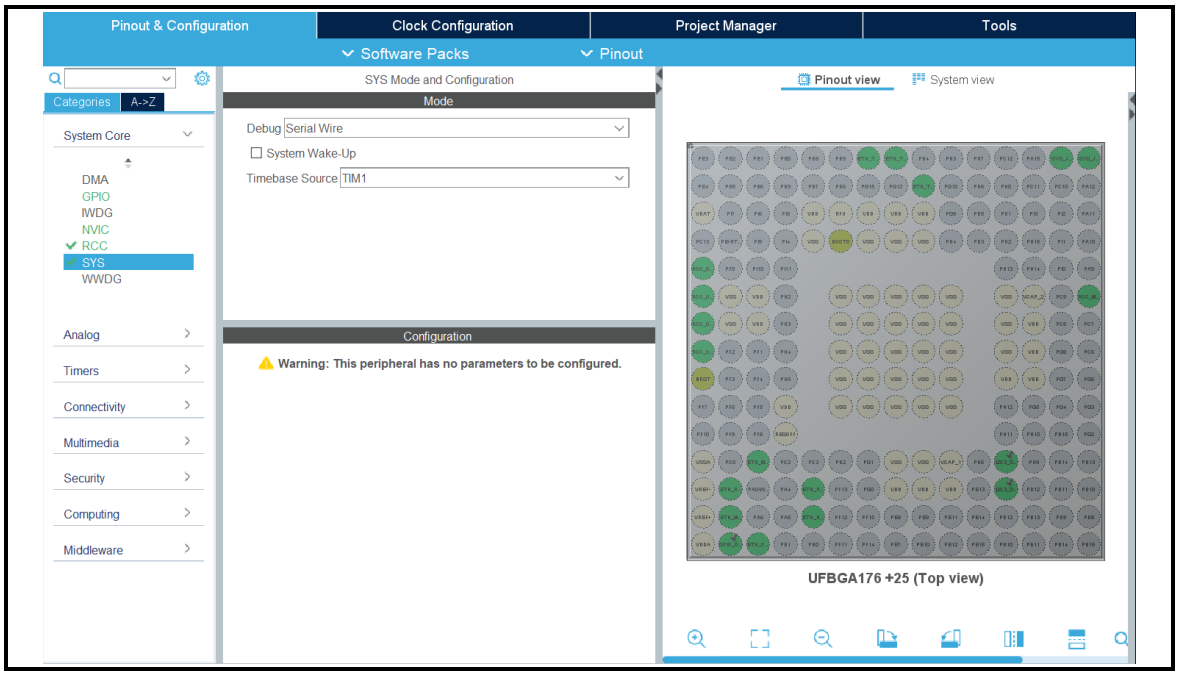

Figure 2.52 : Configuration de SYS (mode de débogage)

f) ETHENRET

Pour configurer le ETH (Ethernet) il faut :

- Sélectionner le mode de l'interface de communication Ethernet RMII
- Active l'auto-négociation
- Sélectionner la vitesse de la transmission à 100MB/s
- Configurer l'adresse MAC
- Configurer l'adresse de PHY
- Configurer le type de PHY a **DP83848_PHY_ADDRESS**

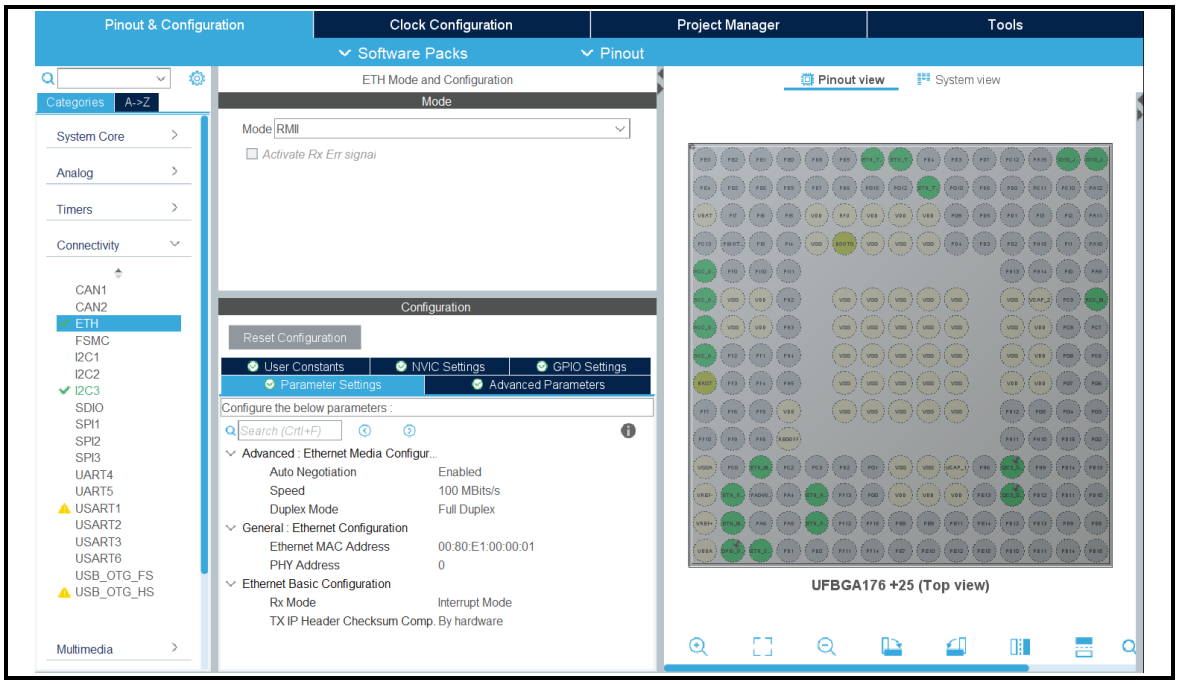

Figure 2.53 : Configuration de l'interface Ethernet

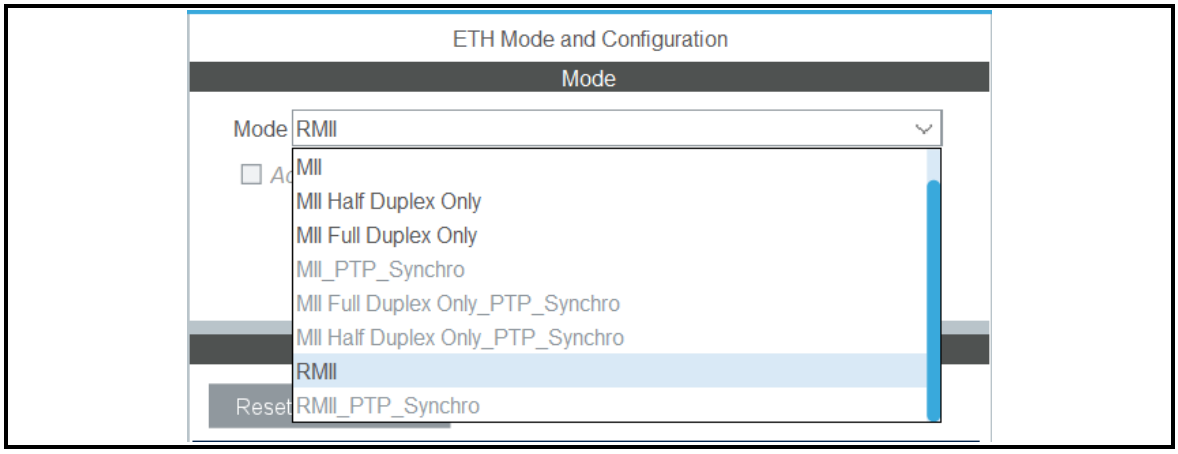

Figure 2.54 : Activer le mode RMII

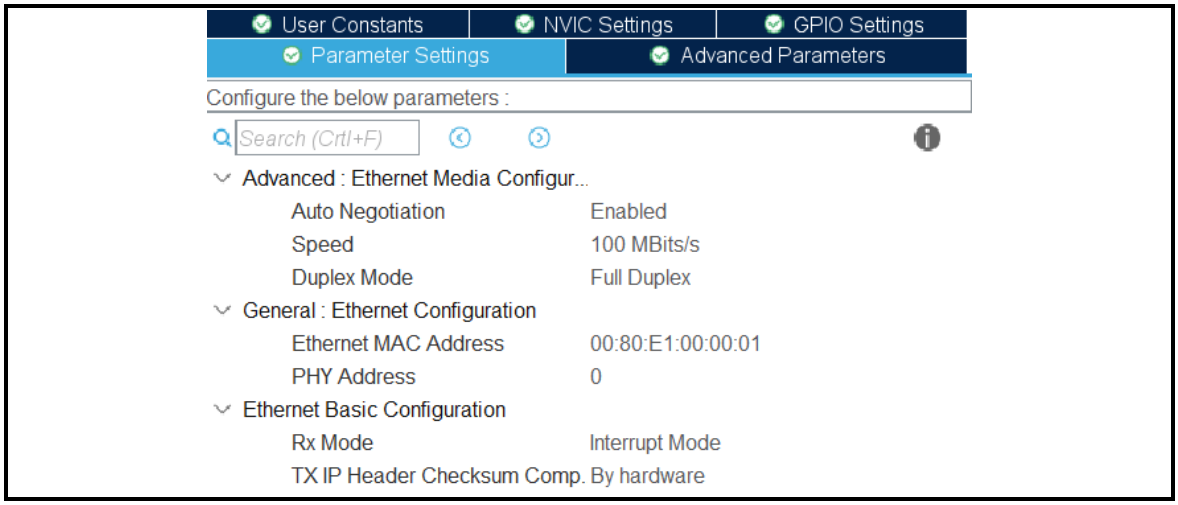

Figure 2.55 : Activer l'auto négociation et configuration de l'adresse MAC

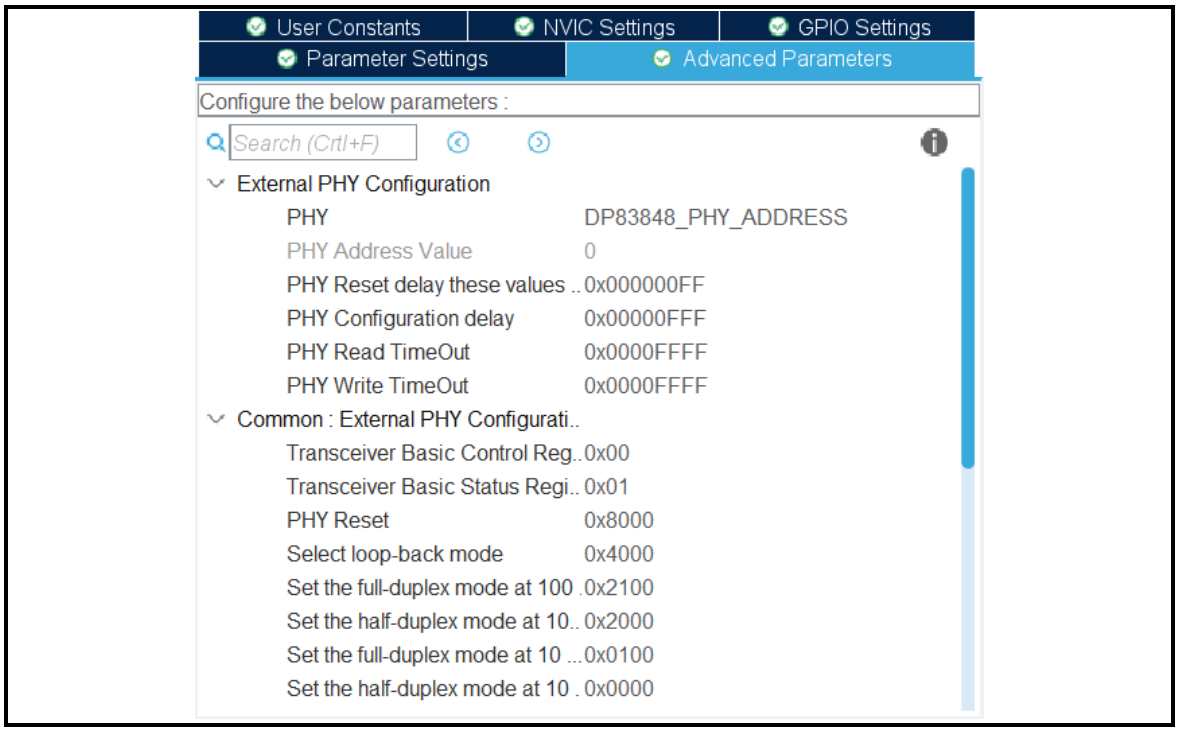

Figure 2.56 : Configuration de PHY

g) Bus I2C

Pour configurer le I2C il faut :

- Activer l'interface I2C
- Configurer le mode de vitesse à **Standard Mode**

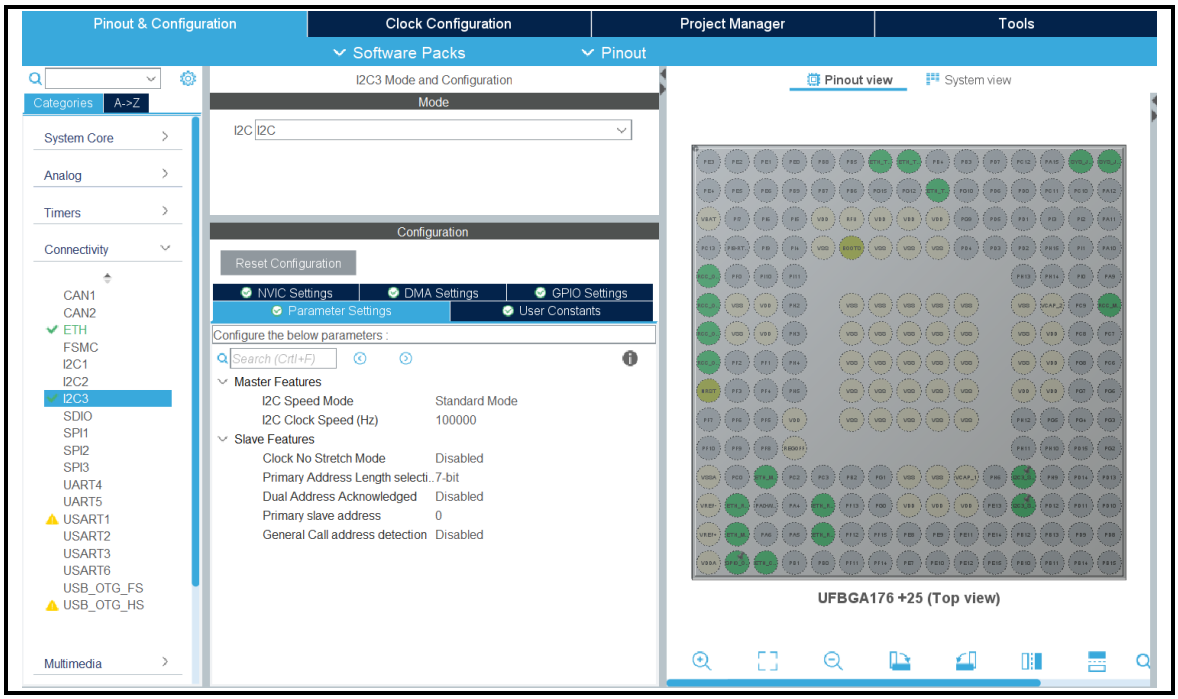

Figure 2.57 : Configuration de l'interface I2C

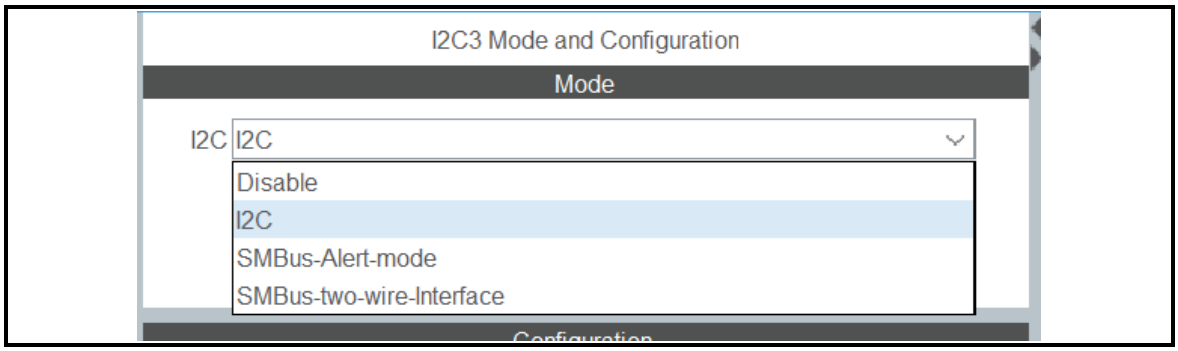

Figure 2.58 : Activer l'interface I2C

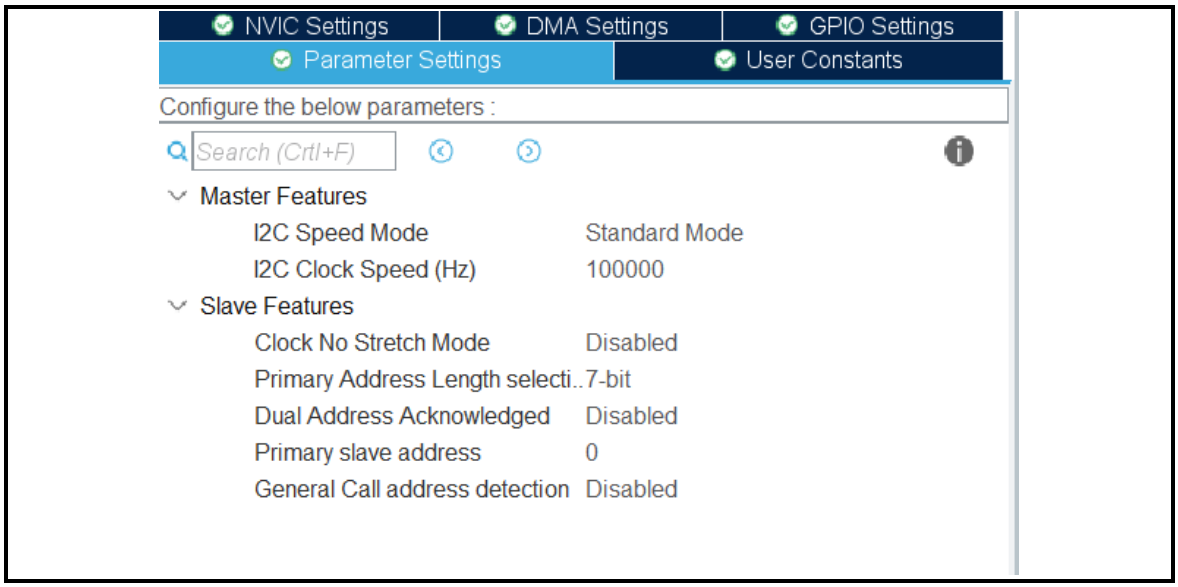

Figure 2.59 : Configurer l'horloge de I2C

2.3.2 Réalisation du programme

a) Serveur BOOTP

Bootp est un Protocol utilisé sur le protocole internet ce qui permet à une machine cliente de découvrir sa propre adresse IP et l'adresse de l'hôte serveur

Dans l'en-tête IP d'une demande de démarrage, le client remplit sa propre adresse IP source si connue, sinon zéro. Lorsque l'adresse du serveur est inconnue, l'adresse IP de destination sera « l'adresse de diffusion » 255.255.255.255. Cette adresse signifie 'diffusion sur le câble local, (Je ne connais pas mon numéro de réseau)'

L'en-tête UDP contient les numéros de port source et de destination. Le protocole BOOTP utilise deux numéros de port réservés, « client BOOTP » (68) et « serveur BOOTP » (67). Le client envoie des requêtes en utilisant le « serveur BOOTP » comme

port de destination, il s'agit généralement d'une émission. Le serveur envoie des réponses en utilisant « client BOOTP » comme port de destination [17]

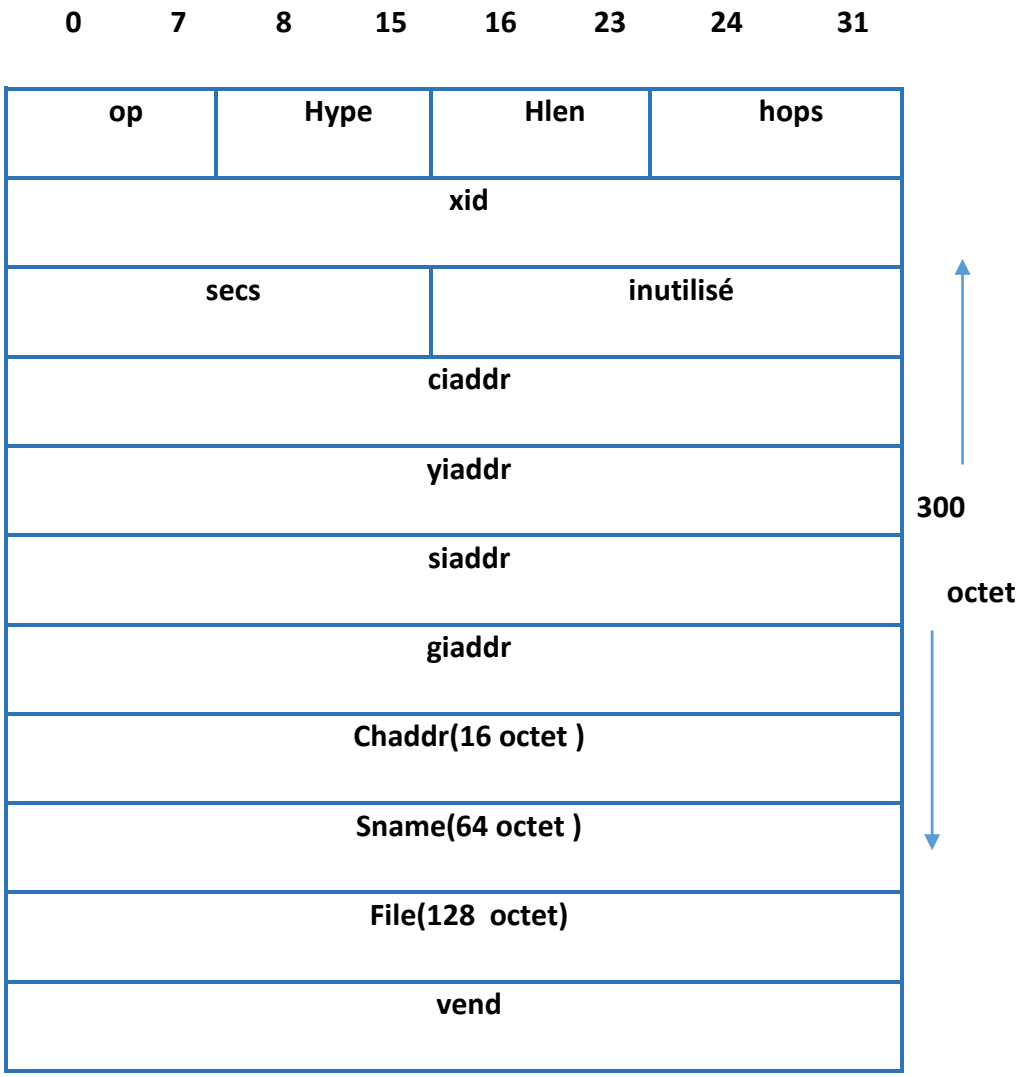

Tableau 2.8 : La trame bootp

Op : code d'opération de paquet / type de message. (1 = bootp requête, 2 = bootp réponse)

htype : type d'adresse matérielle (1 = Ethernet)

hlen : longueur de l'adresse matérielle (par exemple « 6 » pour 10 Mo d'Ethernet).

hops : le client met à zéro

xid : ID de transaction, un nombre aléatoire, utilisé pour faire correspondre cette demande de démarrage avec la réponse qu'il génère.

secs : rempli par le client, secondes écoulées depuis que le client a commencé à essayer de démarrer.

Yiaddr : « votre » adresse IP (client), rempli par le serveur si le client ne le fait pas connaître sa propre adresse (ciaddr était 0).

Siaddr : Adresse IP du serveur, renvoyé en boot réponse par le serveur.

giaddr : adresse IP de la passerelle, utilisé dans le démarrage optionnel de la passerelle croisée.

chaddr : adresse matérielle du client, rempli par le client.

sname : Nom de l'hôte server optionnel, la chaine doit être terminée par le caractère Null

file : nom du fichier de démarrage, chaîne terminée par NULL, nom 'générique' ou NULL dans boot requête, nom de chemin de répertoire complet dans boot réponse .

vend : zone optionnelle spécifique au fournisseur

Nous avons réalisé le bootp avec un programme qui envoi des requêtes bootp pour obtenir une adresse IP, **figure 2.60**

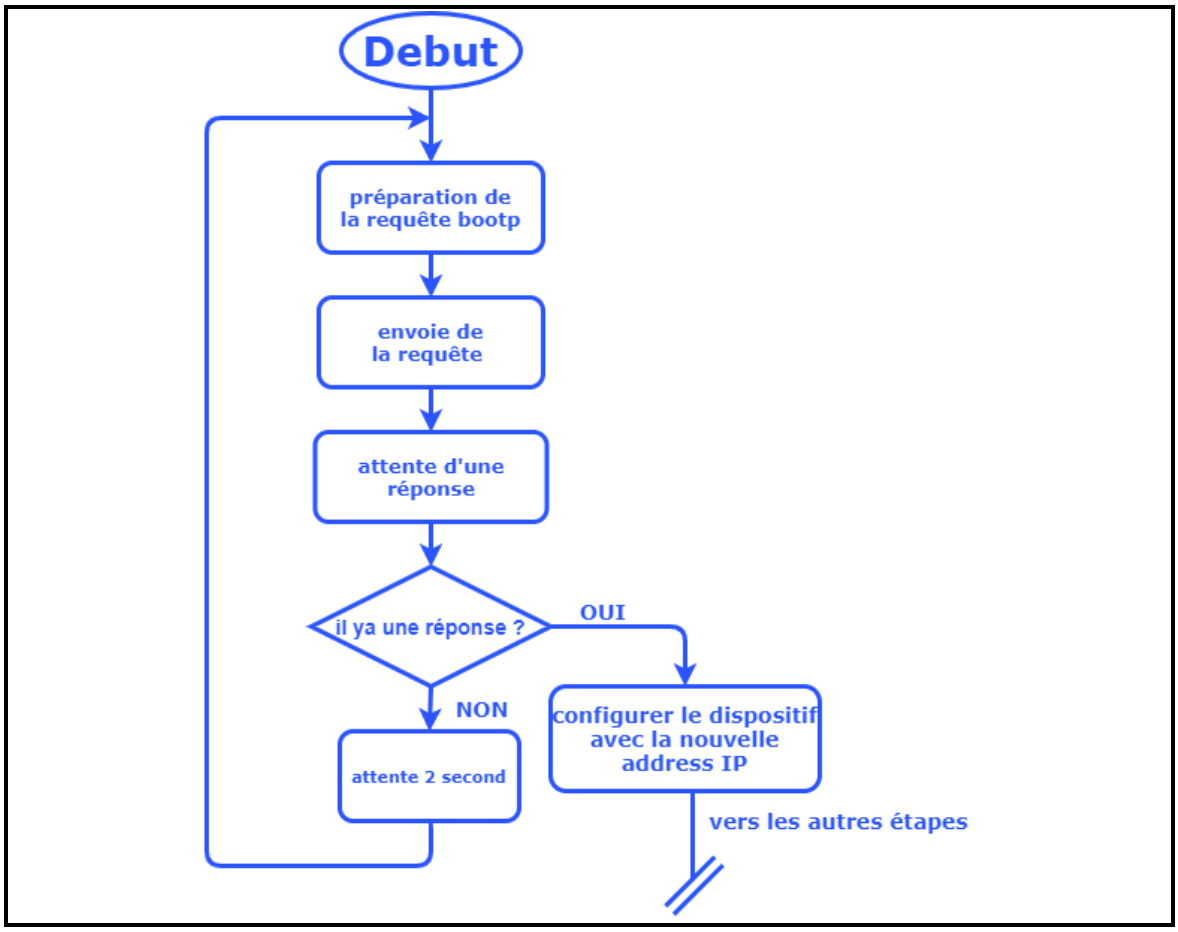

Figure 2.60 : Serveur BOOTP

Le programme commence par l'initialisation du serveur bootp par la configuration de l'adresse IP et du port, **figure 2.61** et la création des socket UDP **figure 2.62**

```
.../
struct sockaddr_in bootpaddr_serv;
struct sockaddr_in bootpaddr_clin;
bootpaddr_serv.sin_family = AF_INET ;
inet_aton("255.255.255.255", &bootpaddr_serv.sin_addr.s_addr);
bootpaddr_serv.sin_port = htons(67);
bootpaddr_clin.sin_family = AF_INET ;
bootpaddr_clin.sin_addr.s_addr = INADDR_ANY ;
bootpaddr_clin.sin_port = htons(68);
/....
```

Figure 2.61 : Configuration du serveur BOOTP

```
.../
int bootpsock_serv = socket(AF_INET,SOCK_DGRAM,0);
if(bootpsock_serv < 0)
   {
     __NOP();
   }

int bootpsock_clin = socket(AF_INET,SOCK_DGRAM,0);
if(bootpsock_clin < 0)
   {
     __NOP();
   }
/...
```

Figure 2.62 : Création des sockets du serveur BOOTP

Lier le 'socket' client avec l'adresse IP et celui déjà configuré **figure 2.63**

```
.../
bind(bootpsock_clin,(struct
sockaddr*)&bootpaddr_clin,sizeof(bootpaddr_clin));
/...
```

Figure 2.63 : Configuration du socket BOOTP

Apres l'initialisation, le serveur prépare la requête bootp **figure 2.64**

```
#include "bootp_ip_config.h"
void UdpPacketPrc(int opc)
{
      switch(opc)
      {
        case SEND_REQUST:
        {
          createReq();
        }
        break;
        case RECIVE_REQUST:
        {
          ProcReq();
        }
        break;
      }
}

void createReq()
{
      for(int i = 0 ;i<300;i++) UdpBuffer[i] = 0;

      Opcode = 1;
      HardwareType = 1;
      HardwarAddressLenth = 6;
      TransactionId =0x77;
      NuberOfSeconds = 2;
      ClientIpAddress = 0x0000;
```

```
UdpBuffer[OPCODE]  = Opcode;
      UdpBuffer[HRDW_TYPE]  = HardwareType;
      UdpBuffer[HARDW_ADDRS]  = HardwarAddressLenth;
    /*transaction ID*/
      UdpBuffer[TRANS_ID0] =TransactionId >> 24;
       UdpBuffer[TRANS_ID1] =TransactionId >> 16;
      UdpBuffer[TRANS_ID2] =TransactionId >> 8;
      UdpBuffer[TRANS_ID3] =TransactionId;
    /*number of seconds*/
      UdpBuffer[SESCS_1] = NuberOfSeconds>>8;
      UdpBuffer[SESCS_1] = NuberOfSeconds;
    /* client IP address */
      UdpBuffer[CLIENT_IP_ADDRS_0] = ClientIpAddress >> 24;
      UdpBuffer[CLIENT_IP_ADDRS_1] = ClientIpAddress >> 16;
      UdpBuffer[CLIENT_IP_ADDRS_2] = ClientIpAddress >> 8;
      UdpBuffer[CLIENT_IP_ADDRS_3] = ClientIpAddress ;
    /* MAC ADDRESS     */
      UdpBuffer[MAC_ADDR_0] = 0x00;
      UdpBuffer[MAC_ADDR_1] = 0x80;
      UdpBuffer[MAC_ADDR_2] = 0xE1;
      UdpBuffer[MAC_ADDR_3] = 0x00;
       UdpBuffer[MAC_ADDR_4] = 0x00;
       UdpBuffer[MAC_ADDR_5] = 0x00;
}
```

Figure 2.64 : Création de la requête BOOTP

Après la préparation de la requête le serveur peut l'envoyer et attendre une réponse, s'il ya une réponse le serveur va configurer le dispositif avec la nouvelle adresse IP **figure 2.65**, s'il n'y a pas de réponse, le serveur attendra deux secondes pour renvoyer une autre requête

```
.../
do{
    UdpPacketPrc(SEND_REQUST);
    if( sendto(bootpsock_serv,UdpBuffer, 300, 0, (const struct
sockaddr*) & bootpaddr_serv, sizeof(bootpaddr_serv))  > 0)
      {
         socklen_t bootplen = sizeof(bootpaddr_clin);
         datarecv = recvfrom(bootpsock_clin, UdpBuffer, 300,
MSG_DONTWAIT, (struct sockaddr*) & bootpaddr_clin,
&bootplen);//MSG_WAITALL
         __NOP();
      }
     else
      {
        __NOP();
      }
     HAL_Delay(2000);
   }
while (datarecv <300 && datarecv < 0);
/...
```

Figure 2.65 : Serveur BOOTP

b) Serveur MODBUS

Le serveur modbus **figure 2.66** commencer par l'écoute des trames reçues sur le port 502 **figure 2.67**, s'il y a une trame reçue sur ce port, le serveur va vérifier le code de fonction, s'il est correct. S'il n'est pas correct, le serveur va préparer une réponse de code exception et l'envoyer au client. Suite à cela le serveur passe à la deuxième étape pour vérifier si l'adresse de registre adressé est correct et vérifie si l'adresse appartient à la plage des adresses configurées dans le dispositif. Si cette adresse appartient à cette plage, le serveur va exécuter le code de fonction et préparer une réponse et l'envoyer au client, si ce n'est pas le cas le serveur va préparer une réponse de DATA exception et l'envoyer au client

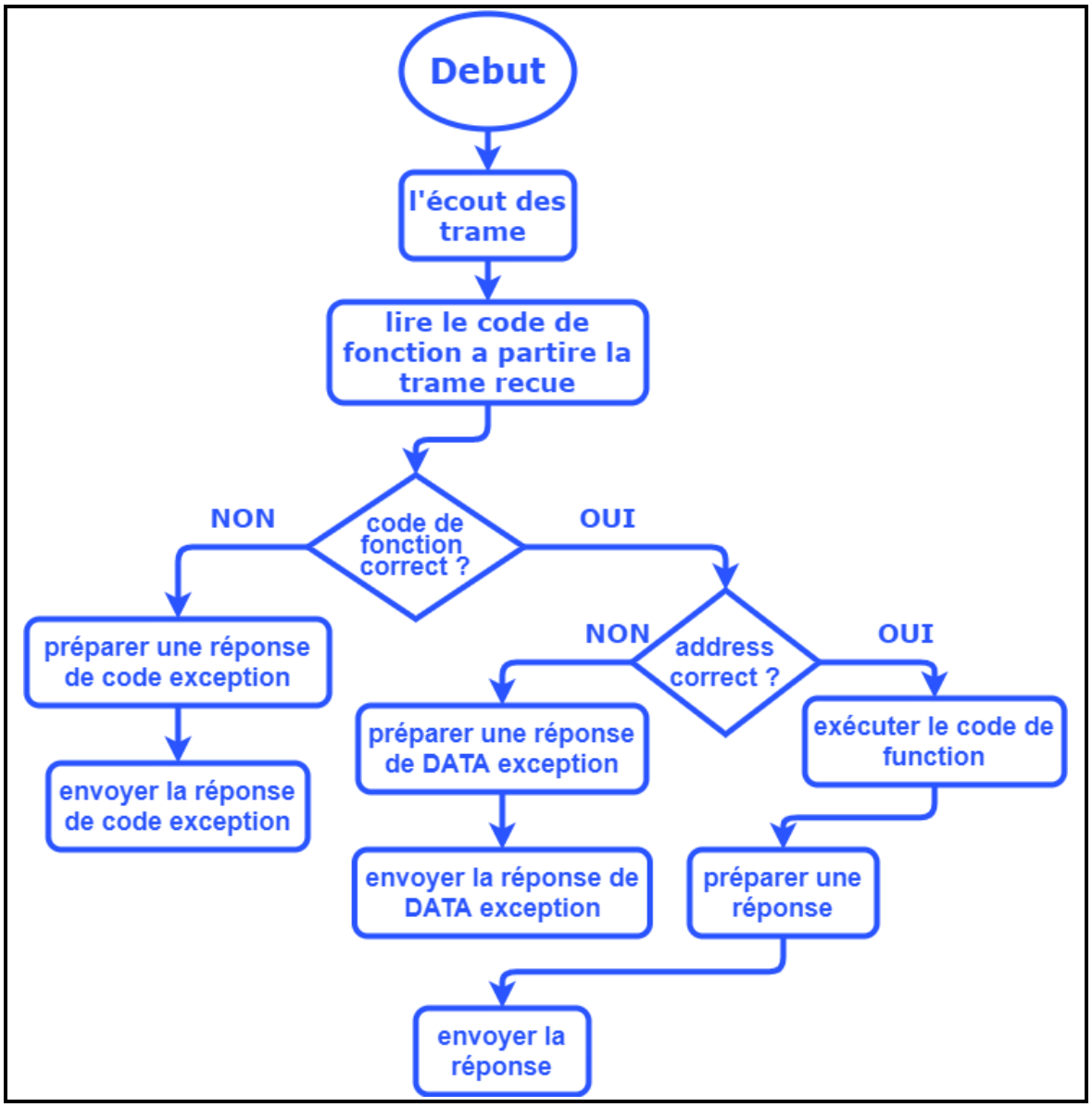

Figure 2.66 : Vu globale du serveur MODBUS

```
.../ int portsoc = ntohs(sina.sin_port);
switch (portsoc)
{
  case 502:// Modbus
  {
   int kk = recv(i, &aucTCPBuf, 260, 0);
   if (!(kk <= 0)){
       ( void )xMBPortEventPost( EV_FRAME_RECEIVED );
       usTCPBufPos = kk;
       cur_sock = i;
       return TRUE;
     }
   else
     {
       close(i);
       FD_CLR(i, &fir);
       return TRUE;
     }
   }
   break;
   case 44818://ENIP
   { ............................ }
   break;
} /...
```

Figure 2.67 : Serveur Modbus

```
.../
ucFunctionCode = ucMBFrame[MB_PDU_FUNC_OFF];
eException = MB_EX_ILLEGAL_FUNCTION;
for( i = 0; i < MB_FUNC_HANDLERS_MAX; i++ )
{
   if( xFuncHandlers[i].ucFunctionCode == 0 )
     {
        break;
     }
   else if( xFuncHandlers[i].ucFunctionCode == ucFunctionCode )
     {
        eException = xFuncHandlers[i].pxHandler( ucMBFrame,
&usLength );
        break;
     }
}
/...
```

Figure 2.68 : Traitement de les requête reçu

c) Serveur ENIP

```
.../
case 44818://ENIP
{
 int tet = recv(i, buff, 260, 0);
if (!(tet <= 0))
  {
    buff[4] = 3;
    buff[5] = 3;
    buff[6] = 2;
    buff[7] = 16;
    buff[40] = buff[40] | 0x80;
    buff[42] = 0;
    buff[43] = 0;
    bootpmod = buff[48];
    send(i, buff, tet, 0);
    cal=cal+1;
  }
 else
  {
    close(i);
    FD_CLR(i, &fir);
    cal = 0;
    bootpmod &= 0x0F;
    if (bootpmod == 2) //enable bootp
      {
         Flash_Read_Data(buff);
         buff[12] = 1;
         Flash_Write_Data(buff);
         soc = fir;
      }
    else //desable bootp
      {
         buff[0] = UdpBuffer[YOUR_IP_ADDRS_0];
         buff[1] = UdpBuffer[YOUR_IP_ADDRS_1];
        buff[2] = UdpBuffer[YOUR_IP_ADDRS_2];
         buff[3] = UdpBuffer[YOUR_IP_ADDRS_3];
         buff[12] = 1;
         buff[12] = 0;
         Flash_Write_Data(buff);
      }
   }
 }
break;
/...
```

Figure 2.69 : Serveur ENIP

d) Capteur

Pour la description du AHT10, se conformer au dadatsheet pour comprendre son fonctionnement [21].

Pour l'initialisation du capteur, le programme commence, par les étapes suivantes : **figure 2.70**

- Attendre 20 ms pour que le capteur soit en état de repos
- Envoyer la commande AHT10_NORMAL_CMD pour installer le capteur en mode normal
- Attendre 350 ms
- Envoyer la commande INIT_CMD pour initialiser le capteur
- Activer la calibration et charger les coefficients de fabricateur par l'envoie de la commande AHT10_INIT_CAL_ENABLE
- Attendre 350 ms

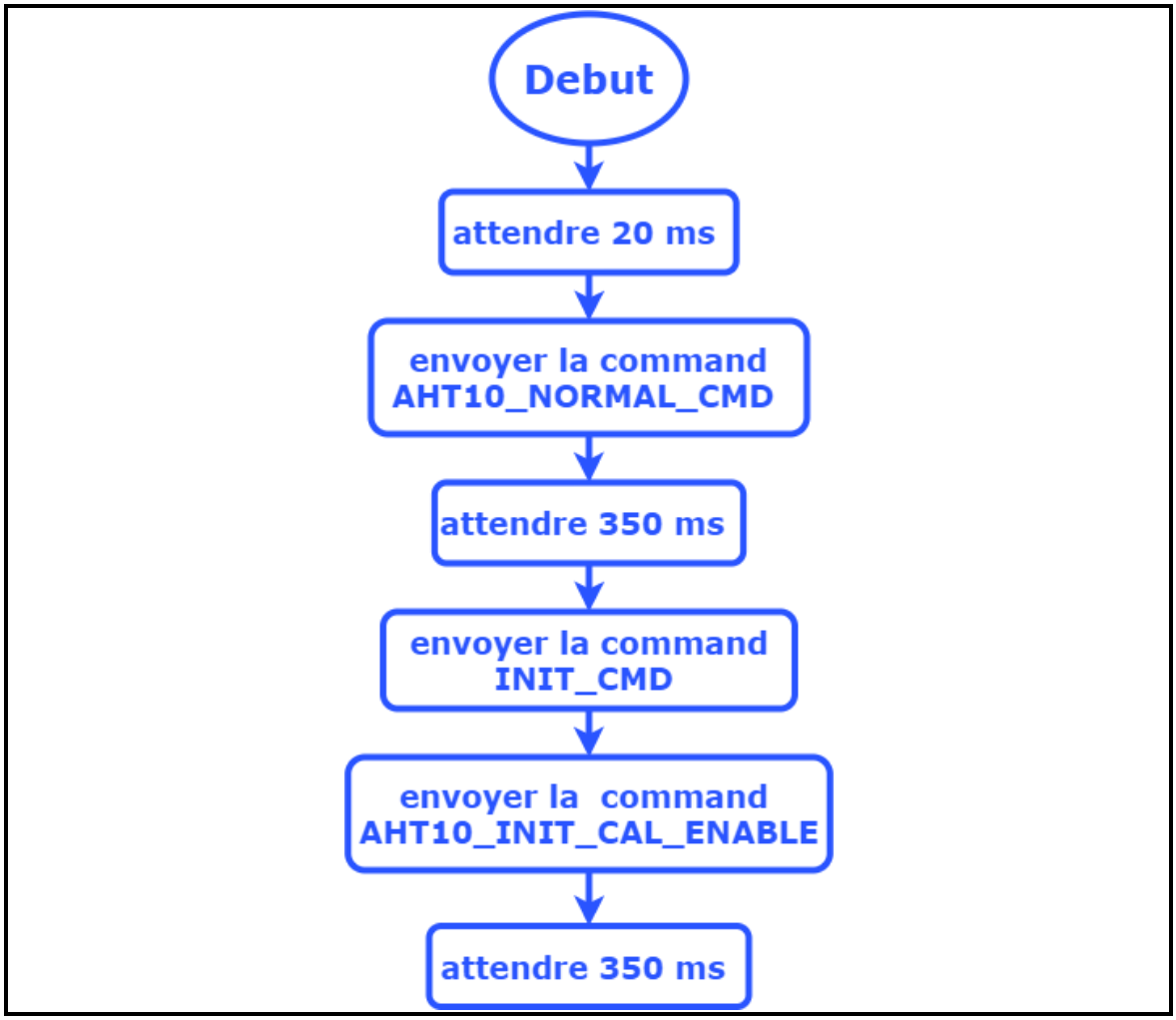

Figure 2.70 : Initialisation du capteur

Apres l'initialisation du capteur, il faut déclencher la mesure et lire les données du capteur **figure 2.71**:

- Envoyer la commande TRIG_MISU_CMD pour déclencher la mesure
- Attendre 80 ms

- Lire les 6 octets

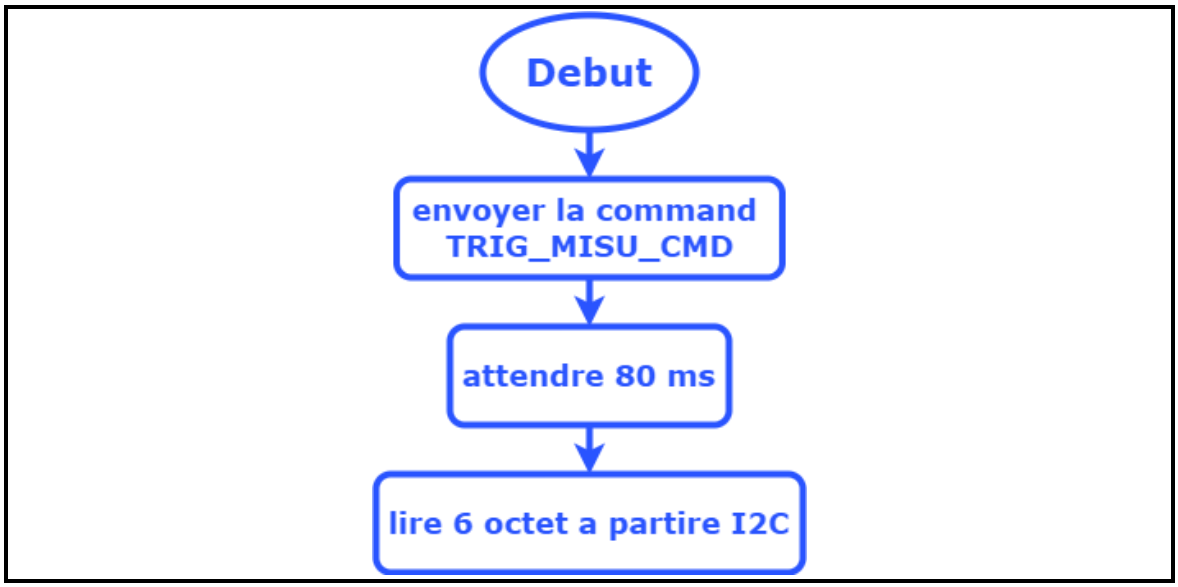

Figure 2.71 : Lire les données du capteur

Pour ensuite interpréter les données qui sont déjà lues à partir du capteur

Le programme pour initialiser le capteur **figure 2.72**

```
.../
int AHT10_int(I2C_HandleTypeDef *i2cHand)
{
      HAL_Delay(20);
      uint8_t bufg[4];

      bufg[0] = AHT10_NORMAL_CMD;
      bufg[1] = AHT10_DATA_NOP;
      bufg[2] = AHT10_DATA_NOP;
      HAL_I2C_Master_Transmit(i2cHand, AHT10_ADDR, bufg, 3,
HAL_MAX_DELAY);
    HAL_Delay(350);

    bufg[0] = INIT_CMD;
      bufg[1] = AHT10_INIT_CAL_ENABLE;
      bufg[2] = AHT10_DATA_NOP;
      HAL_I2C_Master_Transmit(i2cHand, AHT10_ADDR, bufg, 3,
HAL_MAX_DELAY);
      HAL_Delay(350);
return 0;
}
/...
```

Figure 2.72 : Programme de l'initialisation de capteur

Le programme pour Déclencher la mesure **figure 2.73**

```
.../
int ReadData(I2C_HandleTypeDef *i2cHand)
{
      uint8_t bufg[3];
    bufg[0] = TRIG_MISU_CMD;
    bufg[1] = TRIG_MISU_DATA0;
    bufg[2] = TRIG_MISU_DATA1;

HAL_I2C_Master_Transmit(i2cHand,AHT10_ADDR,bufg,3,HAL_MAX_DELAY);

HAL_Delay(80);

int stat = HAL_I2C_Master_Receive(i2cHand, AHT10_ADDR, DATA_buf,
6, HAL_MAX_DELAY);
  if( stat != HAL_OK)
    {
            __NOP();
    }
return 0;
}
/...
```

Figure 2.73 : Déclenchement de la mesure

Le programme pour lire et interpréter la valeur de la température **figure 2.74**

```
.../
int ReadTemp(I2C_HandleTypeDef *i2cHand0)
{
      ReadData(i2cHand0);
      uint8_t a0  = DATA_buf[3];
      uint8_t a1  = DATA_buf[4];
      uint8_t a2  = DATA_buf[5];

      uint32_t tmp = ((uint32_t)(DATA_buf[3] & 0x0F) << 16) |
((uint16_t)DATA_buf[4] << 8) | DATA_buf[5];
      tmp;
return ((int)tmp * 0.000191 - 50);
}
/...
```

Figure 2.74 : Lire la température

Le programme pour lire et interpréter la valeur de l'humidité **figure 2.75**

```
/...
int ReadHum(I2C_HandleTypeDef *i2cHand0)
{     ReadData(i2cHand0);
      uint32_t hum = (((uint32_t)DATA_buf[1] << 16) |
((uint16_t)DATA_buf[2] << 8) | (DATA_buf[3])) >> 4;
      hum = hum * 0.000095;
 return (int)hum;
}
/...
```

Figure 2.75 : Lire l'humidité

e) Basculement des ports

On a réalisé dans ce projet un dispositif à deux ports Ethernet pour la réalisation d'un réseau modbus TCP/IP en anneau **figure 2.76**, l'utilisation du microcontrôleur STM32F207IGH6 possède une seul interface MAC qui ne peut pas faire communiquer simultanément les deux ports intégrés sur le PHY (DP83849i), le DP83849i est équipé par une option qui est le **port switching**, c'est une option qui permet de basculer entre les ports , par exemple on peut connecter le **MAC A** avec le **PORT A** mais on peut le basculer avec le **PORT B** , la même chose pour le **MAC B**, on peut aussi connecter le **PORT A** directement avec le **PORT B**

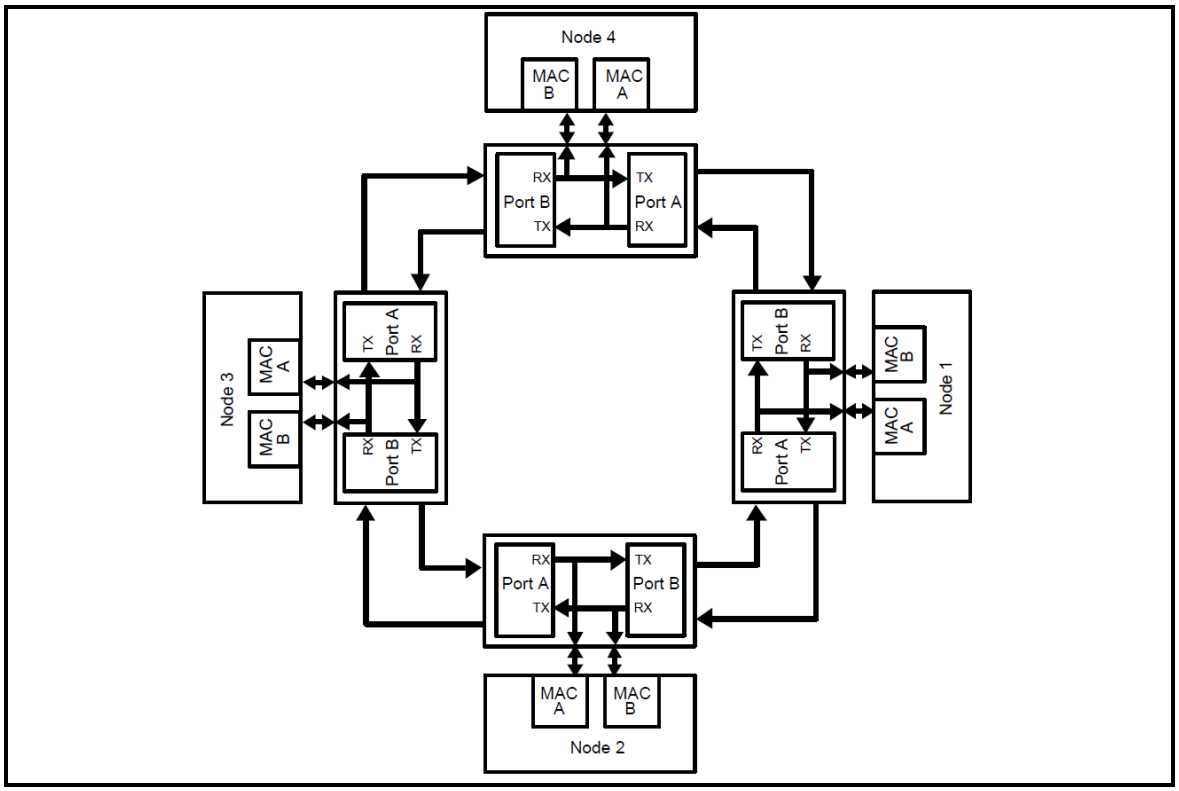

Figure 2.76 : Réseau en anneau

Dans ce projet nous avons mis des règles pour réaliser le réseau en anneau, le **RX** du microcontrôleur STM32 est toujours connecté avec le **PORT A** pour écouter des trames envoyées à partir de dispositif connecté sur le **PORT A**, le **TX** du microcontrôleur est connecté avec le **PORT A** pour envoyer une réponse seulement s'il y a une trame (requête modbus) reçue à partir du **PORT A** avec les adresses IP et MAC du serveur configuré sur le microcontrôleur, s'il n'y a aucune réception de trame le **TX** du

microcontrôleur est déconnecté du **PORT A** ce qui entraine la communication entre le **PORT B** et le **PORT A** ,le basculement des ports est réalisé par un registre interne (BRB)

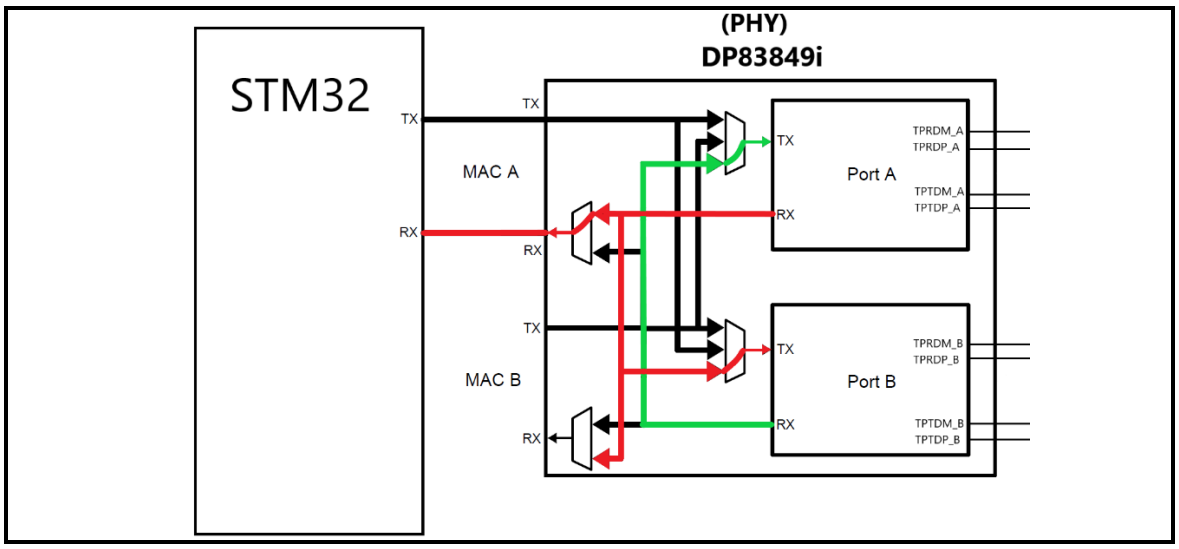

Figure 2.77 : Branchement des ports s'il Il n'y a pas une requête reçue

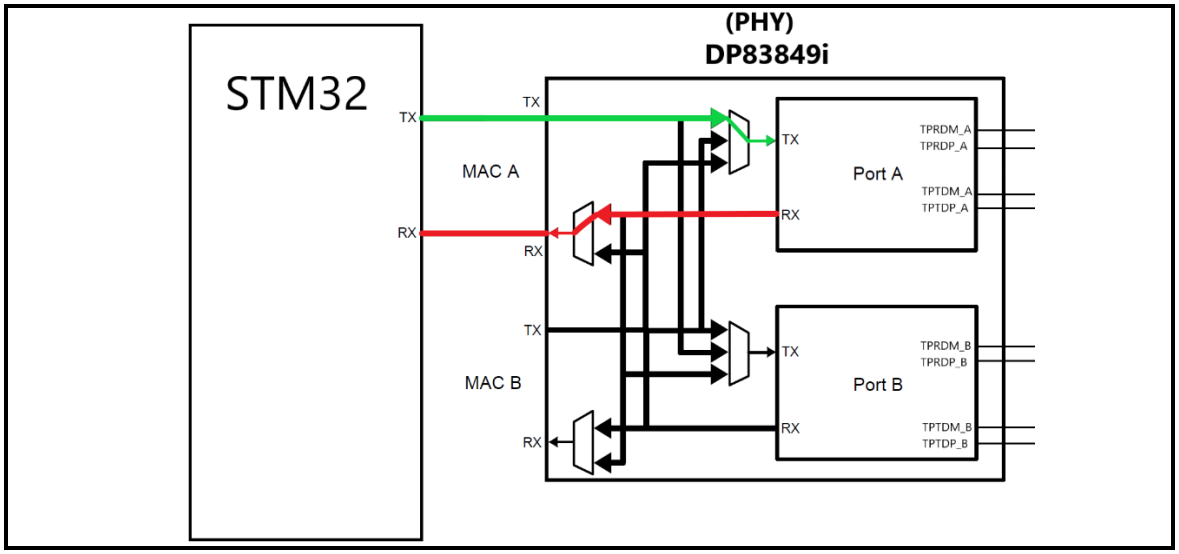

Figure 2.78 : Branchement des ports s'il y a une requête reçue

Les commandes et les configurations de mappage RX sont présentées dans les **tableaux 2.9** et **2.10**

RBR[12 :11]	la Destination de MAC RX vers les ports désirer
00	normal
01	Contraire
10	Les deux
11	Désactivé

Tableau 2.9 : Contrôles de mappage RX

PORT A BRB[12 :11]	PORT B BRB[12 :11]	Source de RX MAC A	Source de RX MAC B
0	0	PORT A	PORT B
0	1	PORT A	PORT B
0	10	PORTA	PORT B
0	11	PORT A	Désactivé
1	0	PORT A	PORT B
1	1	PORT B	PORT A
1	10	PORT B	PORT A
1	11	Désactivé	PORT A
10	0	PORT A	PORT B
10	1	PORT B	PORT A
10	10	PORT A	PORT B
10	11	PORT A	PORT A
11	0	Désactivé	PORT B
11	1	PORT B	Désactivé
11	10	PORT B	PORT B
11	11	Désactivé	Désactivé

Tableau 2.10 : Configurations de mappage du port RX

Les commandes et les configurations de mappage RX sont présentées dans les **tableaux 2.11** et **2.12**

RBR[10 :9]	La source des donne du port TX
00	normal
01	Contraire
10	RX opposée
11	Désactivé

Tableau 2.11 : Contrôles de mappage TX

PORT A BRB[10 :9]	La source des donne du port TX du PORT A	PORT B BRB[10 :9]	La source des donne du port TX du PORT B
00	MAC A	00	MAC B
01	MAC B	01	MAC A
10	RX PORT B	10	RX PORT A
11	Désactivé	11	Désactivé

Tableau 2.12 : Configurations de mappage de port TX

Le programme pour réaliser le basculement :

```
/...
  heth.Init.PhyAddress = 1;
    uint32_t val1=0;

  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val1);
  val1 &= 0xE1DF;
  val1 |= 0x0420;
  HAL_ETH_WritePHYRegister(&heth, 0x17, val1);

  heth.Init.PhyAddress = 0;
  uint32_t val=0;
  uint32_t val2=0;
  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val);

  val &= 0xE1FF;
  val |= 0x0400;
  HAL_ETH_WritePHYRegister(&heth, 0x17, val);
  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val2);
.../
```

Figure 2.79 : Configuration des ports (PORTA &PORTB)

```
/...
  heth.Init.PhyAddress = 1;
    uint32_t val1=0;
  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val1);
  val1 &= 0xE1DF;
  val1 |= 0x0420;
  HAL_ETH_WritePHYRegister(&heth, 0x17, val1);
  heth.Init.PhyAddress = 0;
  uint32_t val=0;
  uint32_t val2=0;
  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val);
  val &= 0xE1FF;
  val |= 0x0400;
  HAL_ETH_WritePHYRegister(&heth, 0x17, val);
  HAL_ETH_ReadPHYRegister(&heth, 0x17,&val2);
.../
```

Figure 2.80 : Désactivation de TX de l'interface MAC

```
/...
 heth->Init.PhyAddress =1;
 val = 0;
 HAL_ETH_ReadPHYRegister(heth, 0x17,&val);
 val &= 0xE1FF;
 val |= 0x0400;
 HAL_ETH_WritePHYRegister(heth, 0x17, val);
 heth->Init.PhyAddress =0;
 val = 0;
 HAL_ETH_ReadPHYRegister(heth, 0x17,&val);
 val &= 0xE1FF;
 val |= 0x0400;
 HAL_ETH_WritePHYRegister(heth, 0x17, val);
.../
```

Figure 2.81 : Activation de TX de l'interface MAC

2.4 Conclusion

Dans ce chapitre nous avons abordé l'implémentation software et hardware, ainsi au choix de chaque composant et la réalisation du circuit électronique faisant partie d'un i/O device qui sera apte à communiquer avec un PLC à l'aide du protocole Modbus TCP/IP. Ensuite la création et le fonctionnement de trois serveurs Modbus, Bootp et ENIP. Enfin la réalisation d'un réseau en anneau par le basculement des ports intégrés sur le PHY.

Chapitre 3 Conception et réalisation

3.1 Introduction

Chaque dispositif moderne possède des composants électroniques, ces composants se rencontrent sur une carte PCB (printed circuit board) qui s'appelle la carte mère.

Dans ce chapitre nous expliquons la conception du circuit imprimé et la réalisation du PCB tout en nous conformant aux normes de l'industrie

3.2 La conception du PCB

Dans la conception du PCB, nous avons suivi les conseils de l'ingénieur Rick Hartley (Expert en intégrité du signal et EMI), il se concentre tout particulièrement sur la conception correcte de circuits imprimés et de cartes dans le but d'éliminer ou d'éviter les interférences électromagnétiques [19].

3.2.1 Les interférences électromagnétiques

Les interférences électromagnétiques sont abrégées par l'acronyme EMI. Les EMI consistent en la perturbation involontaire générée par une source externe qui affecte le circuit électrique par induction électromagnétique, couplage électrostatique ou conduction. Il s'agit d'un problème particulier avec les équipements sensibles où les signaux de transmission peuvent être corrompus ou déformés. La transmission de données peut également entraîner une augmentation du taux d'erreur ou une perte totale de données.

Pour éviter les EMI nous devons savoir que l'énergie électrique n'est pas due la tension ou au courant mais plutôt aux champs électrique et magnétique donc le

transport d'énergie n'est pas du tout dans le cuivre mais dans l'espace entre les pistes et les plans.

a) Ligne de transmission

La ligne de transmission est n'importe quelle paire de lignes de transmission ou de guide d'ondes, les deux sont des paires de conducteurs utilisés pour déplacer l'énergie du point A au point B. Il y a une tension à travers le cuivre de la ligne de transmission et du courant la traversant, l'énergie quant à elle, est dans les champs voyageant dans le diélectrique [19].

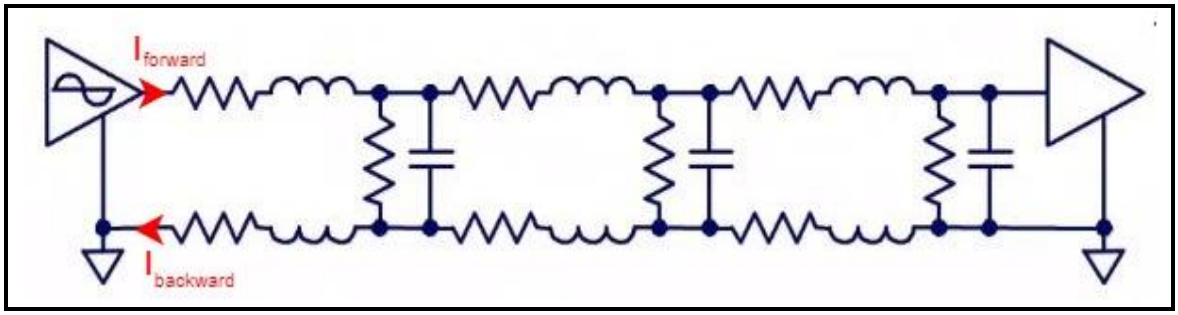

Figure 3.1 : Formation du courant dans le circuit [19]

Dans la **figure 3.1**, l'énergie lancée dans le circuit est sous la forme d'une onde constituée de champ, les champs créent une tension entre les deux éléments en cuivre, en même temps ils créent un courant direct et un courant de retour dans les deux éléments en cuivre simultanément, de sorte que le courant de retour se forme exactement au même moment que le courant direct dans la ligne de transmission, ce n'est pas un aller-retour mais un événement simultané.

Cette ligne est une ligne de transmission à trace centrée **figure 3.2**.

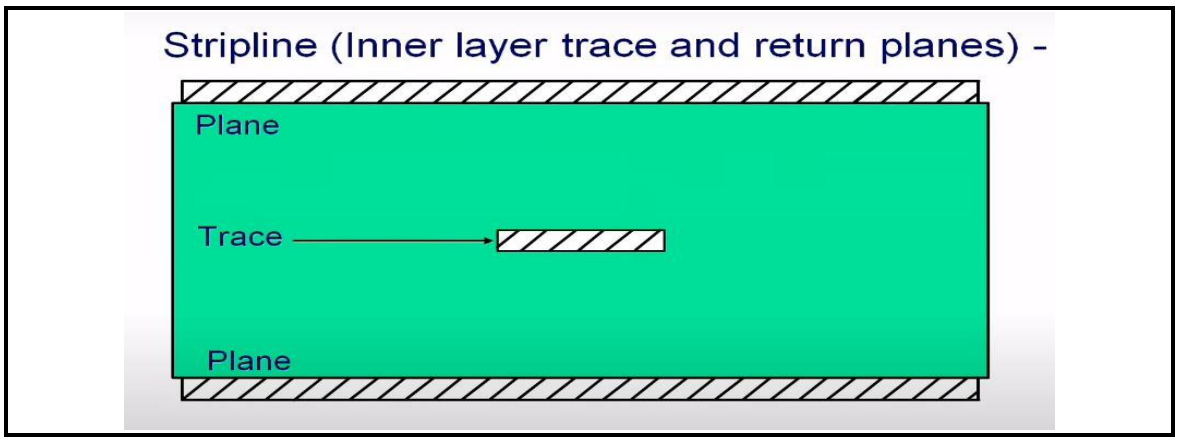

Figure 3.2 : Trace de la couche intérieure et les plans de retour [19]

Pour cette Ligne de transmission centrée entre la paire de plans, qui peuvent être des plans de masse ou des plans d'alimentation et de masse, lorsque nous lançons l'énergie dans cette ligne de transmission la première chose qui apparaît est le champ électrique **figure 3.3**

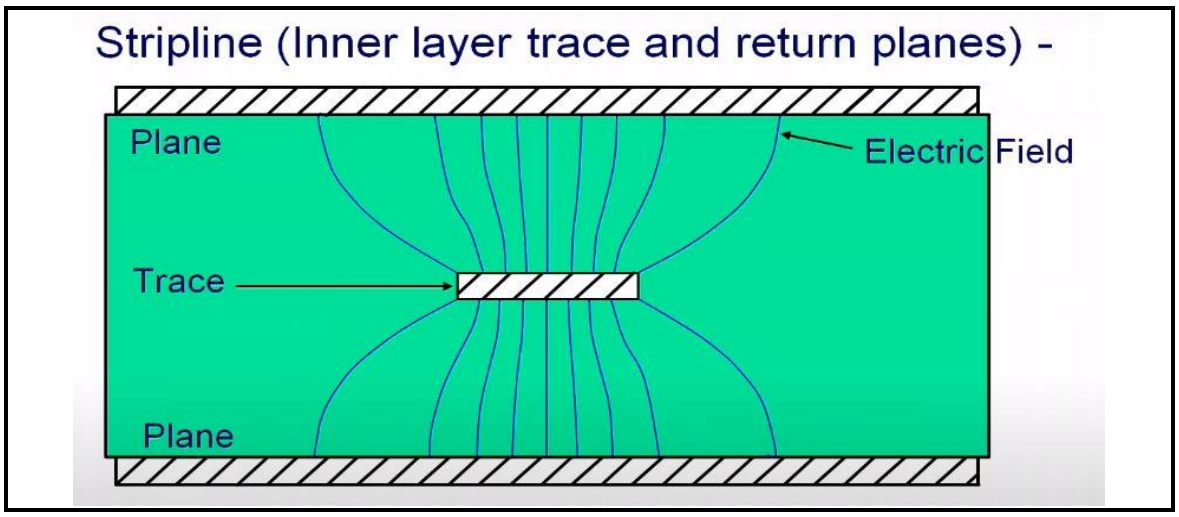

Figure 3.3 : Champ électrique dans le PCB [19]

Cette énergie excite les électrons provoquant un flux de courant dans le cuivre des plans et dans le cuivre de la trace **figure 3.4**

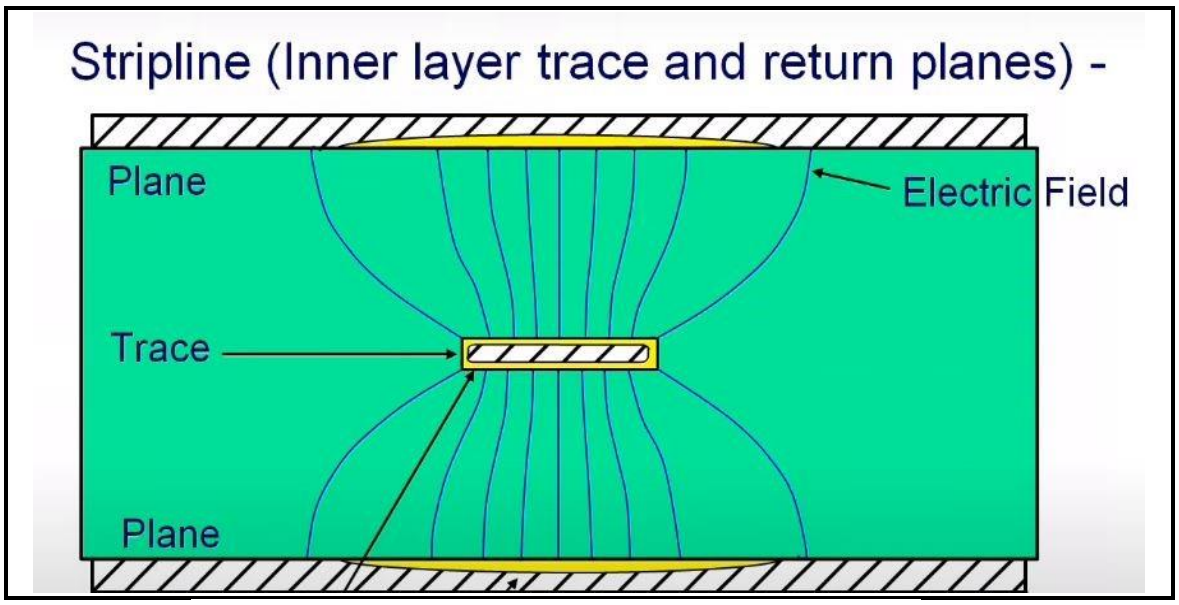

Figure 3.4 : Courants induits à travers un champ électrique

La réaction à cela est de former un champ magnétique qui entoure la trace **figure 3.5**

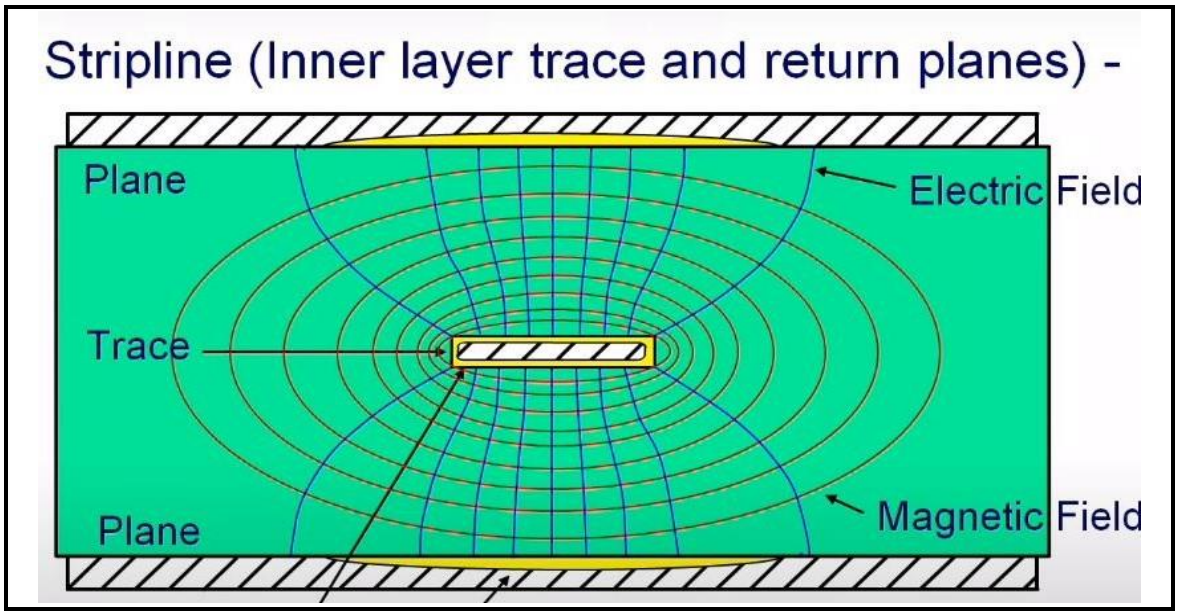

Figure 3.5 : Champs électrique et magnétique dans le PCB [19]

Remarquant que toute l'énergie est dans cet espace entre la piste et les plans, il y a une diffusion d'énergie et cette diffusion cause la diaphonie ainsi que d'autres formes d'interférence.

Dans la **figure 3.6,** les champs d'une trace de transmission de couche externe sont visualisés

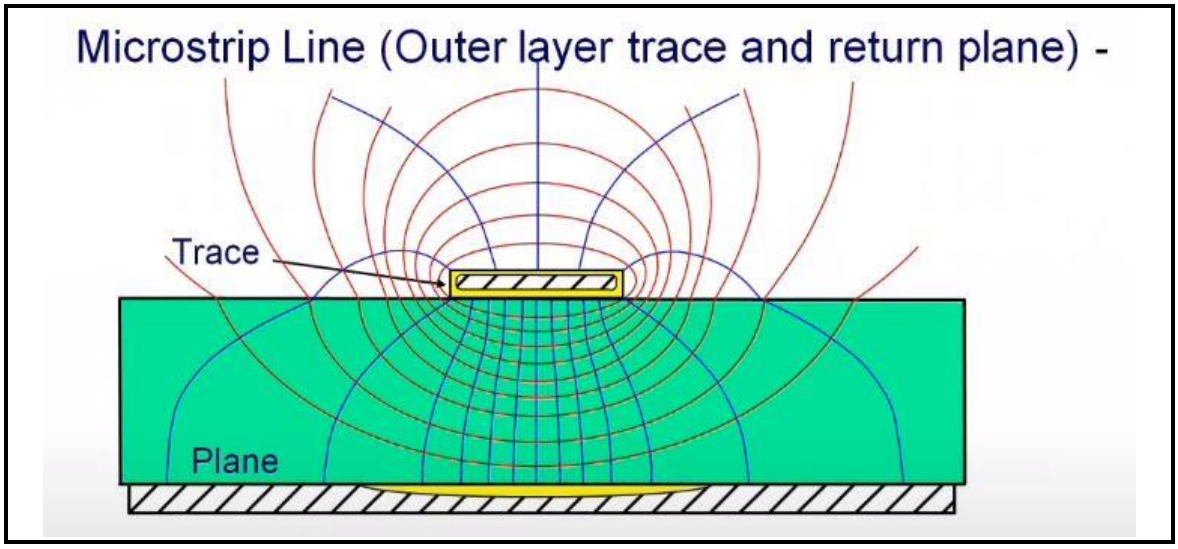

Figure 3.6 : Champs électrique et magnétique dans le PCB [19]

Pour Une trace routée au-dessus d'un plan de masse et une deuxième couche d'une carte, les champs s'étendent plus loin que les champs dans une couche interne et cette expansion des champs provoque certains rayonnements de la surface extérieure de la carte et cela est un problème dans un certain cas.

b) Effet du ground

Le ground joue un grand rôle dans la conception des circuits imprimés, une expérience réalisée par le professeur Wu de l'université nationale de Taiwan permet de savoir lequel de ces quatre circuits imprimés suivants a le moins d'EMI.

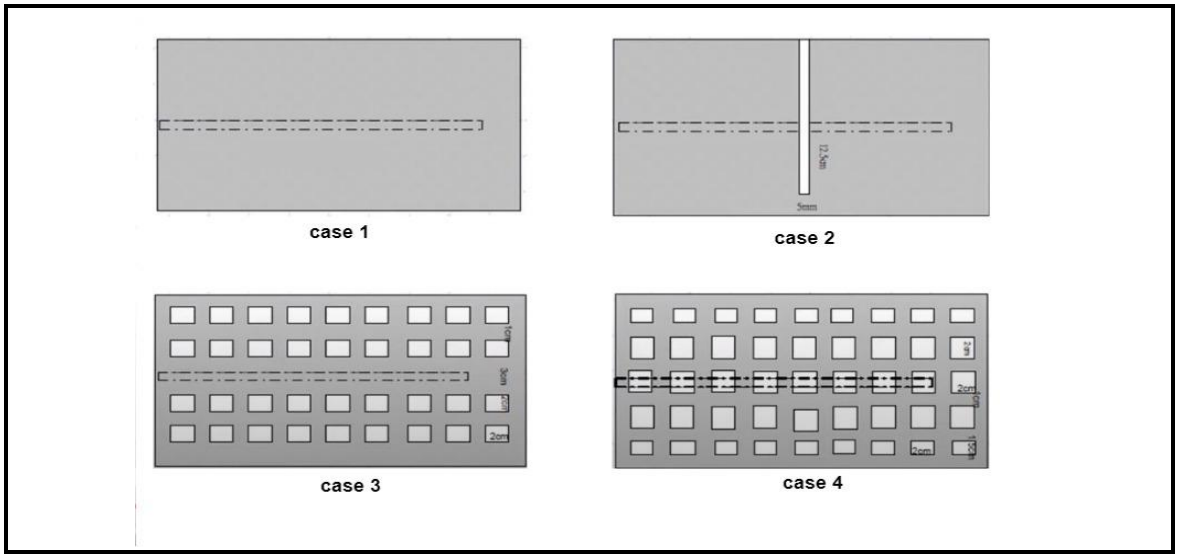

Figure 3.7 : Circuits imprimés avec différents ground

La **figure 3.7** présente quatre circuits imprimés, la grande zone grise est un plan de masse sur la deuxième couche de la carte et la ligne hachurée est une piste de transmission, à des fréquences plus élevées au-dessus de l'audio le courant de retour prend le chemin directement sous la trace.

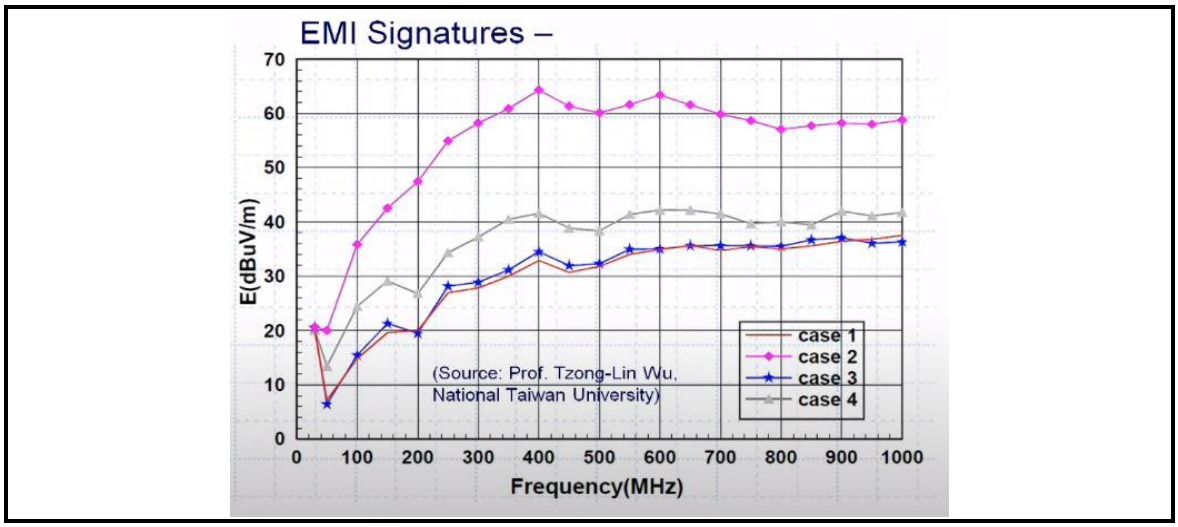

Figure 3.8 : EMI mesurée d1ns les quatre circuits imprimés

La **figure 3.8** illustre la sortie EMI mesurée par une antenne dans les direction horizontal et verticales, voir case 2 en fonction de la fréquence est de 20 à 30 dB que case 1 et case 3

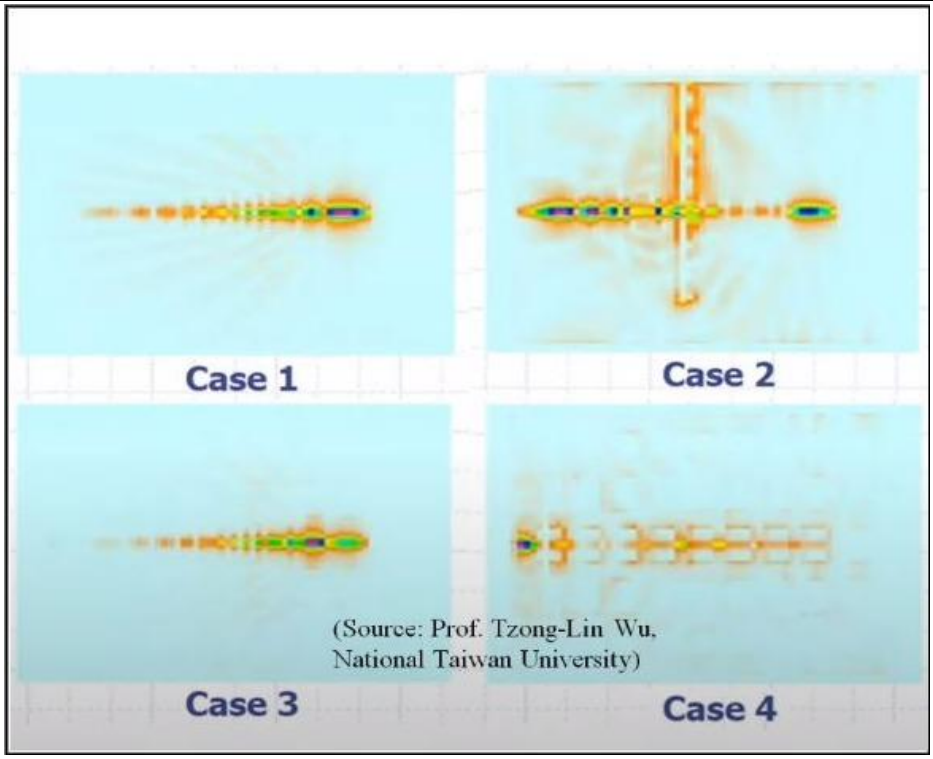

Figure 3.9 : La dissipation d'énergie dans chaque circuit

La **figure 3.9** illustre les modèles actuels développés dans les plans du circuit, la case 2 est la pire situation donc l'auteur a déduit une règle " ne jamais traverser les divisions dans les plans de sol".

c) Ground via

La bonne façon de passer d'une couche à l'autre est d'utiliser une masse via un couplage entre les couches qui est relativement proche a via de signal.

Le Dr. Howard Johnson a fait une étude à ce sujet et il a déterminé qu'il y a beaucoup d'avantage du point de vue EMI [19], ceci est illustré sur la **figure 3.10**.

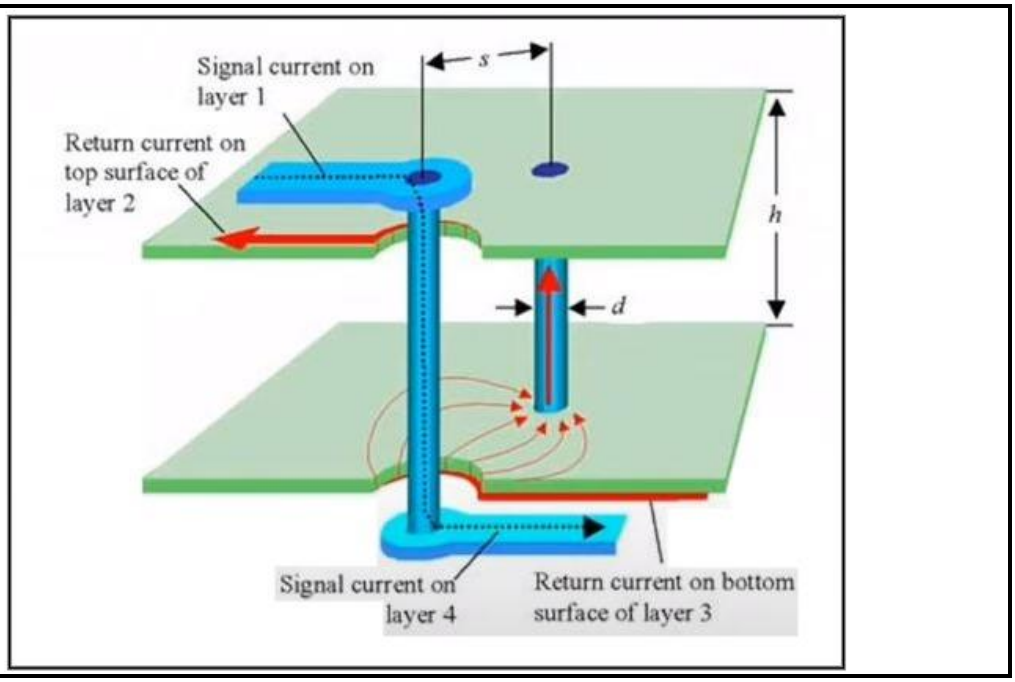

Figure 3.10 : Ground via [19]

3.2.2 La conception du I/O Device

Nous avons choisi de concevoir notre I/O Device avec un circuit imprimé de quatre couches pour plusieurs raisons notamment :

- Réduire l'EMI pour assurer le bon fonctionnement.
- Faciliter le routage.

D'après les conseils de l'ingénieur Rick Hartley à propos des PCB à quatre couches, la meilleure combinaison d'ordre des couches est la suivante :

Couche 1: GROUND

Couche 2: SIGNAL / POWER

Couche 3: SIGNAL / POWER

Couche 4 : GROUND

Selon Rick Hartley, cette combinaison est la meilleure pour réduire l'EMI et de notre côté il empêche la rétro-ingénierie qui rend l'imitation presque impossible.

La **figure 3.11** illustre l'ordre des couches dans le logiciel

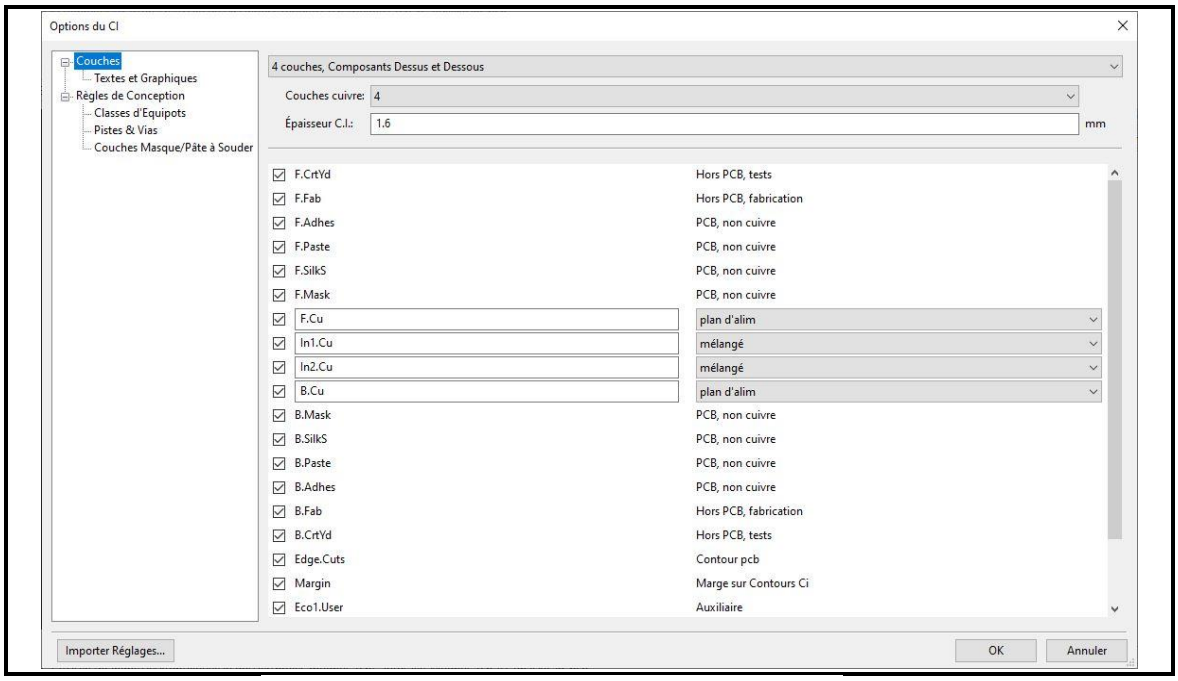

Figure 3.11 : Combinaison d'ordre des couches.

La chose importante avant de commencer à concevoir le PCB est de connaître les capacités du constructeur et ses possibilités de fabrication, la fabrication de notre circuit est faite par la société JLCPCB [18].

a) Les couches de cuivre

Notre I/O Device est construit d'un PCB de quatre couches :

- **La couche F.Cu**

Est la couche de cuivre dessus (1er couche : plan de masse) les **figures 3.12**, et **3.13** illustrent cette couche

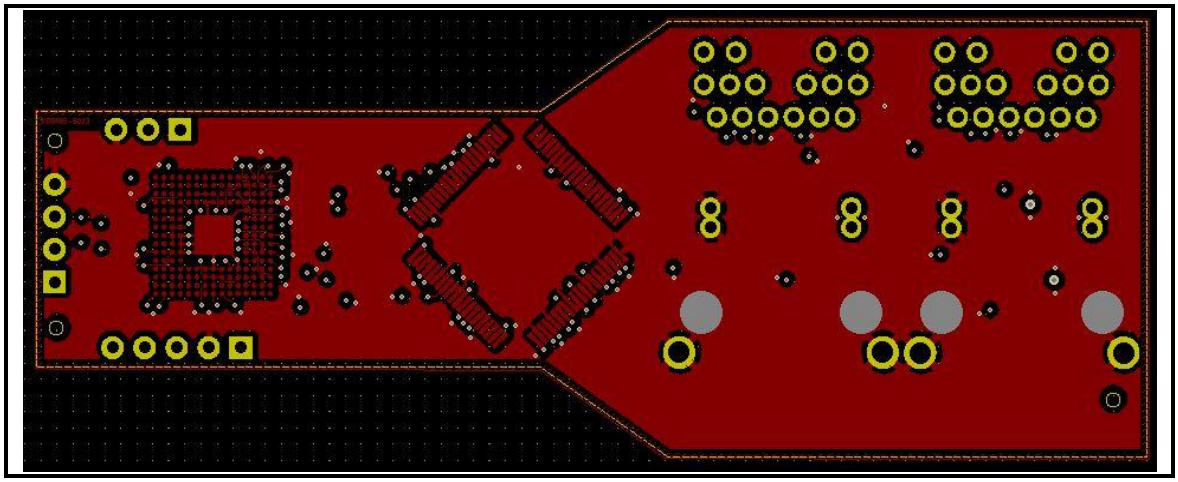

Figure 3.12 : Couche de cuivre dessus.

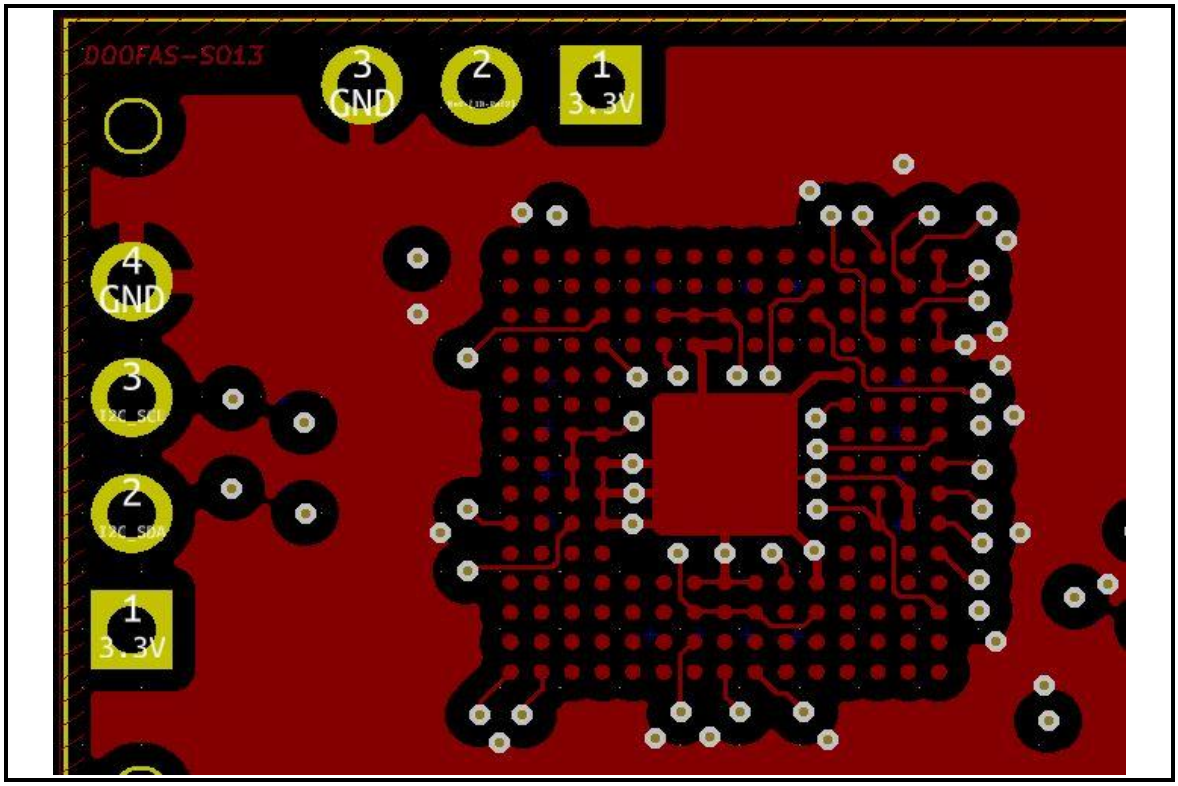

Figure 3.13 : Figure approximative de la zone du MCU.

- **La couche In1.Cu**

Est la première couche de cuivre interne (2eme couche : signal / 3.3V), La **figure 3.14** illustre cette couche

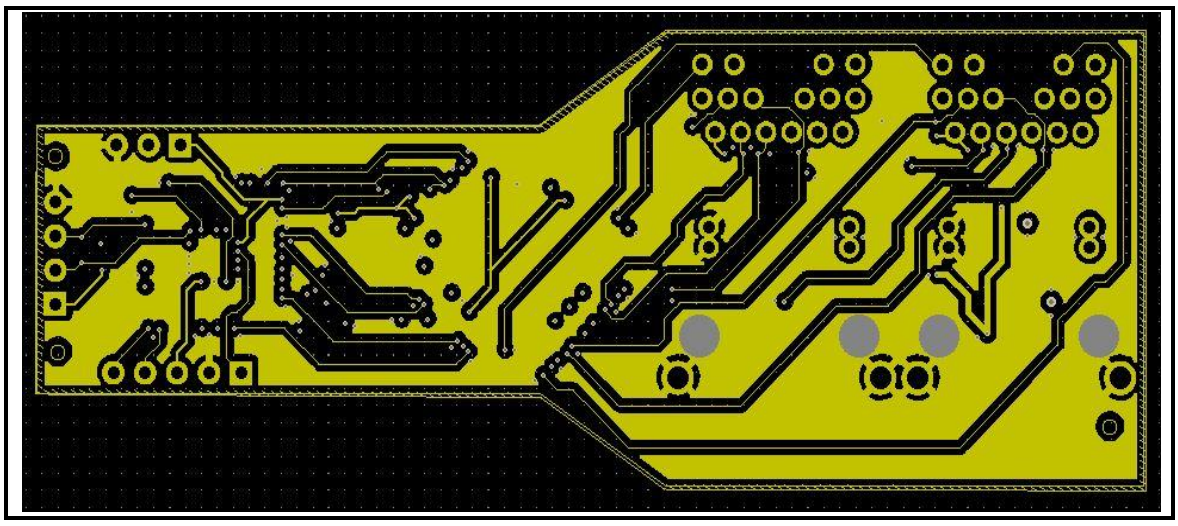

Figure 3.14 : 1er couche interne

- **La couche In2.Cu**

Est la deuxième couche de cuivre interne (3eme couche : signal / 3.3V), La **figure 3.15** illustre cette couche

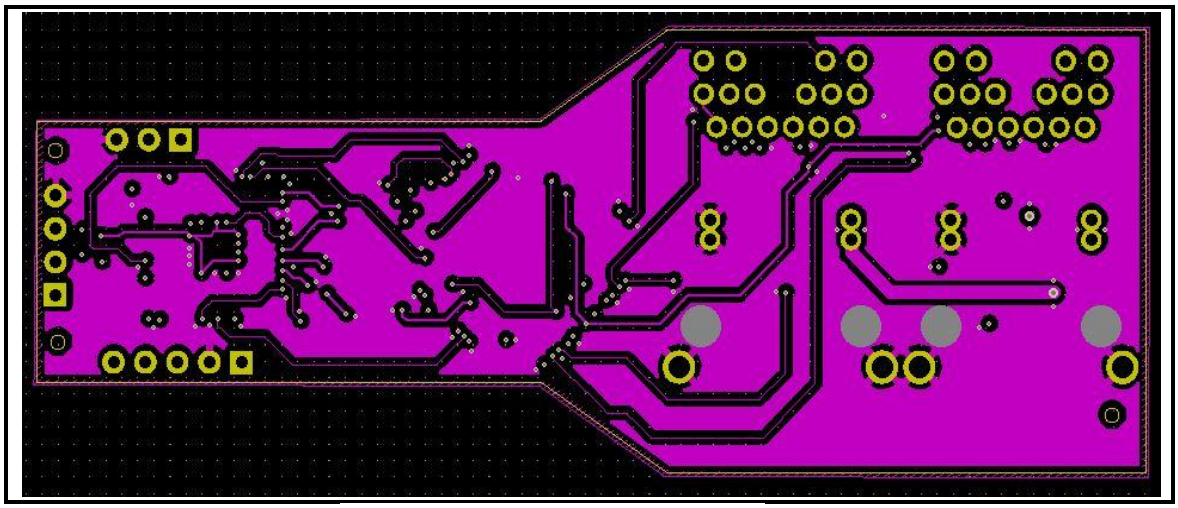

Figure 3.15 : 2eme couche interne

- **La couche B.Cu**

Est la couche de cuivre dessous (4eme couche : plan de masse) la **figure 3.16** illustre cette couche

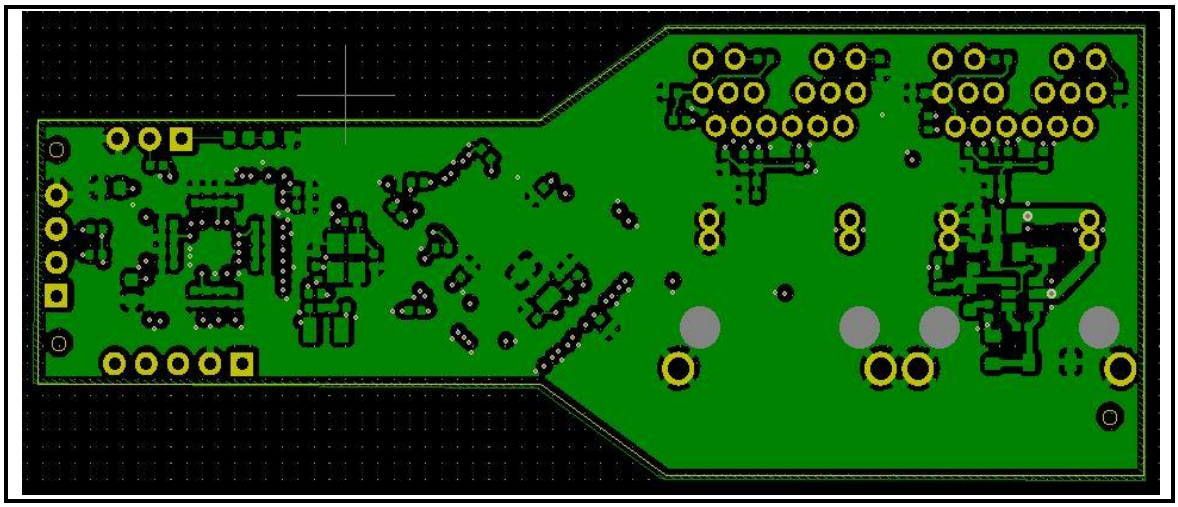

Figure 3.16 : 4eme couche de cuivre

- **Méthode de routage**

Nous avons suivi la règle " ne jamais traverser les divisions dans les plans de sol".

La **figure 3.17** illustre le routage de la 1^{iere} couche interne au-dessous du plan de masse

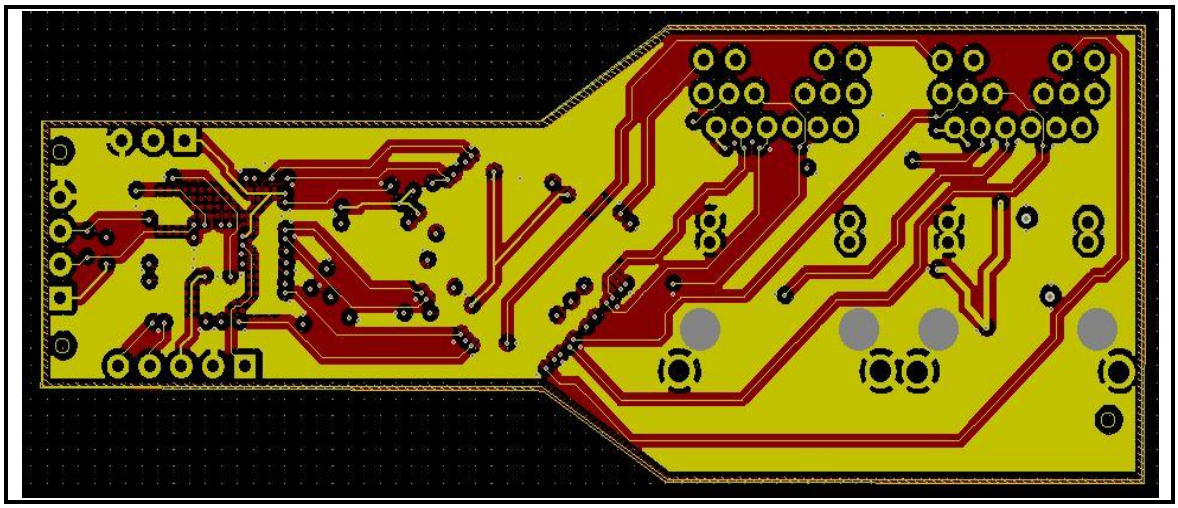

Figure 3.17 : 1iere couche interne au-dessous du plan de masse.

La **figure 3.18** illustre le routage de la 2ieme couche interne au-dessus du plan de masse

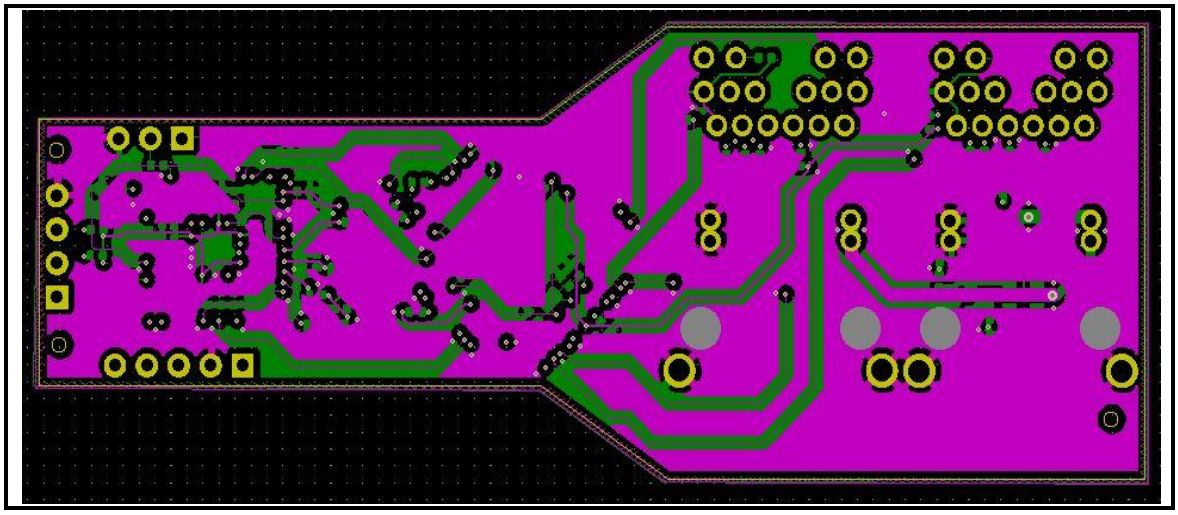

Figure 3.18 : 2eme couche interne au-dessus du plan de masse

b) Modèles 3D

Le modèle 3D de notre I/O Device a été conçu avec le logiciel KiCad. La **figure 3.19**, **3.20**, et **3.21** montrent le modèle 3D de notre I/O Device

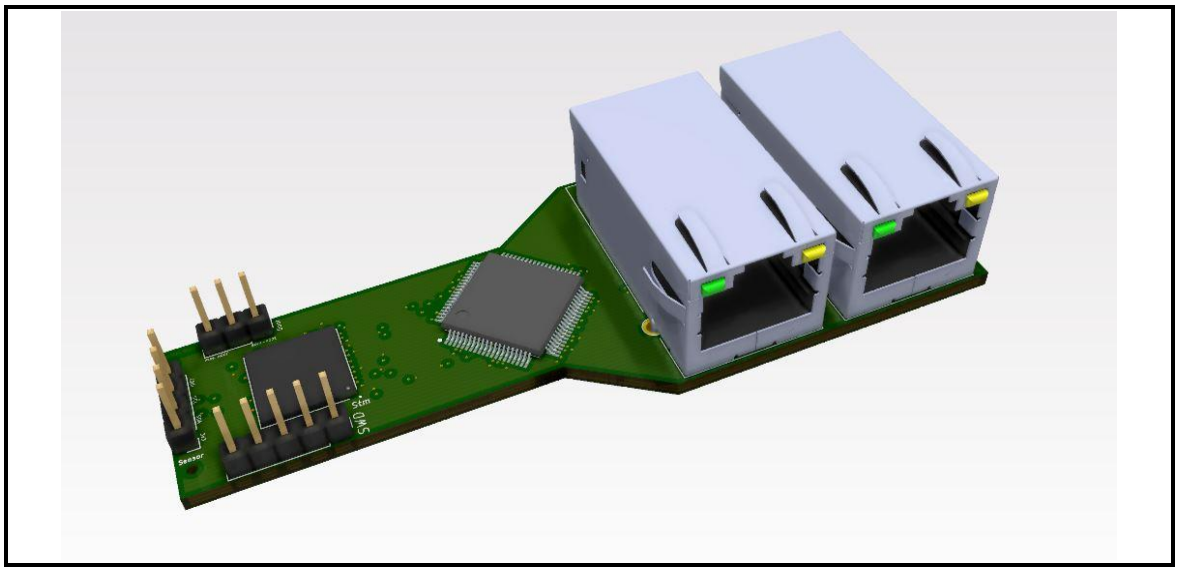

Figure 3.19 : Modèle 3D de notre I/O Device dans KiCad

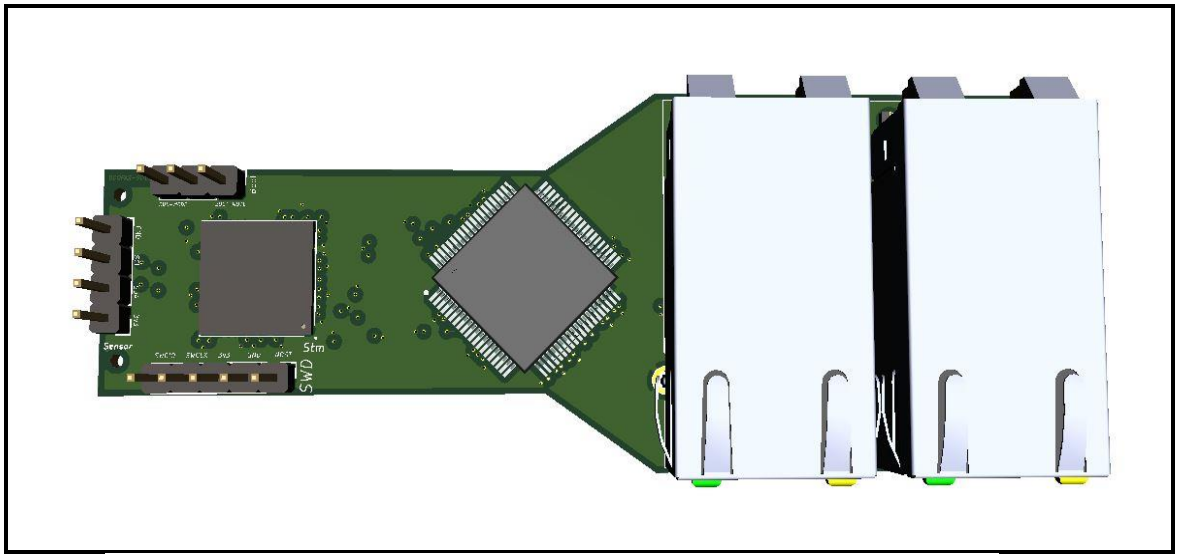

Figure 3.20 : Vue de dessus du Modèle 3D de notre I/O Device dans KiCad

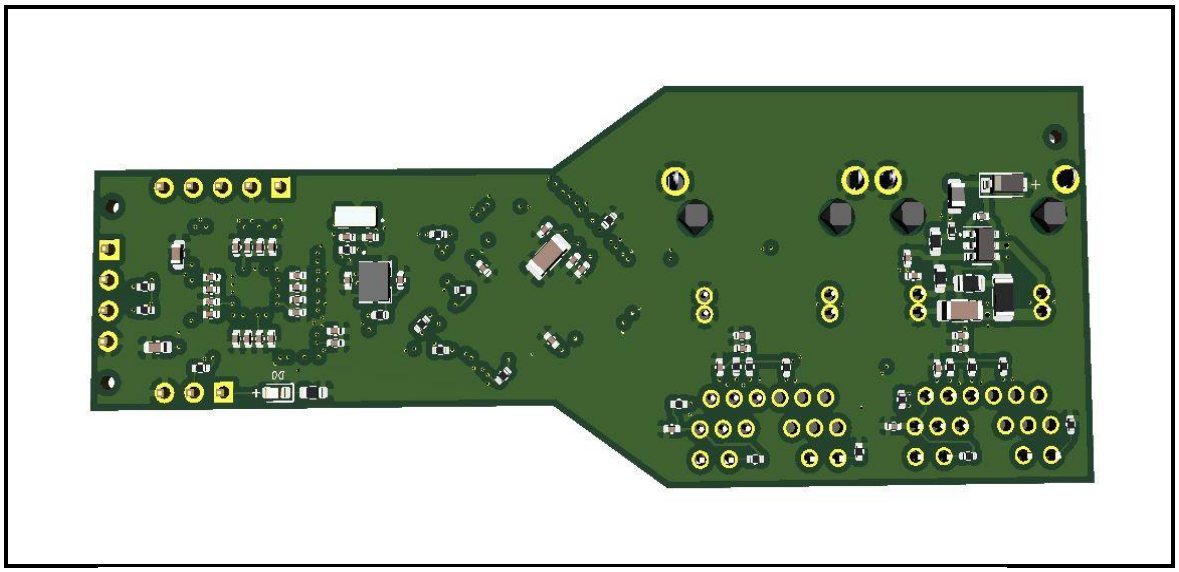

Figure 3.21 : Vue de dessous du Modèle 3D de notre I/O Device dans KiCad

c) Description matériel

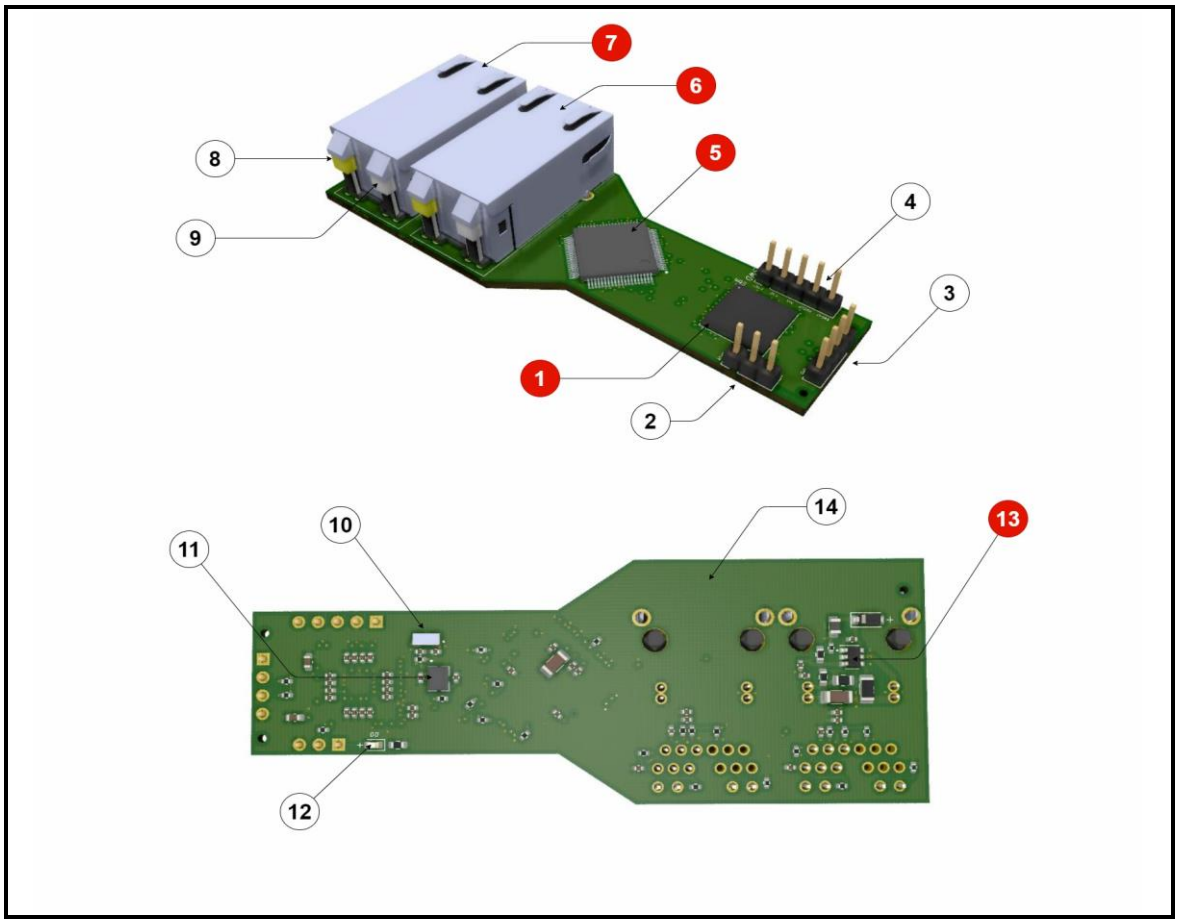

Figure 3.22 : Descriptif de matériel

La **figure 3.22** illustre le descriptif de matériel

-1 : le microcontrôleur STM32F207IGH6

-2 : Interface de BOOT

-3 : Interface de Capteur

-4 : Interface de SWD

-5: Dual-PHY DP83849IF

-6: RJ45 Port B

-7: RJ45 Port A

-8: LED ACT

-9: LED LINK

-10 : crystal 32Khz

-11 : crystal 16Mhz

-12 : LED d'indication d'allumage

-13 : CI Buck MP2451DT-LF-Z

-14 : la carte mere PCB

3.2.3 La conception du power injector

Le power injector est composé d'un PCB à deux couches seulement pour sa simplicité

a) Les couches de cuivre

- **La couche F.Cu**

Est la couche de cuivre dessus, la **figure 3.23** illustre cette couche

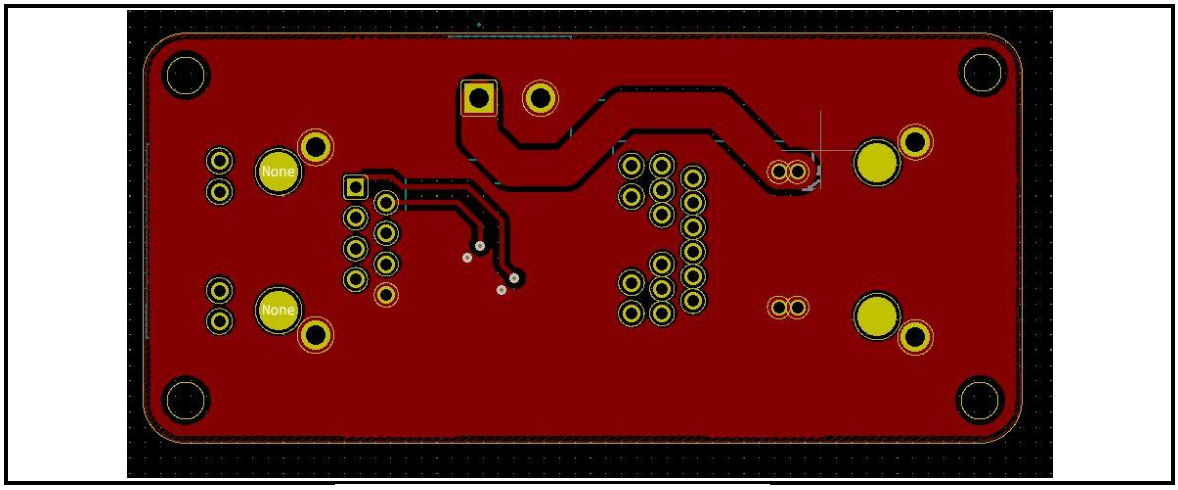

Figure 3.23 : Couche de cuivre F.Cu

- **La couche B.Cu**

Est la couche de cuivre dessous la **figure 3.24** illustre cette couche

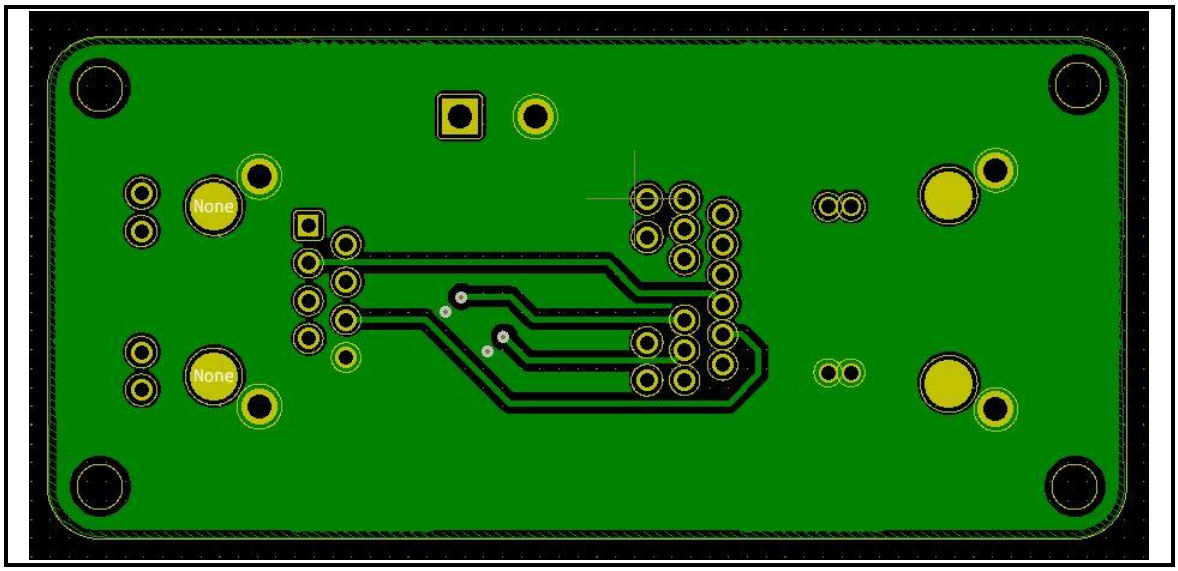

Figure 3.24 : Couche de cuivre B.Cu

b) Modèle 3D

Le modèle 3D de notre power injector a été conçu avec le logiciel KiCad. La **figure 3.25**, **3.26**, et **figure 3.27** montrent le modèle 3D de notre power injector

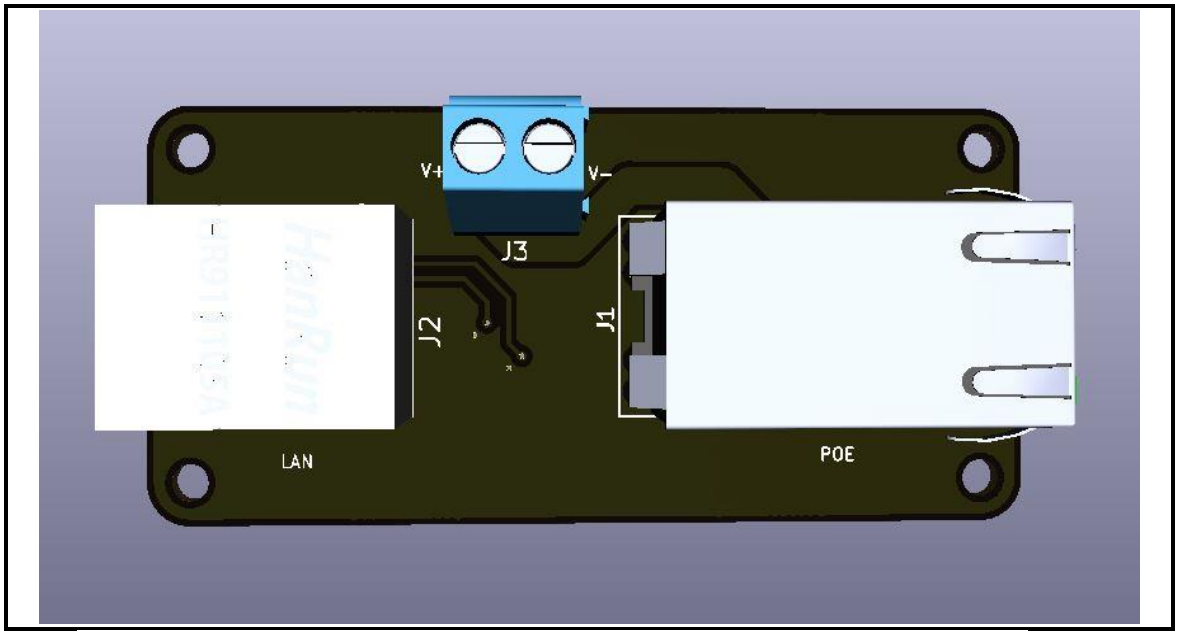

Figure 3.25 : Vue de dessus du Modèle 3D de notre power injector dans KiCad

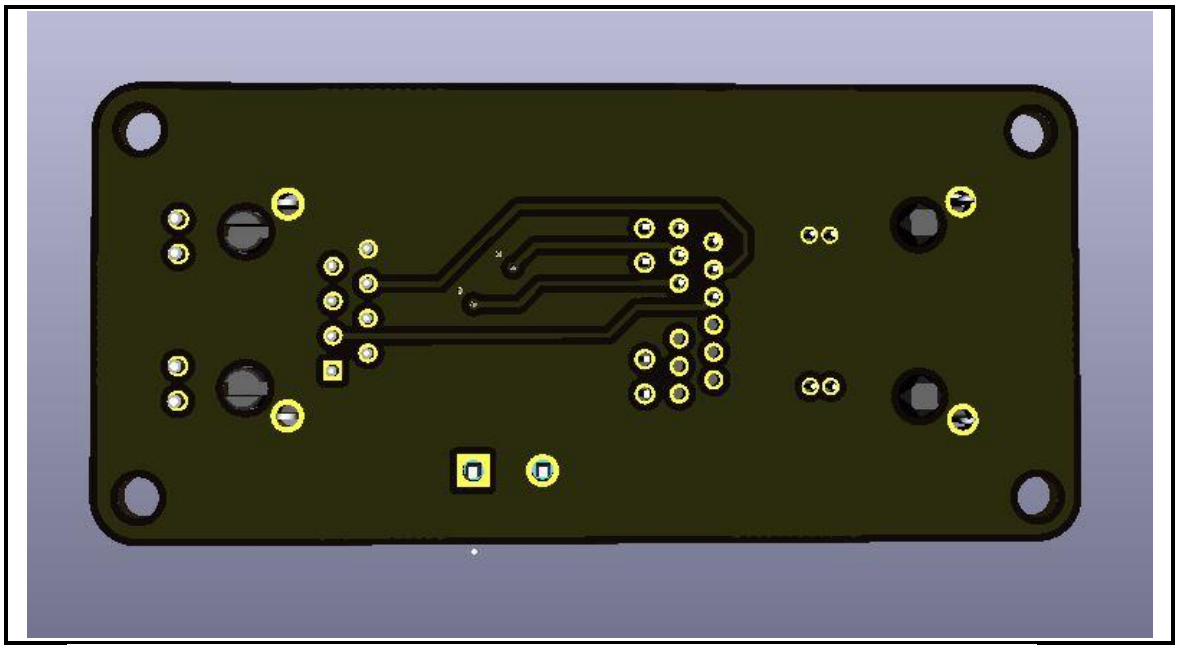

Figure 3.26 : Vue de-dessous du Modèle 3D de notre power injector dans KiCad

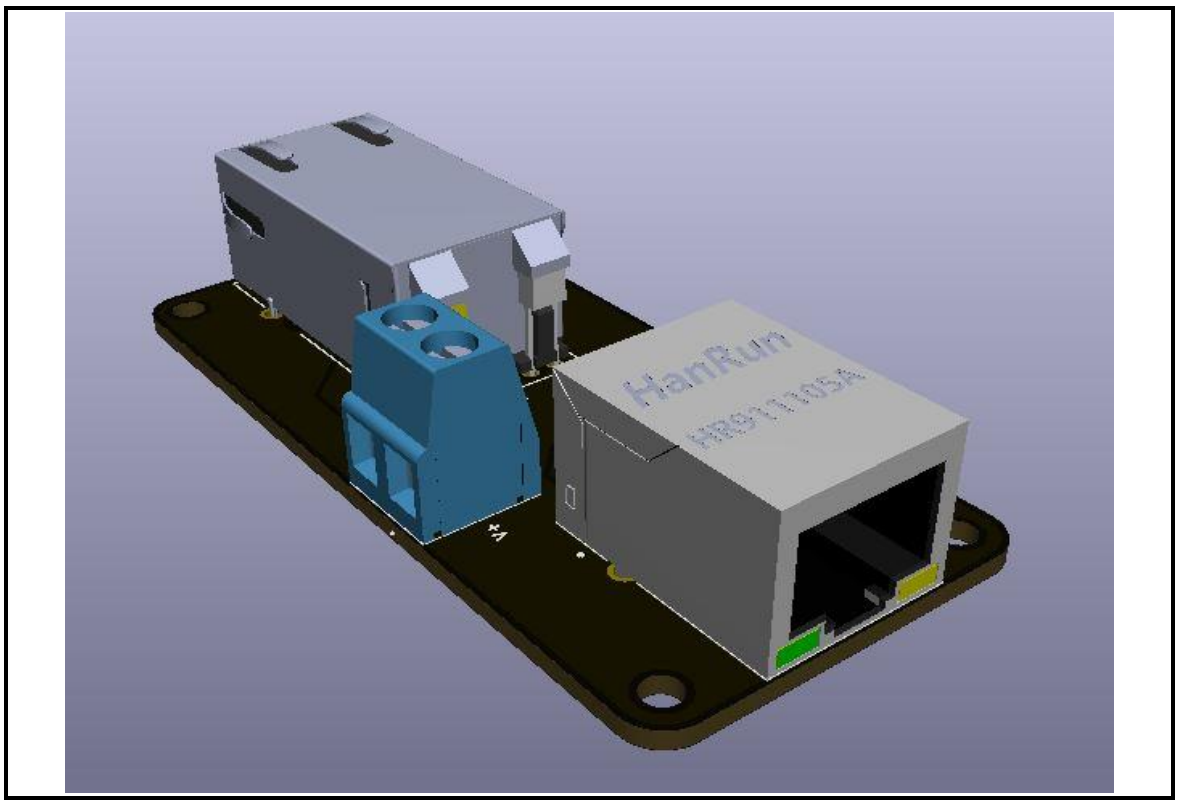

Figure 3.27 : Modèle 3D de notre power injector dans KiCad

3.3 La réalisation du I/O Device

La réalisation de nôtre I/O Device a été faite par l'entreprise JLCPCB. Cette réalisation est divisée en deux parties

3.3.1 PCB

Il faut tout d'abord générer le fichier gerber dans le logiciel KiCad ensuite, le télécharger sur le site de JLCPCB [20], la **figure 3.28** montre comment télécharger le fichier gerber dans JLCPCB.

Figure 3.28 : Comment télécharger le fichier gerber

Après avoir télécharger le fichier gerber dans JLCPCB, le site reconnaît le PCB comme l'indique la **figure 3.29**

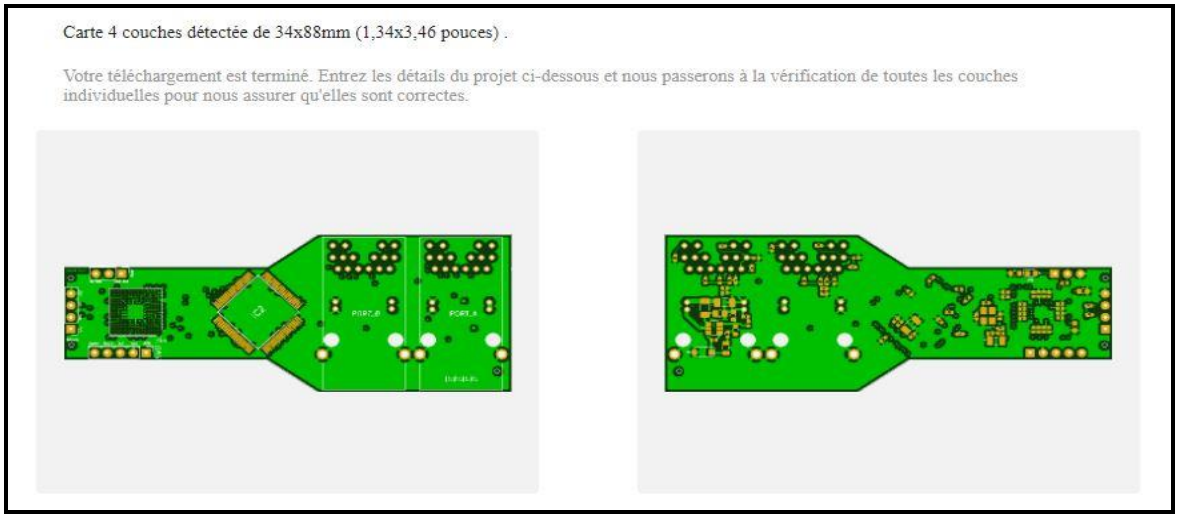

Figure 3.29 : GERBER viewer

Nous devons ensuite faire entrer les détails du projet comme l'ordre des couches, la couleur du PCB...etc.

3.3.2 PCBA

C'est l'assemblage des composants dans le pcb (PCB Assembly)

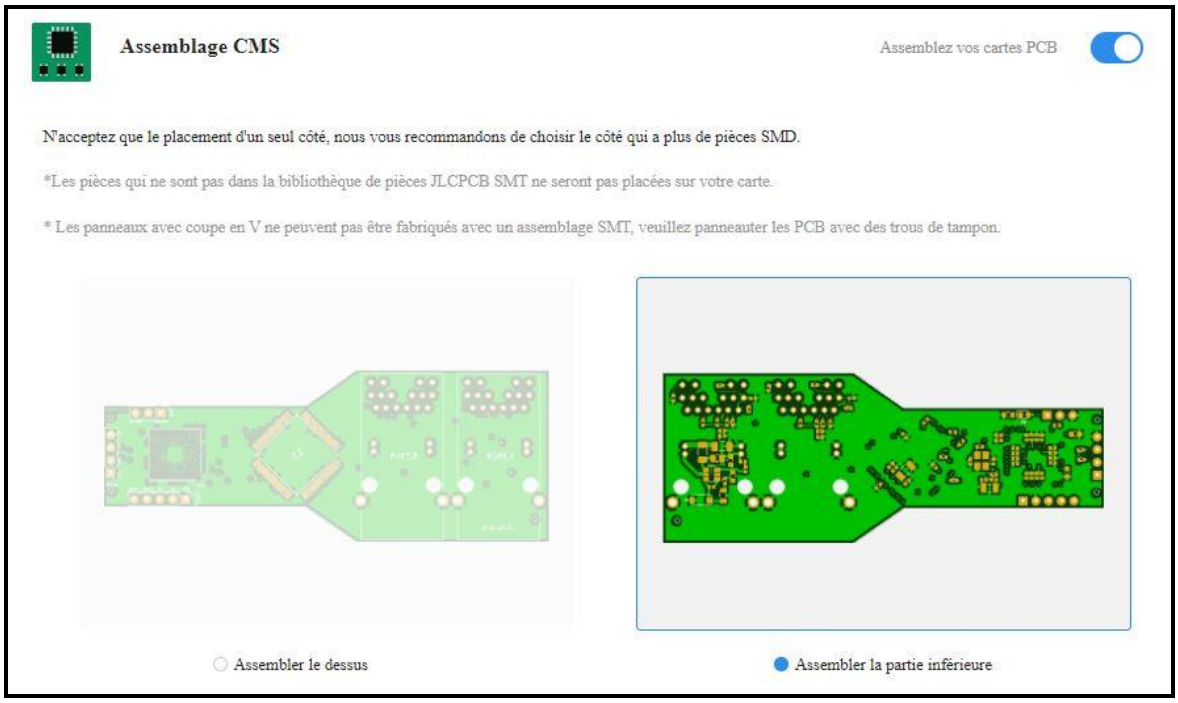

Figure 3.30 : PCBA et le côté de l'assemblage

La **figure 3.30** illustre comment activer l'option PCBA dans JLCPCB et le choix du côté d'assemblage.

L'entreprise JLCPCB peuvent assembler seulement un seul côté donc nous avons choisi le côté qui contient beaucoup de composants, ensuite nous avons téléchargez deux fichiers, le fichier BOM et le fichier POS qui sont les deux fichiers nécessaires pour la PCBA.

Le montage et la soudure des composants au-dessus du pcb est faite par BOUCETTA Belaid technicien dans la maintenance des smartphones.

3.4 Conclusion

La conception des circuits imprimés nécessite des connaissances de l'EMI et comment l'éviter pour assurer le bon fonctionnement de l'appareil conçu ainsi que pour les équipements qui l'entoure.

Chapitre 4 Tests et résultats

4.1 Introduction

Après la réalisation et la fabrication du dispositif, on a procédé aux tests software et hardware, en utilisant une alimentation de tension DC variant de 0v à 15 v pour situer la plage des tensions dont le dispositif est fonctionnel.

Pour le software, on a utilisé trois logiciels, **BOOTP-DHCP TOOL** pour tester le serveur BOOTP et ENIP, **QMOdMaster** pour tester le serveur Modbus et finalement **wireshark** pour voir toutes les trames envoyées entre le dispositif et l'ordinateur.

4.2 Test

Pour les tests on a trois dispositifs et un injecteur de puissance (power injector) fabriqués selon notre conception **figure 4.1**

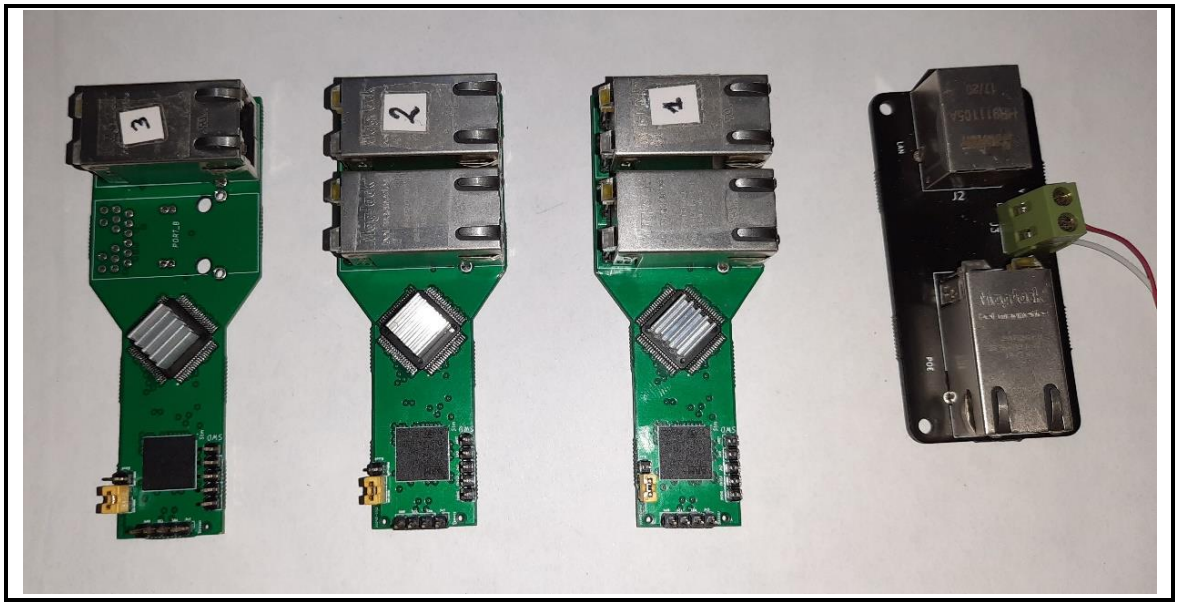

Figure 4.1 : Trois dispositifs et un injecteur

4.2.1 Test des tensions de Marche

Théoriquement selon les calculs que nous avons fait au niveau du Buck, les tensions de marche sont à partir de 8V jusqu'à 36V, mais pratiquement en industrie les tensions d'alimentation sont standard (5v, 12v, 24v), on a testé notre dispositif **figure 4.2** et **figure 4.3** avec un générateur de tension max 15 V et on a remarqué qu'il fonctionne entre 8.1V et 15v, après nous l'avons utilisé avec une tension de 24 V au niveau de l'entreprise DOOFAS vu qu'en industrie on utilise la tension 24 V

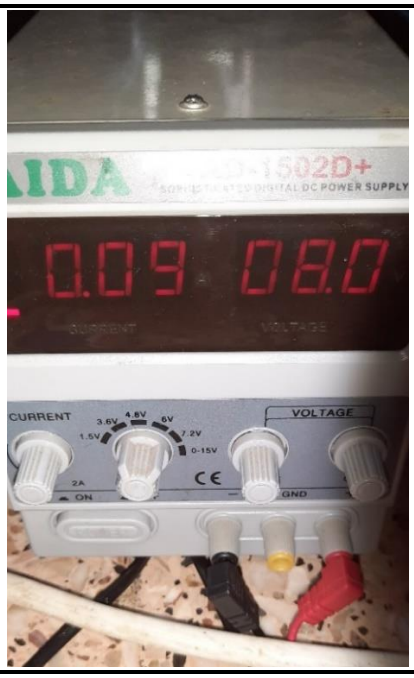

Figure 4.2 : Tension minimale

Figure 4.3 : Tension maximale (maximal du l'alimentation)

4.2.2 Compilation du code

La compilation du code **figure 4.5** se fait sur le logicielle **STM32cubeIDE figure 4.4** pour vérifier s'il n'y a pas des erreurs au niveau du programme

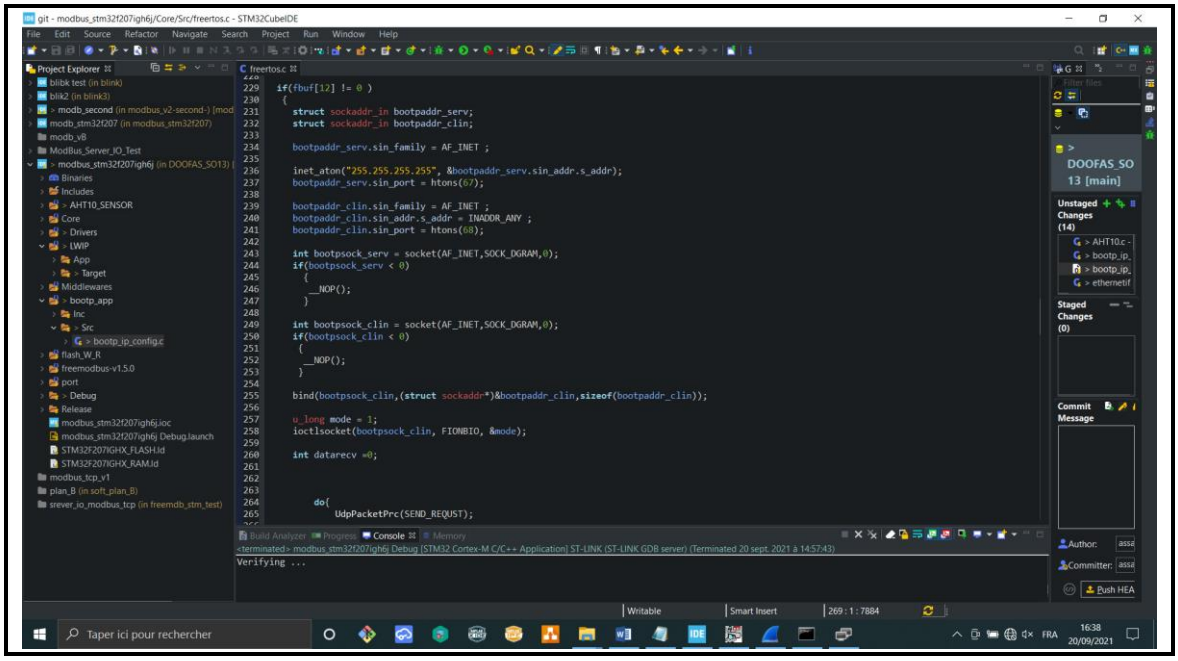

Figure 4.4 : Interface du STM32cubeIDE

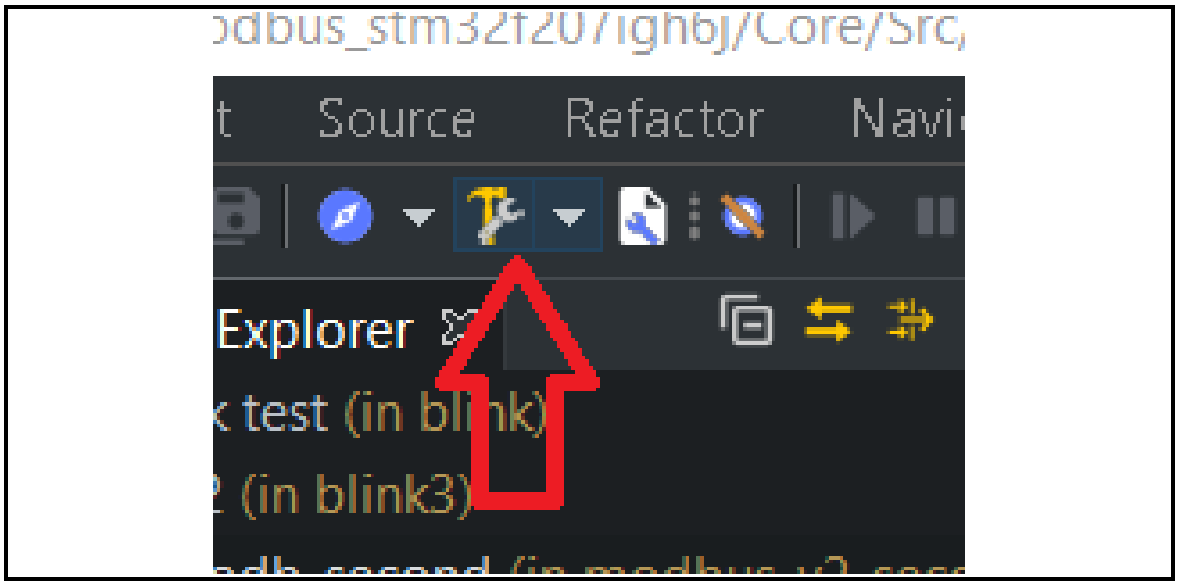

Figure 4.5 : Compilation du code

Après la compilation on a remarqué que le programme utilise 36.17 % de la mémoire RAM et 11.11 % de la mémoire FLASH ce qui est acceptable **figure 4.6**

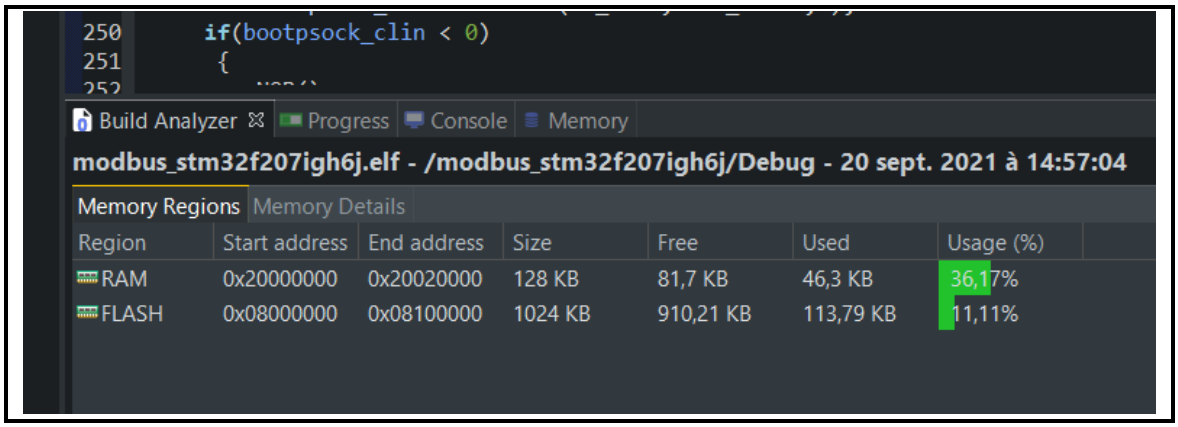

Figure 4.6 : Billon d'utilisation des ressource

4.2.3 Débogage

Pour faire le débogage **figure 4.7** et l'injection du programme sur le STM il faut connecter le programmer ST-LINK/V2 sur le port SWD, pour alimenter le dispositif, connecter le premier port du dispositif et le port POE de l'injecteur avec un câble Ethernet, pour injecter la tension, connecter l'injecteur avec l'alimentation et Connecter le programmeur avec l'ordinateur via l'USB **figure 4.8**

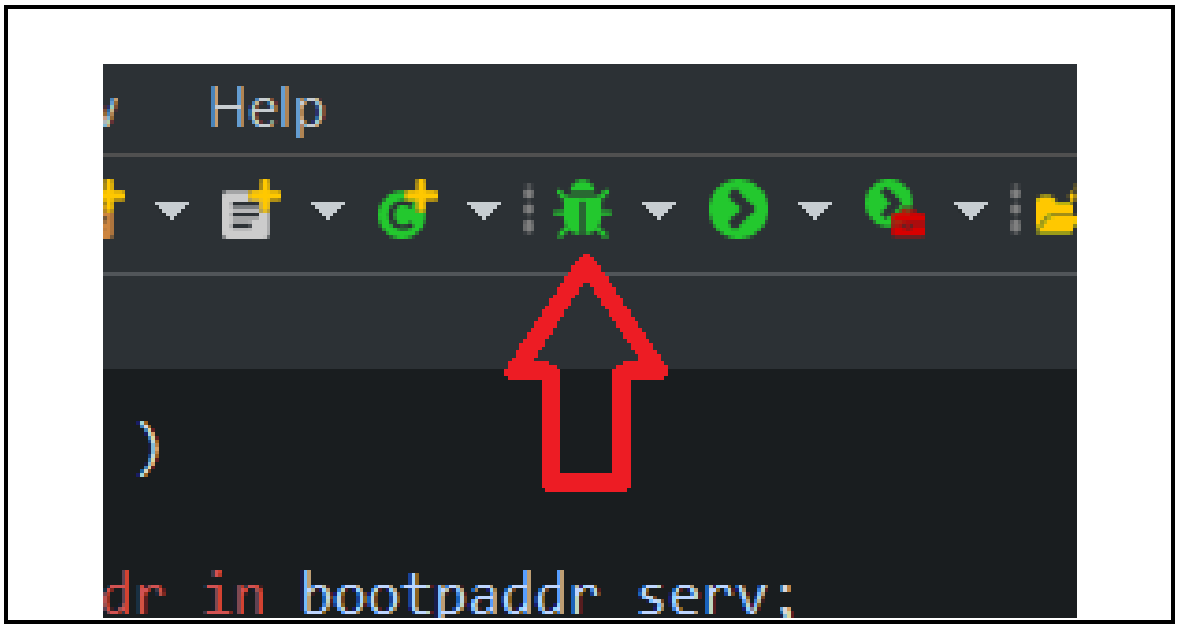

Figure 4.7 : Lancement du débogage

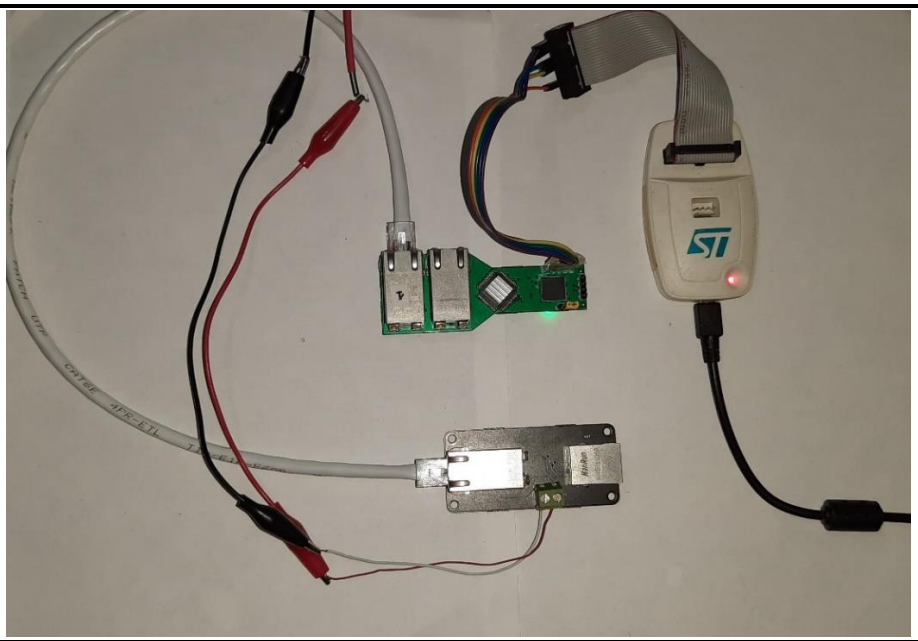

Figure 4.8 : Transfert du programme

4.2.4 Configuration du port Ethernet de l'ordinateur

La configuration de la communication entre le dispositif et l'ordinateur se fait selon les étapes suivantes :

- Sélectionner le port Ethernet pour le configurer, **figure 4.9**
- Cliquer deux fois sur le **Protocol Internet version 4(TCP/UPv4), figure 4.10**
- Sélectionner **Utiliser l'adresse IP suivante** et configurer l'adresse IP et le masque **figure 4.11**

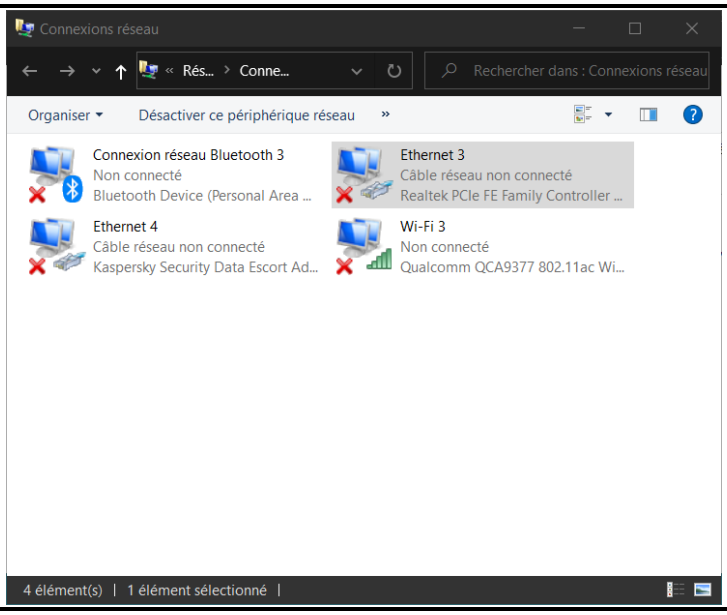

Figure 4.9 : Sélection du port Ethernet

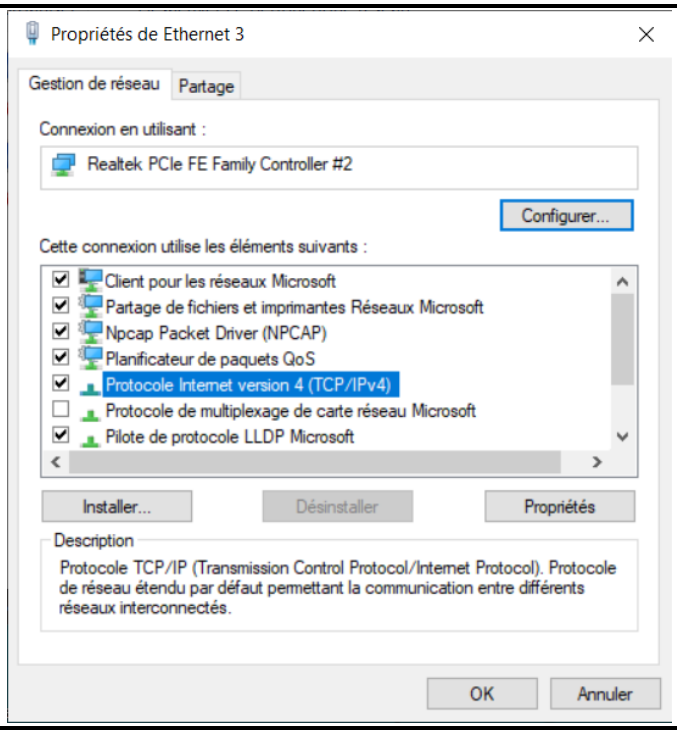

Figure 4.10 : Propriétés du port Ethernet

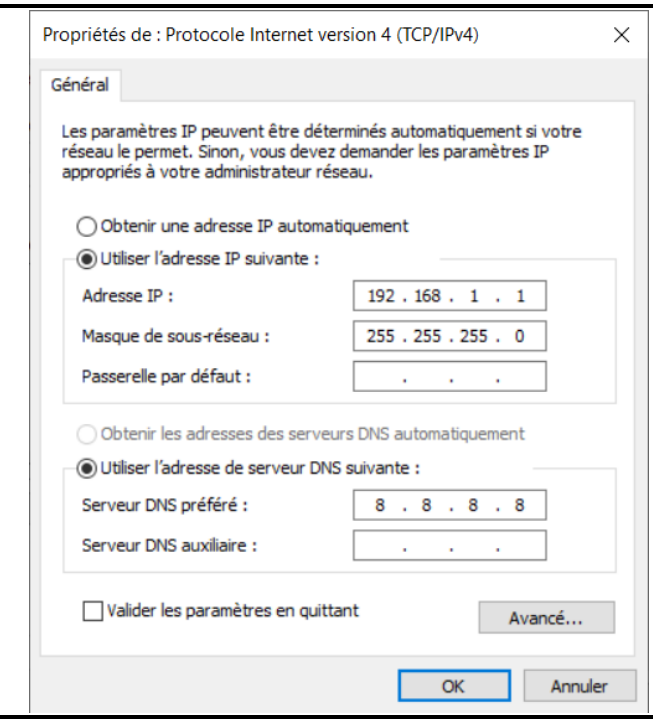

Figure 4.11 : Configuration de l'adresse IP et masque

4.2.5 Test du serveur BOOTP

Après l'injection du programme sur le STM, le bootp est automatiquement activé pour l'obtention de l'adresse IP, on a programmé et tester les serveurs des trois dispositifs avec **BOOTP-DHCP TOOL**

Premièrement il faut sélectionner l'interface réseau **figure 4.12**

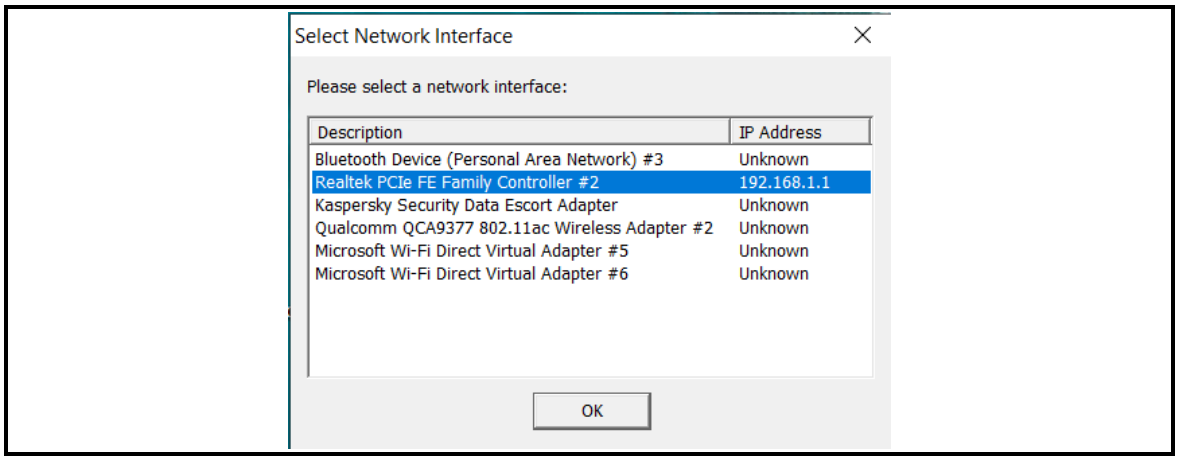

Figure 4.12 : Sélection de l'interface réseau

Après la sélection de l'interface réseau, le logiciel entre en mode écoute des requetés bootp reçues sur l'adresse IP de l'ordinateur et le port 68 **figure 4.13**

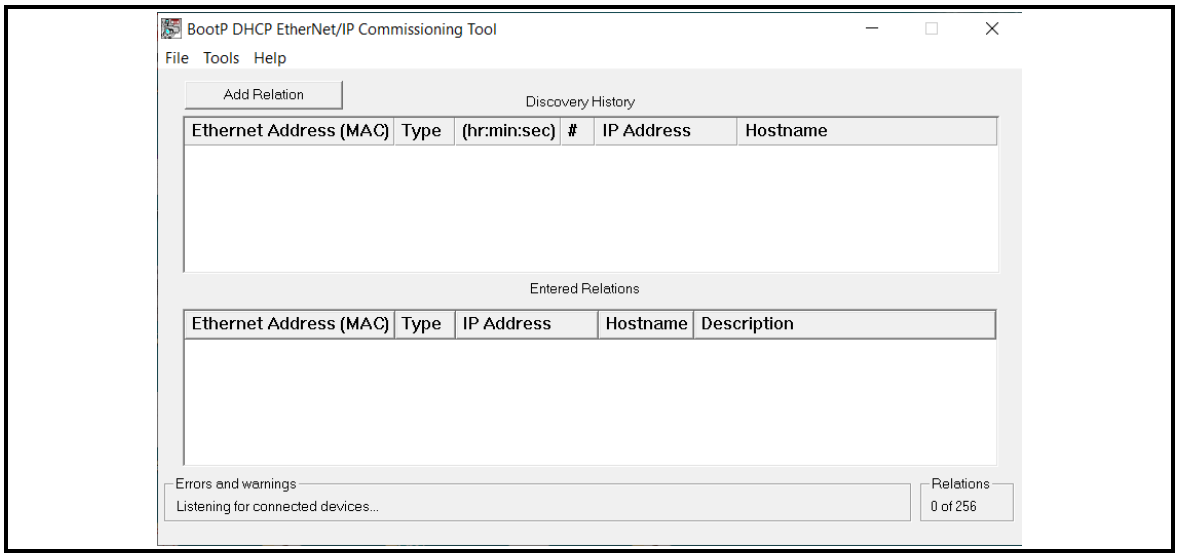

Figure 4.13 : Etat initial : pas des requête bootp

S'il y a une réception d'une requête Bootp, elle va être affichée sur le logiciel avec l'adresse MAC de l'envoyeur **figure 4.14.**

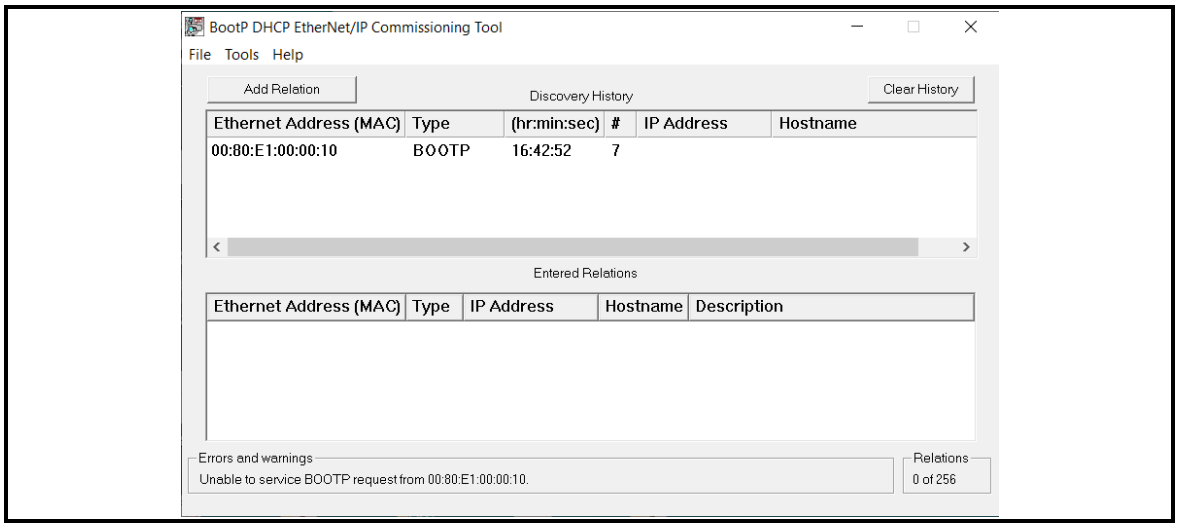

Figure 4.14 : Une requeté bootp reçue

On aperçoit sur le **wireshark** les requetés bootp reçues qui seront transmises vers l'adresse IP 255.255.255.255 avec l'adresse MAC 00:80:E1:00:00:10, **figure 4.15** .

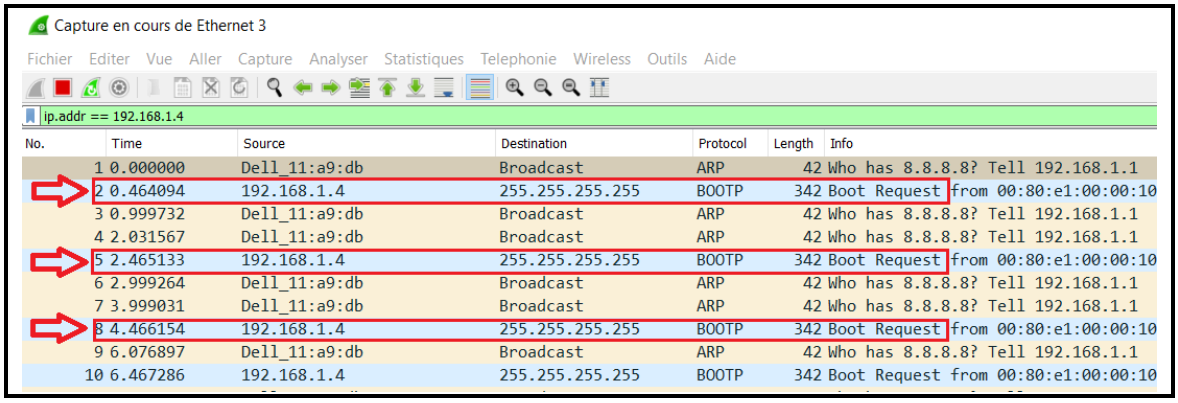

Figure 4.15 : Vue des requetés bootp reçu sur **wireshark**

En cliquant deux fois sur la requête reçue sur **BOOTP-DHCP TOOL** puis configurer l'adresse IP qui sera transmise vers le dispositif, **figure 4.16.**

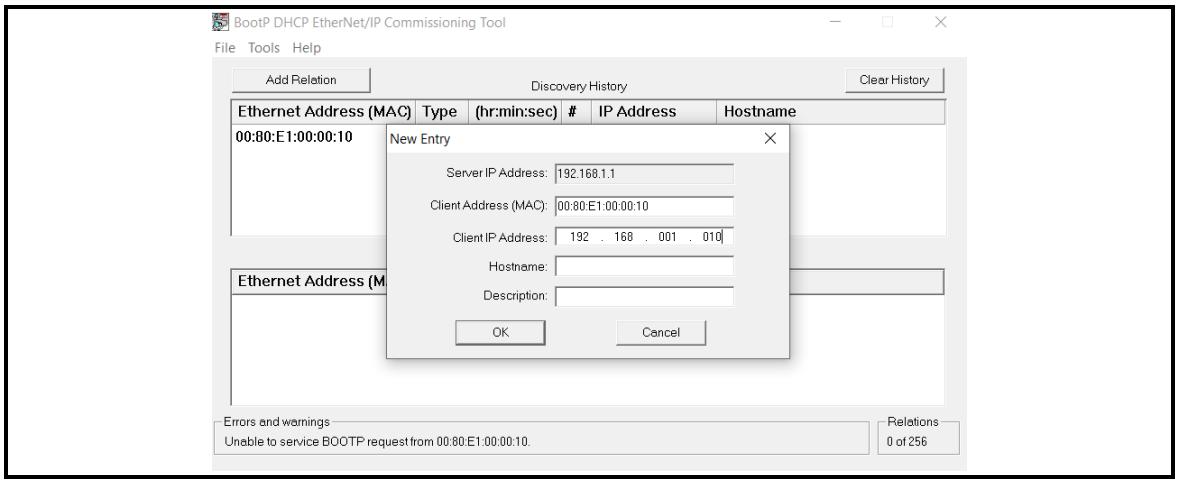

Figure 4.16 : Configuration de l'adresse IP du dispositif

Finalement le logiciel va envoyer la réponse Bootp vers le dispositif qui va configurer sa nouvelle adresse IP

4.2.6 Test si le dispositif est actif

Après la configuration de l'adresse IP avec le Bootp il faut tester si la configuration a réussi, pour cela on a utilisé l'invite de commandes (CMD) **figure 4.17** pour envoyer la trame Ping.

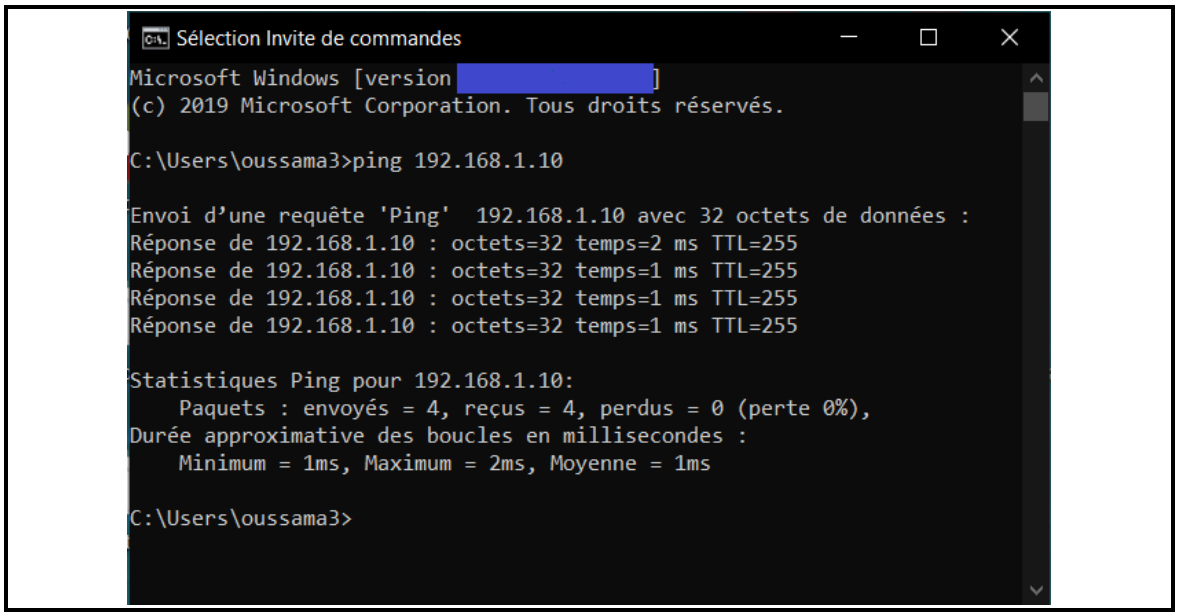

Figure 4.17 : Test Ping sur l'invite de commandes

On aperçoit que le dispositif répond à la trame Ping,

4.2.7 Test du serveur ENIP

Après la configuration Bootp, le dispositif a eu une adresse IP tant qu'il est alimenté, si le dispositif est débranché de l'alimentation ou a subi un redémarrage, l'adresse IP va être oublier et le serveur va envoyer les requêtes Bootp pour avoir une adresse IP, dans ce cas il y aura des conflits de communication(avec bootp activé) car dans l'industrie le dispositif va être connecté sur un réseau à plusieurs dispositifs , pour cela il faut configurer l'adresse IP et l'enregistrer sur la mémoire flash du dispositif en désactivant le bootp.

Pour tester le serveur ENIP il faut sélectionner l'adresse IP du dispositif, **figure 4.18** et cliquer sur le bouton **Disable BOOTP/DHCP.**

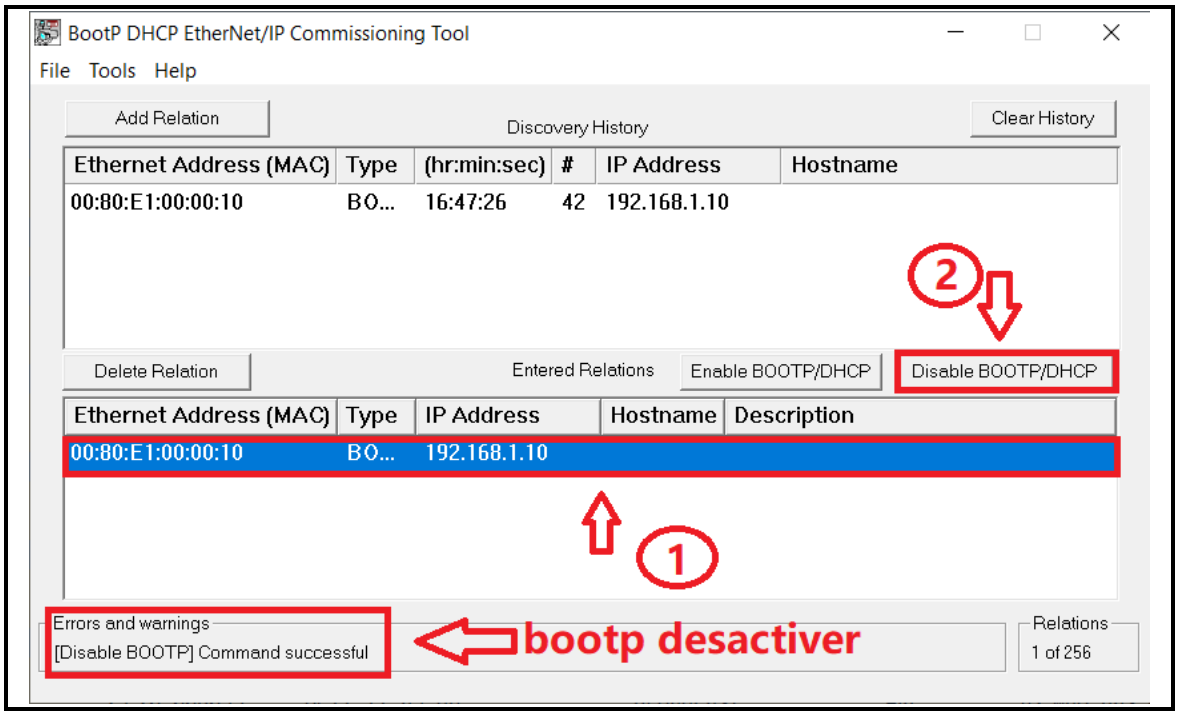

Figure 4.18 : Désactivation du bootp

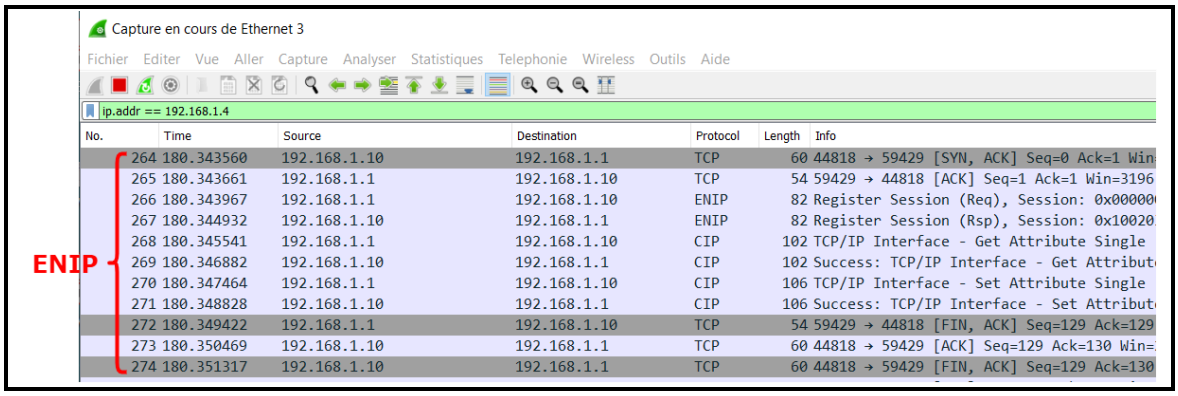

Figure 4.19 : Vue de Communication ENIP sur **wireshark**

Le bootp est désactivé, pour l'activer on procède aux mêmes étapes déjà citées précédemment et cliquer sur le bouton **Enable BOOTP/DHCP**

4.2.8 Test du serveur Modbus

Le test du serveur Modbus se fait avec le logiciel **QMOdMaster figure 4.22**, on a connecté le capteur AHT10 **figure 4.20** et alimenter le dispositif **figure 4.21**

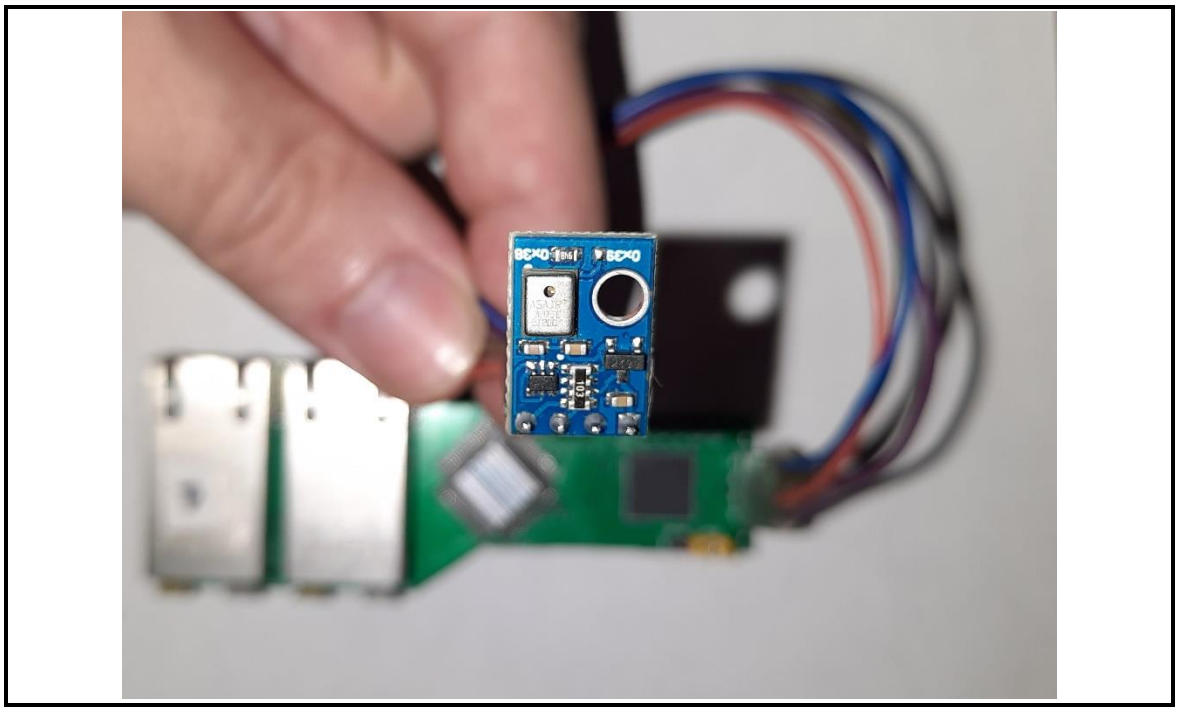

Figure 4.20 : Capteur de température et l'humidité AHT10

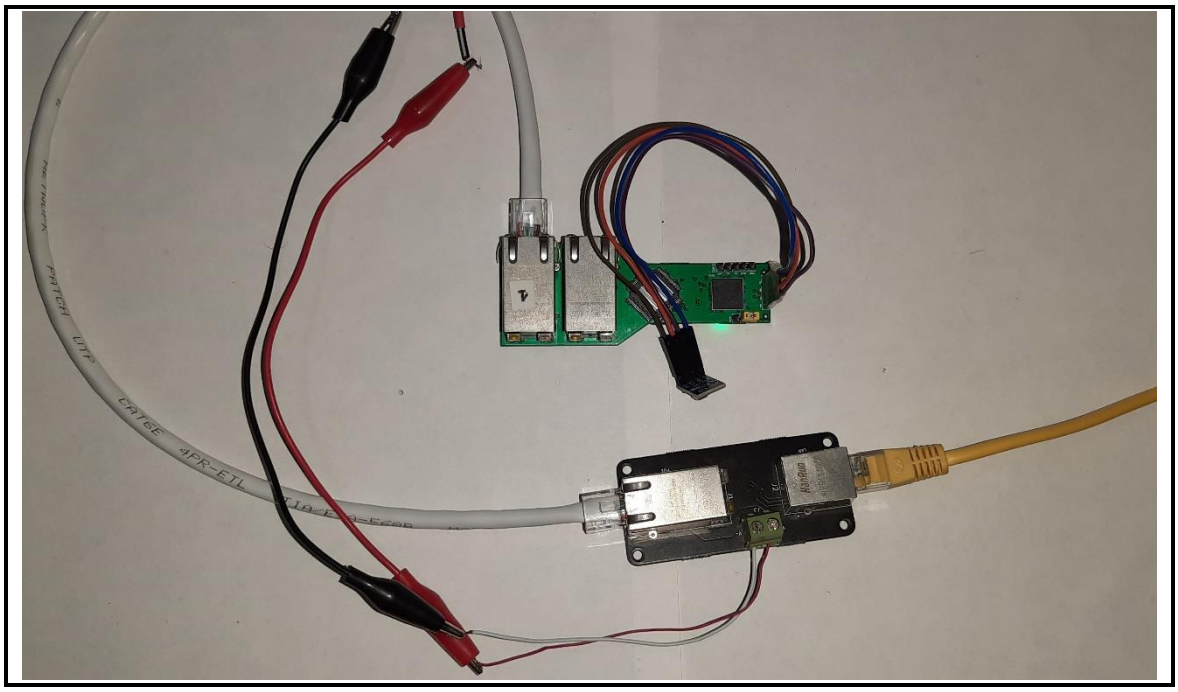

Figure 4.21 : Test avec le capteur AHT10

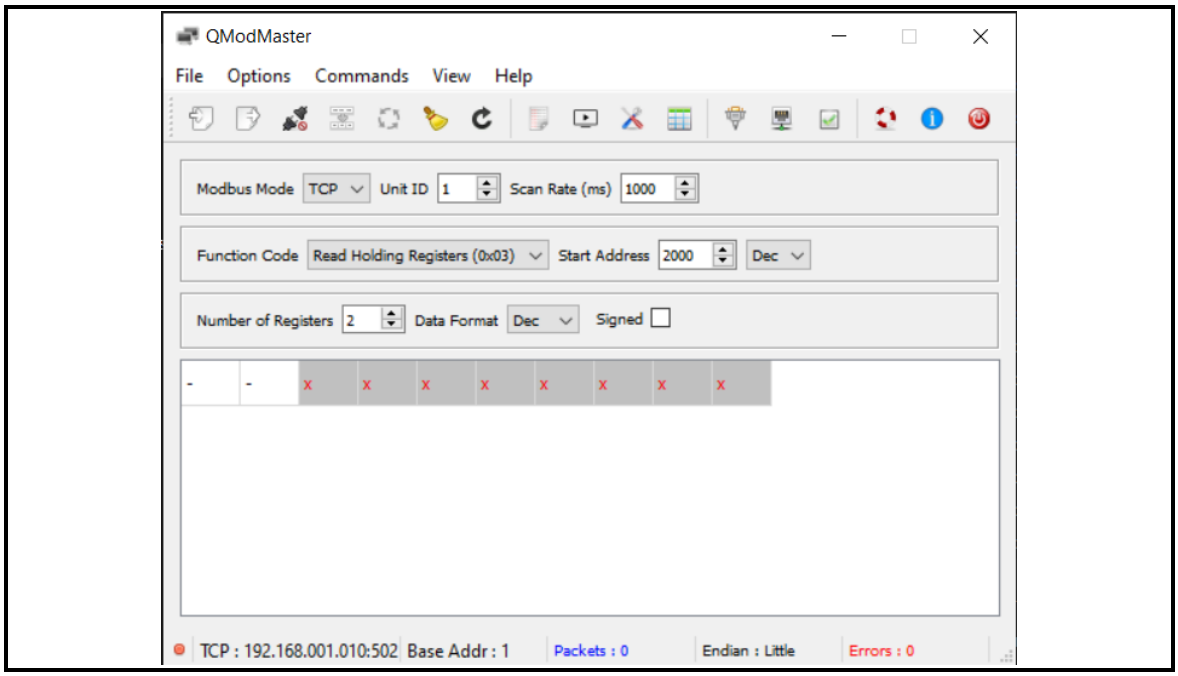

Figure 4.22 : QMOdMaster

La configuration de Modbus TCP, **figure 4.23** et **figure 4.24**

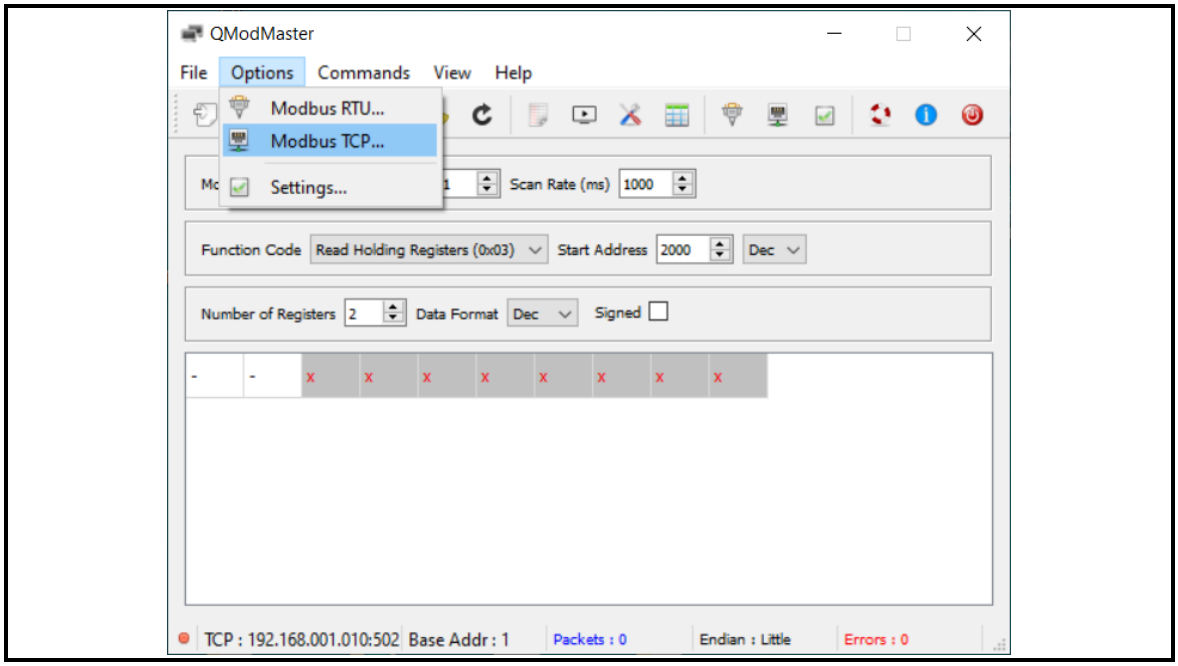

Figure 4.23 : Configuration modbus TCP

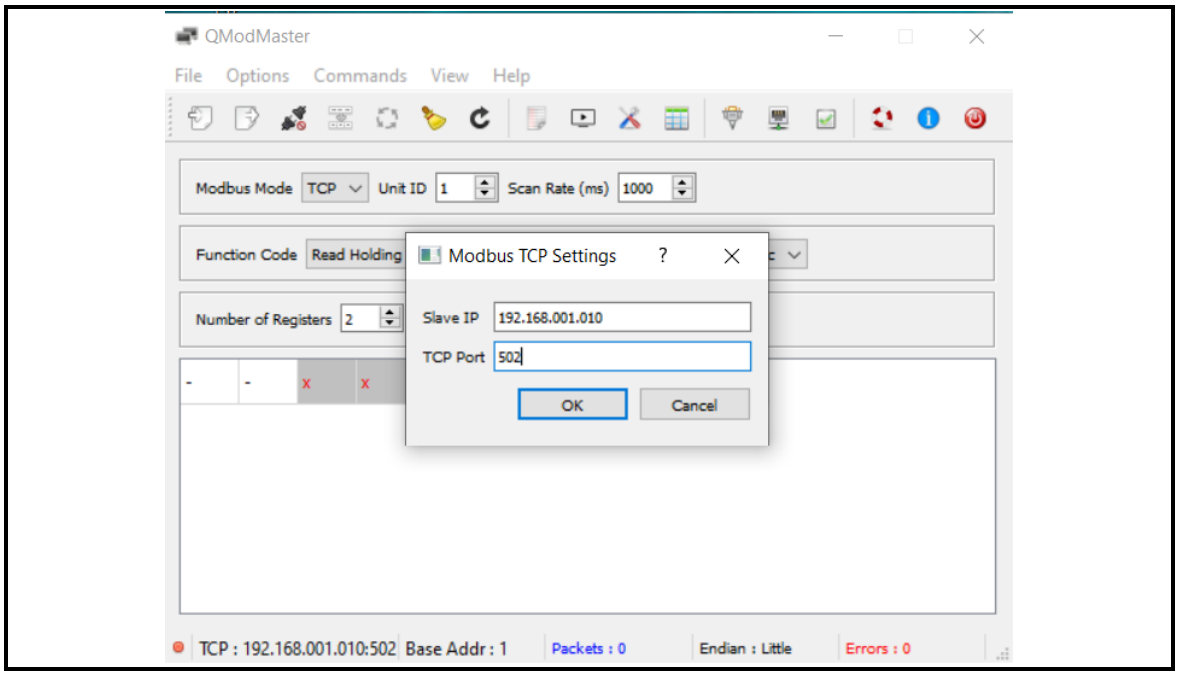

Figure 4.24 : Configuration de l'adresse IP est Port

Finalement on clique sur le bouton 1 pour la connexion avec le serveur Modbus et on clique sur le bouton 2 pour l'envoie de la requête Modbus afin de lire la température et l'humidité à partir du dispositif **figure 4.25**

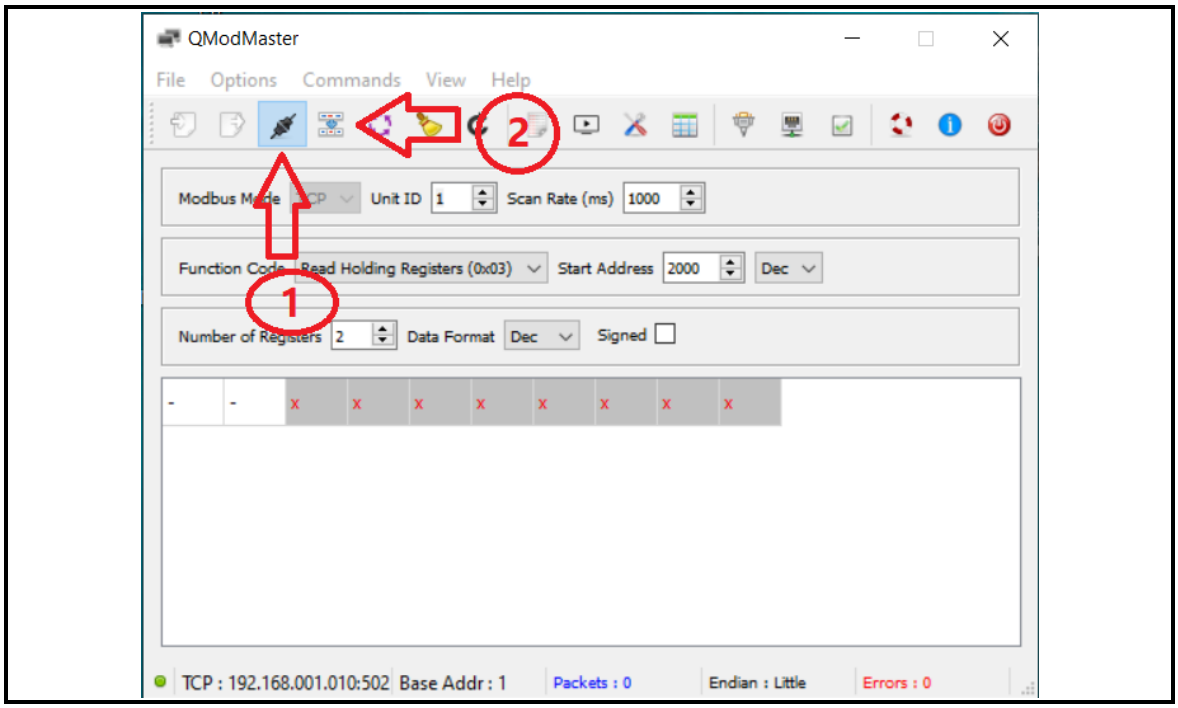

Figure 4.25 : Connexion avec le serveur modbus

Après l'envoi de la requête Modbus on reçoit la valeur de la température et de l'humidité, température = 27C°, l'humidité = 61% **figure 4.26**

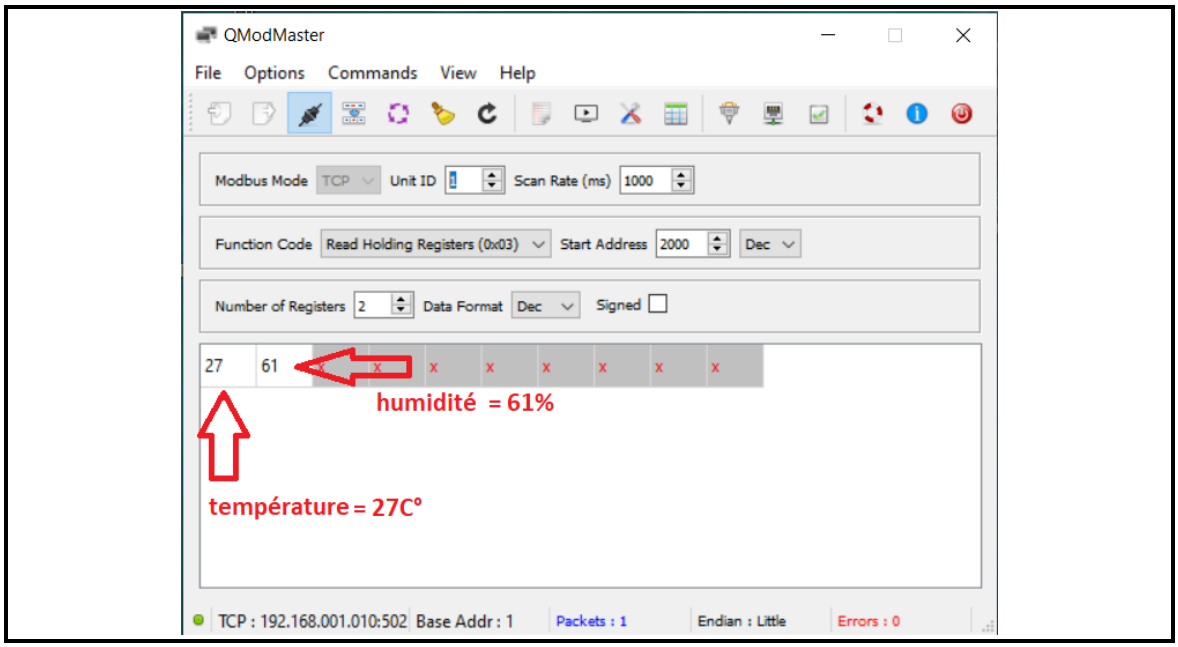

Figure 4.26 : Température et l'humidité reçu

On aperçoit la connexion du client Modbus avec le serveur sur **wireshark, figure 4.27**

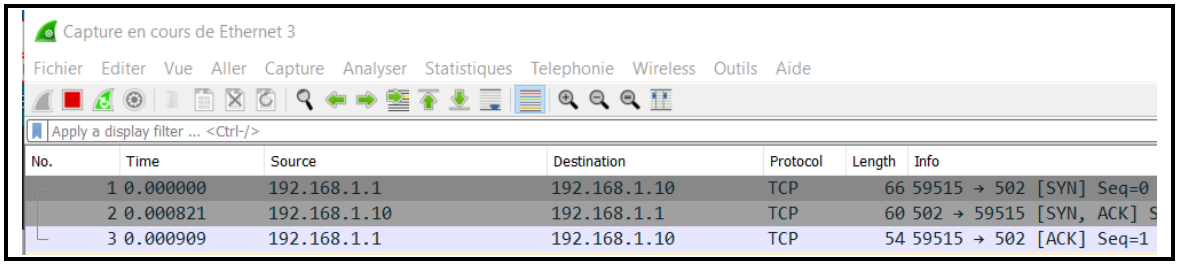

Figure 4.27 : Connexion du client Modbus avec le serveur vu sur **wireshark**

On aperçoit la requête Modbus et la réponse sur **wireshark**, **figure 4.28**

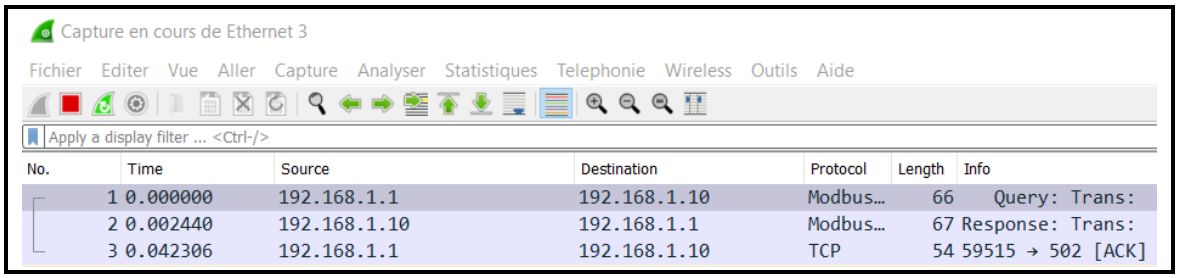

Figure 4.28 : Vue de requête Modbus et la réponse sur **wireshark**

On a testé le montage en anneau **figure 4.29** et refait toutes les étapes précédentes avec la configuration suivante :

Configuration du dispositif numéro 1 :

- Adresse IP : **192.168.1.10**
- Adresse MAC : **00 :80 : E1 :00 :00 :10**

Configuration du dispositif numéro 2 :

- Adresse IP : **192.168.1.12**
- Adresse MAC : **00 :80 :E1 :00 :00 :20**

Configuration du dispositif numéro 3 :

- Adresse IP : **192.168.1.14**
- Adresse MAC : **00 :80 :E1 :00 :00 :40**

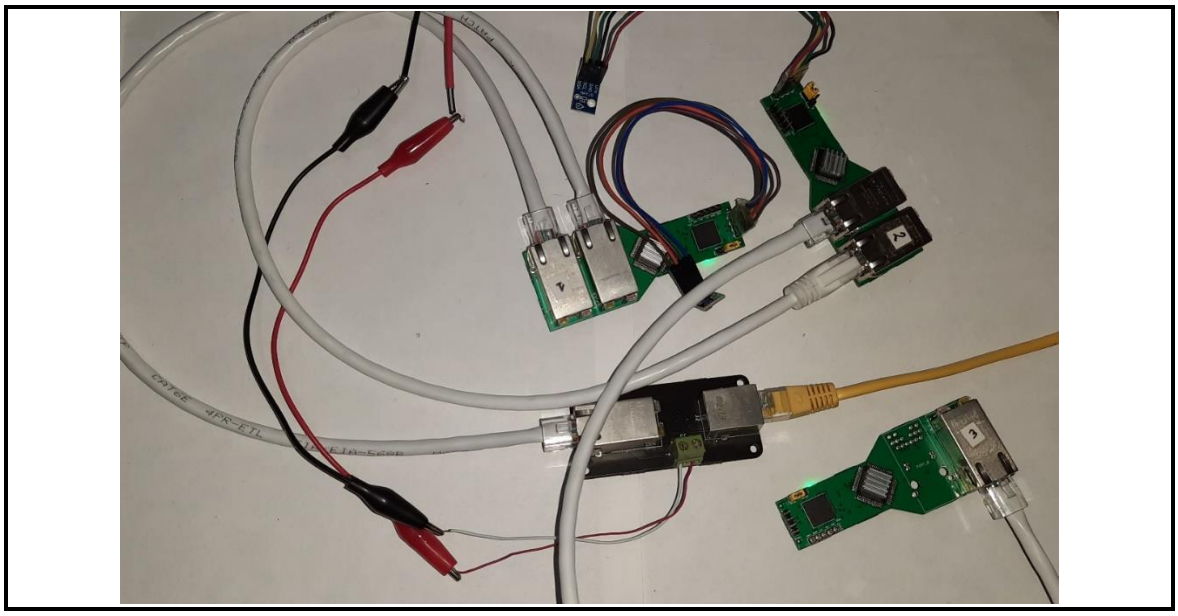

Figure 4.29 : Montage en anneau

4.3 Résultat

Après les tests qu'on a faits nous avons obtenus les résultats suivants :

- **Résultat de test des tensions de marche :** Les tensions de marche après les tests sont à partir de 8.1v jusqu'à 24v
- **Résultat de test du serveur Bootp :** le serveur Bootp est fonctionnel
- **Résultat de test du serveur ENIP :** le serveur ENIP est fonctionnel
- **Résultat de test du serveur Modbus :** le serveur Modbus est fonctionnel

- **Résultat de branchement en anneau :** on a remarqué que le troisième dispositif branché sur le réseau parfois ne répond pas aux trames Ping et on a conclu que le problème se situe au niveau de l'ordinateur car il n'envoie pas Juste les Ping mais aussi des trames ARP en même temps ce qui crée des conflits

4.4 Conclusion

Dans ce chapitre nous avons procédé aux explications étape par étape des tests hardware et software que nous allons réellement concevoir et réaliser.

Conclusion générale

Ce travail contribue à l'intégration du protocole de communication Modbus TCP/IP sur un système embarqué.

Nous avons eu la chance de côtoyer des ingénieurs et de voir l'importance des bus de communication dans un environnement industriel réel.

Ensuite, nous avons découvert et assimilé le protocole Modbus TCP/IP en montrant toutes ses fonctionnalités et en faisant une analyse de chaque partie caractérisant ce protocole.

Nous avons, à l'aide de solutions open source, implémenté et conçu une application spécifique exploitant toutes les fonctionnalités du Modbus TCP/IP. Nous avons aussi réalisé un IO device fonctionnel pour les tests.

Enfin nous avons assuré le bon fonctionnement de notre application implémentée en captant toutes les données transférées et nous avons contrôlé l'efficacité de nos méthodes.

Dans l'avenir notre projet pourrait être développé pour son utilisation dans un réseau IOT ainsi que la réalisation d'un produit fini pour une éventuelle utilisation dans le domaine industriel

Annexe A : code des fonction Modbus

Définition du code de fonction publique

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Figure 1: Code des fonctions [2]

Annexe B : DP83849IFVS

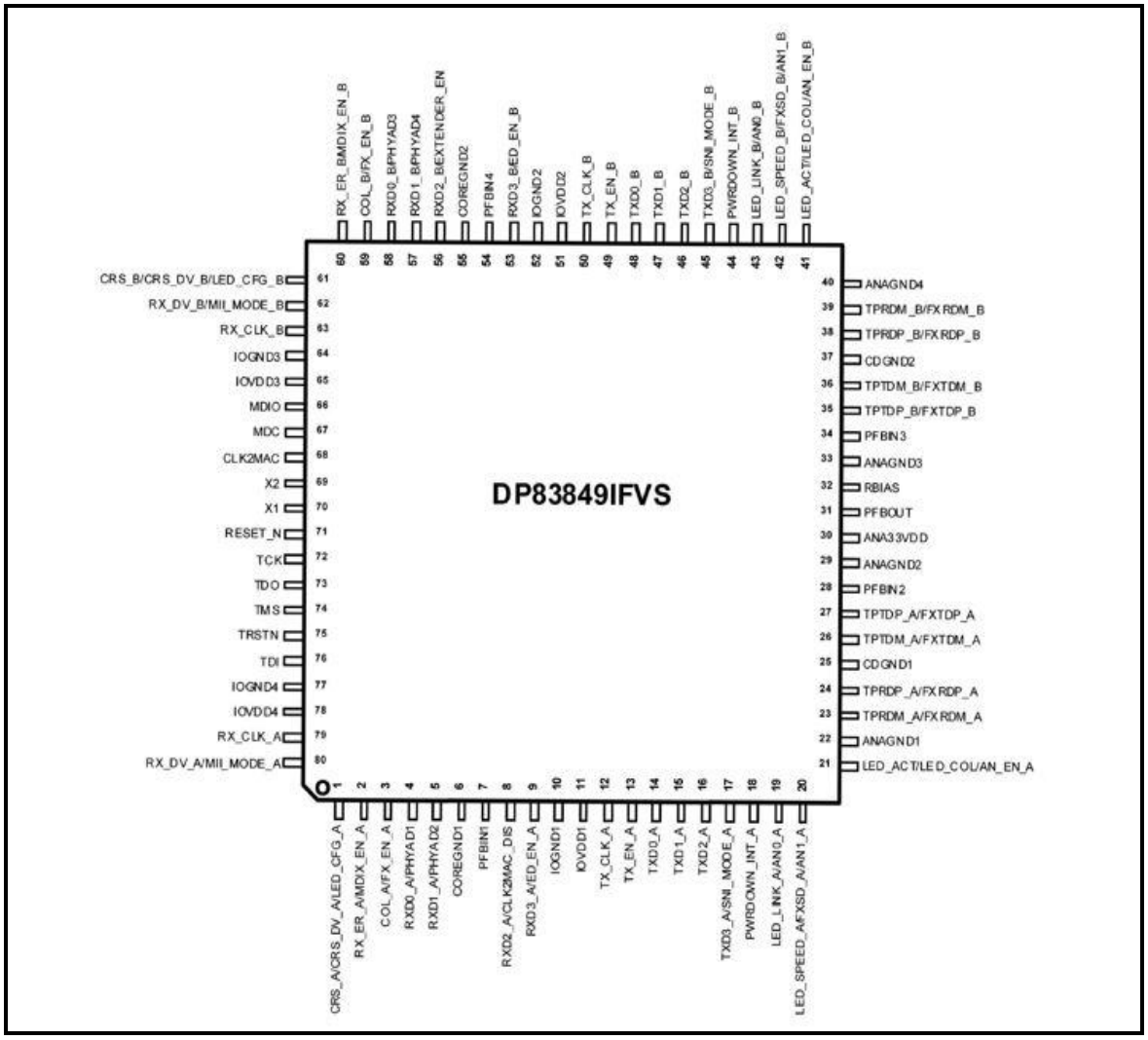

figure 1: Diagramme de connexion [14].

Caractéristiques du DP83849IFVS/NOPB

- **Interface de gestion série**

Signal name	Type	Pin#	Description
MDC	I	67	**MANAGEMENT DATA CLOCK:** Synchronous clock to the MDIO management data input/output serial interface which may be asynchronous to transmit and receive clocks. The maximum clock rate is 25 MHz with no minimum clock rate.
MDIO	I/O	66	**MANAGEMENT DATA I/O:** Bi-directional management instruction/data signal that may be sourced by the station management entity or the PHY. This pin requires a 1.5 kΩ pullup resistor

Tableau 1: Interface de gestion série.

- **Interface MAC Data**

Signal name	Type	Pin #	Description
TX_CLK_A TX_CLK_B	O	12 50	**MII TRANSMIT CLOCK:** 25 MHz Transmit clock output in 100 Mb/s mode or 2.5 MHz in 10 Mb/s mode derived from the 25 MHz reference clock. Unused in RMII mode. The device uses the X1 reference clock input as the 50 MHz reference for both transmit and receive. **SNI TRANSMIT CLOCK:** 10 MHz Transmit clock output in 10 Mb SNI mode. The MAC should source TX_EN and TXD_0 using this clock.

TX_EN_A TX_EN_B	I	13 49	**MII TRANSMIT ENABLE:** Active high input indicates the presence of valid data inputs on TXD[3:0]. **RMII TRANSMIT ENABLE:** Active high input indicates the presence of valid data on TXD[1:0]. SNI TRANSMIT ENABLE: Active high input indicates the presence of valid data on TXD_0.
TXD[3:0]_A TXD[3:0]_B	I	17, 16,15 ,14,4 5,46, 47,48	**MII TRANSMIT DATA:** Transmit data MII input pins, TXD[3:0], that accept data synchronous to the TX_CLK (2.5 MHz in 10 Mb/s mode or 25 MHz in 100 Mb/s mode). **RMII TRANSMIT DATA:** Transmit data RMII input pins, TXD[1:0], that accept data synchronous to the 50 MHz reference clock. **SNI TRANSMIT DATA:** Transmit data SNI input pin, TXD_0, that accept data synchronous to the TX_CLK (10 MHz in 10 Mb/s SNI mode)
RX_CLK_A RX_CLK_B	O	79 63	**MII RECEIVE CLOCK:** Provides the 25 MHz recovered receive clocks for 100 Mb/s mode and 2.5 MHz for 10 Mb/s mode. Unused in RMII mode. The device uses the X1 reference clock input as the 50 MHz reference for both transmit and receive. **SNI RECEIVE CLOCK:** Provides the 10 MHz recovered receive clocks for 10 Mb/s SNI mode
RX_DV_A RX_DV_B	O	80 62	**MII RECEIVE DATA VALID:** Asserted high to indicate that valid data is present on the corresponding RXD[3:0]. **RMII RECEIVE DATA VALID:** Asserted high to indicate that valid data is present on the corresponding RXD[1:0]. This

			signal is not required in RMII mode, since CRS_DV includes the RX_DV signal, but is provided to allow simpler recovery of the Receive data. This pin is not used in SNI mode
RX_ER_A RX_ER_B	O	2 60	**MII RECEIVE ERROR:** Asserted high synchronously to RX_CLK to indicate that an invalid symbol has been detected within a received packet in 100 Mb/s mode. **RMII RECEIVE ERROR:** Asserted high synchronously to X1 whenever an invalid symbol is detected, and CRS_DV is asserted in 100 Mb/s mode. This pin is also asserted on detection of a False Carrier event. This pin is not required to be used by a MAC in RMII mode, since the Phy is required to corrupt data on a receive error. This pin is not used in SNI mode
RXD[3:0]_A RXD[3:0]_B	O	9,8,5,4,53,56,57,58	**MII RECEIVE DATA:** Nibble wide receive data signals driven synchronously to the RX_CLK, 25 MHz for 100 Mb/s mode, 2.5 MHz for 10 Mb/s mode). RXD[3:0] signals contain valid data when RX_DV is asserted. **RMII RECEIVE DATA:** 2-bits receive data signals, RXD[1:0], driven synchronously to the X1 clock, 50 MHz. **SNI RECEIVE DATA:** Receive data signal, RXD_0, driven synchronously to the RX_CLK. RXD_0 contains valid data when CRS is asserted. RXD[3:1] are not used in this mode.
CRS_A/CRS_DV_A CRS_B/CR	O	1 61	**MII CARRIER SENSE:** Asserted high to indicate the receive medium is non-idle. **RMII CARRIER SENSE/RECEIVE DATA VALID:** This signal combines the RMII Carrier and Receive Data Valid

S_DV_B			indications. For a detailed description of this signal, see the RMII Specification. **SNI CARRIER SENSE:** Asserted high to indicate the receive medium is non-idle. It is used to frame valid receive data on the RXD_0 signal.

Tableau 2: Interface MAC Data

- **Interface d'horloge**

Signal name	Type	Pin#	Description
X1	I	70	**CRYSTAL/OSCILLATOR INPUT:** This pin is the primary clock reference input for the DP83849IF and must be connected to a 25 MHz 0.005% (±50 ppm) clock source. The DP83849IF supports either an external crystal resonator connected across pins X1 and X2, or an external CMOS-level oscillator source connected to pin X1 only. **RMII REFERENCE CLOCK:** This pin is the primary clock reference input for the RMII mode and must be connected to a 50 MHz 0.005% (±50 ppm) CMOS-level oscillator source
X2	O	69	**CRYSTAL OUTPUT:** This pin is the primary clock reference output to connect to an external 25 MHz crystal resonator device. This pin must be left unconnected if an external CMOS oscillator clock source is used.
CLK2MAC	O	68	**CLOCK TO MAC:** In MII mode, this pin provides a 25 MHz clock output to the system. In RMII mode, this pin provides a 50 MHz clock output to the system. This allows other

			devices to use the reference clock from the DP83849IF without requiring additional clock sources. If the system does not require the CLK2MAC signal, the CLK2MAC output should be disabled via the CLK2MAC disable strap.

Tableau 3: Interface d'horloge.

- **Interface LED**

Signal name	Type	Pin#	Description
LED_LINK_A LED_LINK_B	I/O	19 43	**LINK LED:** In Mode 1, this pin indicates the status of the LINK. The LED will be ON when Link is good. **LINK/ACT LED:** In Mode 2 and Mode 3, this pin indicates transmit and receive activity in addition to the status of the Link. The LED will be ON when Link is good. It will blink when the transmitter or receiver is active.
LED_SPEED_A LED_SPEED_B	I/O	20 42	**SPEED LED:** The LED is ON when device is in 100 Mb/s and OFF when in 10 Mb/s. Functionality of this LED is independent of mode selected.
LED_ACT/LED_COL_A LED_ACT/LED_COL_B	I/O	21 41	**ACTIVITY LED:** In Mode 1, this pin is the Activity LED which is ON when activity is present on either Transmit or Receive. **COLLISION/DUPLEX LED:** In Mode 2, this pin by default indicates Collision detection. For Mode 3, this LED output may be programmed to indicate

			Full Duplex status instead of Collision.

Tableau 4: Interface LED.

- **Interface JTAG**

Signal name	Type	Pin#	Description
TCK	I, PU	72	**TEST CLOCK :** This pin has a weak internal pullup
TDO	O	73	**TEST OUTPUT**
TMS	I, PU	74	**TEST MODE SELECT :** This pin has a weak internal pullup
TRSTN	I, PU	75	**TEST RESET Active low test reset :** This pin has a weak internal pullup
TDI	I, PU	76	**TEST DATA INPUT:** This pin has a weak internal pullup

Tableau 5: Interface JTAG

- **Réinitialiser et éteindre**

Signal name	Type	Pin#	Description
RESET_N	I, PU	71	**RESET:** Active Low input that initializes or re-initializes the DP83849IF. Asserting this pin low for

			at least 1 µs will force a reset process to occur. All internal registers will re-initialize to their default states as specified for each bit in the Register Block section. All strap options are re-initialized as well.
PWRDOWN_INT_A PWRDOWN_INT_B	I, PU	18 44	**POWER DOWN:** The pin is an active low input in this mode and should be asserted low to put the device in a Power Down mode. **INTERRUPT:** The pin is an open drain output in this mode and will be asserted low when an interrupt condition occurs. Although the pin has a weak internal pull-up, some applications may require an external pull-up resistor. Register access is required for the pin to be used as an interrupt mechanism. See Section 8.6.2 for more details on the interrupt mechanisms

Tableau 6: Réinitialiser et éteindre.

- **Sélection du LED mode**

Mode	LED_CFG[1]	LED_CFG[0]	LED_LINK	LED_SPEED	LED_ACT/LED_COL
1	don't care	1	ON for Good Link OFF for No Link	ON in 100 Mb/s OFF in 10 Mb/s	ON for Activity OFF for No Activity
2	0	0	ON for Good Link BLINK for	ON in 100 Mb/s OFF in 10	ON for Collision OFF for No Collision

			Activity	Mb/s	
3	1	0	ON for Good Link BLINK for Activity	ON in 100 Mb/s OFF in 10 Mb/s	ON for Full Duplex OFF for Half Duplex

Tableau 7: Sélection du LED mode.

- **Interface PMD 10/100 Mb/s**

Signal name	**Type**	**Pin#**	**Description**
TPTDM_A/FXTDM_A TPTDP_A/FXTDP_A TPTDM_B/FXTDM_B TPTDP_B/FXTDP_B	I/O	26 27 36 35	**10BASE-T or 100BASE-TX or 100BASE-FX Transmit Data** In 10BASE-T or 100BASE-TX: Differential common driver transmit output (PMD Output Pair). These differential outputs are automatically configured to either 10BASE-T or 100BASE-TX signaling. In Auto-MDIX mode of operation, this pair can be used as the Receive Input pair. In 100BASE-FX mode, this pair becomes the 100BASE-FX Transmit pair. These pins require 3.3V bias for operation.

TPRDM_A/FXRDM_A TPRDP_A/FXRDP_A TPRDM_B/FXRDM_B TPRDP_B/FXRDP_B	I/O	23 24 39 38	**10BASE-T or 100BASE-TX or 100BASE-FX Receive Data** In 10BASE-T or 100BASE-TX: Differential receive input (PMD Input Pair). These differential inputs are automatically configured to accept either 100BASE-TX or 10BASE-T signaling. In Auto-MDIX mode of operation, this pair can be used as the Transmit Output pair. In 100BASE-FX mode, this pair becomes the 100BASE-FX Receive pair. These pins require 3.3V bias for operation
FXSD_A(LED_SPEED_A/ANA) FXSD_B(LED_SPEED_B/AN1_B)	I	20 42	**FX Signal Detect:** This pin provides the Signal Detect input for 100BASE-FX mode.

Tableau 8: Interface PMD 10/100 Mb/s.

- **Broches de connexion spéciales**

Signal name	Type	Pin#	Description
RBIAS	I	32	**Bias Resistor Connection:** A 4.87 kΩ 1% resistor should be connected from RBIAS to GND
PFBOUT	O	31	**Power Feedback Output:** Parallel caps, 10µF and 0.1µF, should be placed close to the PFBOUT. Connect this pin to PFBIN1 (pin 13), PFBIN2 (pin 27), PFBIN3 (pin 35), PFBIN4

			(pin 49). See figure 2.38 for proper placement pin.
PFBIN1 PFBIN2 PFBIN3 PFBIN4	I	7 28 34 54	**Power Feedback Input:** These pins are fed with power from PFBOUT pin. A small capacitor of 0.1µF should be connected close to each pin.

Tableau 9: Broches de connexion spéciales.

Signal name	Type	Pin#	Description
PHYAD1 (RXD0_A) PHYAD2 (RXD1_A) PHYAD3 (RXD0_B) PHYAD4 (RXD1_B)	S, O, PD S, O, PD S, O, PD S, O, PD	4 5 58 57	PHY ADDRESS [4:1]: The DP83849IF provides four PHY address pins, the state of which are latched into the PHYCTRL register at system Hardware-Reset. Phy Address[0] selects between ports A and B. The DP83849IF supports PHY Address strapping for Port A even values 0 (<0000_0>) through 30 (<1111_0>). Port B will be strapped to odd values 1 (<0000_1>) through 31 (<1111_1>). PHYAD[4:1] pins have weak internal pull-down resistors.
FX_EN_A (COL_A) FX_EN_B (COL_B) AN_EN(LED_ACT/LED_COL_A) AN1_A (LED_SPEED_A) AN0_A (LED_LINK_A) AN_EN(LED_ACT/LED_COL_B)	S, I, PD S, O, PU	3 59 21 20 19 41	**FX ENABLE:** Default is to disable 100BASE-FX (Fiber) mode. This strapping option enables 100BASE-FX. An external pull-up will enable 100BASE-FX mode. **Auto-Negotiation Enable:** When high, this enables Auto-Negotiation with the capability set by AN0 and AN1 pins. When low, this puts the part into Forced Mode with the capability set by AN0 and AN1 pins. **AN0 / AN1:** These input pins control the forced or advertised operating mode of the

AN1_B (LED_SPEED_B)

AN0_B (LED_LINK_B)

42

43

DP83849IF according to the following table. The value on these pins is set by connecting the input pins to GND (0) or VCC (1) through 2.2 kΩ resistors. **These pins should NEVER be connected directly to GND or VCC.**

Fiber Mode Duplex Selection: If Fiber mode is strapped using the FX_EN pin, the AN0 strap value is used to select Half or Full Duplex. AN_EN and AN1are ignored if FX_EN is asserted, since Fiber mode is 100Mb only and does not support Auto-Negotiation.

The value set at this input is latched into the DP83849IF at Hardware-Reset. The float/pull-down status of these pins are latched into the Basic Mode Control Register and the Auto_Negotiation Advertisement Register during Hardware-Reset. The default is 0111 since the FX_EN pin has an internal pull-down and the AutoNegotiation pins have internal pull-ups.

FX_EN	AN_EN	AN1	AN0	Forced Mode
0	0	0	0	10BASE-T, Half-Duplex
0	0	0	1	10BASE-T, Full-Duplex
0	0	1	0	100BASE-TX, Half-Duplex
0	0	1	1	100BASE-TX, Full-Duplex
1	X	X	0	100BASE-FX, Half-

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td></td><td></td><td>Duplex</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>1</td><td>100BASE-FX, Full-Duplex</td></tr>
<tr><td>FX_EN</td><td>AN_EN</td><td>AN1</td><td>AN0</td><td>Advertised Mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>10BASE-T, Half/Full-Duplex</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>100BASE-TX, Half/Full-Duplex</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>10BASE-T, Half-Duplex, 100BASE-TX, Half-Duplex</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>10BASE-T, Half/Full-Duplex, 100BASE-TX, Half/Full-Duplex</td></tr>
</table>
</td></tr>
<tr><td>MII_MODE_A (RX_DV_A)
SNI_MODE_A (TXD3_A)
MII_MODE_B (RX_DV_B)
SNI_MODE_B (TXD3_B)</td><td>S, O, PD</td><td>80
17
62
45</td><td>MII MODE SELECT: This strapping option pair determines the operating mode of the MAC Data Interface. Default operation (No pull-ups) will enable normal MII Mode of operation. Strapping MII_MODE high will cause the device to be in RMII or SNI modes of operation, determined by the status of the SNI_MODE strap. Since the pins include internal pull-downs, the default values are 0. Both MAC Data Interfaces must have their RMII Mode settings the same, i.e. both in RMII mode or both not in RMII mode. The following table details the</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>configurations:

<table><tr><th>MII_MODE</th><th>SNI_MODE</th><th>MAC Interface Mode</th></tr><tr><td>0</td><td>X</td><td>MII Mode</td></tr><tr><td>1</td><td>0</td><td>RMII Mode</td></tr><tr><td>1</td><td>1</td><td>10 Mb SNI Mode</td></tr></table></td></tr>
<tr><td>LED_CFG_A(CRS_A/CRS_DV_A)

LED_CFG_B(CRS_B/CRS_DV_B)</td><td>S, O, PU</td><td>1

61</td><td>LED CONFIGURATION: This strapping option determines the mode of operation of the LED pins. Default is Mode 1. Mode 1 and Mode 2 can be controlled via the strap option.

All modes are configurable via register access.</td></tr>
<tr><td>MDIX_EN_A (RX_ER_A)

MDIX_EN_B (RX_ER_B)</td><td>S, O, PU</td><td>2

60</td><td>MDIX ENABLE: Default is to enable MDIX. This strapping option disables Auto-MDIX

An external pull-down will disable Auto-MDIX mode.</td></tr>
<tr><td>ED_EN_A (RXD3_A)

ED_EN_B (RXD3_B)</td><td>S, O, PD</td><td>9

53</td><td>Energy Detect ENABLE: Default is to disable Energy Detect mode. This strapping option enables Energy Detect mode for the port. In Energy Detect mode, the device will initially be in a low-power state until detecting activity on the wire. An external pull-up will enable Energy Detect mode.</td></tr>
<tr><td>CLK2MAC_DIS (RXD2_A)</td><td>S, O, PD</td><td>8</td><td>Clock to MAC Disable: This strapping option disables (floats) the CLK2MAC pin. Default is to enable CLK2MAC output. An external pullup will disable (float) the CLK2MAC pin. If the system does not require the CLK2MAC signal, the CLK2MAC output should be disabled via this</td></tr>
</table>

			strap option
EXTENDER_EN (RXD2_B)	S, O, PD	56	**Extender Mode Enable:** This strapping option enables Extender Mode for both ports. When enabled, the strap will enable Single Clock MII TX and RX modes unless RMII Mode is also strapped. SNI Mode cannot be strapped if Extender Mode is strapped.

Tableau 10: Option sangle

Bibliographie

[1] Acromag Inc : 'INTRODUCTION TO MODBUS TCP/IP', https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf

[2] Modbus.org : 'MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3' , https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[3] Wikipédia : 'Alimentation électrique par câble Ethernet', https://fr.wikipedia.org/wiki/Alimentation_%C3%A9lectrique_par_c%C3%A2ble_Ethernet

[4] Mouser , https://www.mouser.fr/

[5] Mouser, https://www.mouser.fr/ProductDetail/STMicroelectronics/STM32F207IGH6?qs=ZjM2s42DDybplwSgHpDsQw%3D%3D

[6] Mouser ,https://www.mouser.fr/ProductDetail/Texas-Instruments/DP83849IFVS-NOPB?qs=%2Fha2pyFaduh0lMBxOii8P69NZ0ezw3JlEBD40ktLt1p5Gkj%252BSqHCXA%3D%3D

[7] Mouser, https://www.mouser.fr/ProductDetail/Monolithic-Power-Systems-MPS/MP2451DT-LF-Z?qs=rC7bBWoQAAn%252B7%252BeP%252Btpamw%3D%3D

[8] Mouser , https://www.mouser.fr/ProductDetail/Bel-Magnetic-Solutions/0826-1X1T-KL-F?qs=%2Fha2pyFadujy9UBrk%252BQdopq78tSL6ERsBmBfr57yZj7SbWpN5r2nHA%3D%3D

[9] Mouser, https://www.mouser.fr/datasheet/2/389/cd00237391-1796869.pdf

[10] ST, https://www.st.com/content/ccc/resource/technical/document/application_note/c6/eb/5e/11/e3/69/43/eb/CD00221665.pdf/files/CD00221665.pdf/jcr:content/translations/en.CD00221665.pdf

[11] ST, https://www.st.com/resource/en/reference_manual/cd00225773-stm32f205xx-stm32f207xx-stm32f215xx-and-stm32f217xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

[12] analog, https://www.analog.com/en/technical-articles/i2c-cabling.html

[13] Mouser , https://www.mouser.fr/datasheet/2/277/MP2451-1384199.pdf

[14] ti, https://www.ti.com/lit/ds/symlink/dp83849if.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1630967536694&ref_url=https%253A%252F%252Fwww.mouser.fr%252F

[15] Ti, https://www.ti.com/lit/ds/symlink/dp83849id.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1631400455586&ref_url=https%253A%252F%252Fwww.mouser.fr%252F

[16] Mouser, https://www.mouser.fr/datasheet/2/643/dr-MAG-0826-1X1-TKL-F-1670779.pdf

[17] https://www.rfc-editor.org/rfc/pdfrfc/rfc951.txt.pdf

[18] jlcpcb, https://jlcpcb.com/capabilities/Capabilities

[19] youtube, https://www.youtube.com/watch?v=ySuUZEjARPY&t=1253s

[20] jlcpcb, https://cart.jlcpcb.com/quote?orderType=1

[21] asong , https://server4.eca.ir/eshop/AHT10/Aosong_AHT10_en_draft_0c.pdf