E. N. S. T
المدرسة الوطنية العليا للتكنولوجيا
Ecole nationale supérieure de technologie

الجمهورية الجزائرية الديمقراطية الشعبية
Populaire et Démocratique Algérienne République
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Département : Génie électrique Et Informatique Industrielle

GEii
Génie Électrique et Infomatique Industrielle

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en

Électronique

Spécialité :

Systèmes Embarqués

Thème

Enregistreur de données connecté via un modem radio LoRa

Réalisé par

BELARBI Naouel

Les membres de Jury :

Président	M. BEGHAMI Sami	M.A.A	ENST
Promoteur	Mme. SAADI Hadjer	M.C.A	ENSTICP
Examinateur	Mme. HABANI Lamia	M.A.A	ENST
Examinateur	M. BOUSSOUFA Ahmed	M.A.A	ENST
Co-Promoteur	M. ZAOUI Abdelhalim	PROF	ENST

Année universitaire 2022-2023

Remerciement

Remerciement

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué à la réalisation de ce mémoire de projet de fin d'études. Le succès de ce travail n'aurait pas été possible sans leur soutien, leurs conseils et leur encouragement.

Tout d'abord, je tiens à remercier chaleureusement mes superviseurs Monsieur BELLOULA, Madame SAADI et Monsieur ZAOUI, pour leur expertise, leur disponibilité et leur dévouement tout au long de ce projet. Leurs précieux conseils ont été d'une aide inestimable, qui ont amélioré ce travail de plus en plus.

Je tiens également à exprimer ma gratitude envers le comité de soutenance, composé de Monsieur BEGHAMI, Madame HABANI et Monsieur BOUSSOUFA pour avoir pris le temps d'examiner mon travail.

Mes remerciements s'étendent également à ma famille et à mes amies, qui m'ont soutenu(e) de manière inconditionnelle tout au long de mes études. Leur encouragement et leur compréhension ont été essentiels pour surmonter les défis de ce parcours académique.

Enfin, je remercie tous les enseignants de l'ENST, qui ont partagé leurs connaissances et leurs expériences avec moi, contribuant ainsi à mon développement académique.

Ce mémoire de projet de fin d'études a été une expérience enrichissante, et je suis reconnaissant(e) envers chacun d'entre vous pour avoir rendu cela possible.

Table des matières

Introduction Générale 1

Chapitre I : Généralités sur les DataLoggers

I.1. Introduction 3
I.2. Définition de DataLogger 3
I.3. Schéma de principe et composition de base d'un DataLogger 4
I.3.1. Capteurs 4
I.3.2. Conditionnement du signal 4
I.3.3. Convertisseur A/D 5
I.3.4. Unité de traitement 5
I.3.5. Unité de stockage 5
I.4. Caractéristiques des DataLoggers 5
I.5. Types des entrées des DataLoggers 6
I.6. Fonctionnalités annexes et options points fort des DataLoggers 6
I.7. Types des DataLoggers 7
I.1. Domaine d'application des DataLoggers 8
I.8. DataLoggers versus les systèmes DAQ classiques 9
I.10. Interface de communication 10
I.10.1. Technologie de communication LoRa 10
I.10.2. Différence entre LoRa et LoRaWAN 10
I.11. Conclusion 11

Chapitre II : Conception d'un DataLogger Personnalisé

II.1. Introduction 12
II.2. Objectif du travail 12
II.3. Description générale du système 12
II.4. Description matérielle de Datalogger 13
II.4.1. Unité de traitement 14
II.4.1.1. Plateforme de développement matérielle STM32F407VGT6 16
II.4.1.2. Sonde de programmation ST-LINK 17
II.4.2. Unité de stockage 17
II.4.3. Unité d'acquisition de données 18
II.4.3.1. Capteurs 18
II.4.3.2. Interfaces d'entrées et conditionnement du signal 20
II.4.4.1. Interface USB 30
II.4.5. Alimentation 33
II.5. Description matérielle de l'IHM (Interface Homme Machine) 34
II.5.1. Raspberry Pi 3 34
II.5.2. Schéma fonctionnel de l'IHM 35
III.6. Conclusion 36

Chapitre III : Implémentation et Mise en Œuvre du Système de DataLogger

III.1.Introduction 37
III.2. Description des logiciels utilisés et conçus 37
III.2.1. Partie datalogger 37
III.2.1.1. STM32CubeMX 37
III.2.1.2. CubeIDE 38
III.2.2. Partie IHM 38
III.2.2.1. Raspberry Pi Imager 38
III.2.2.2. CMD Windows 39
III.2.2.3. VNC viewer 39
III.2.2.4. Node-red 39
III.3. Principe de fonctionnement du DataLogger 39
III.4. Etapes de réalisation du DataLogger 42
III.4.1. Configuration du MCU sur CubeMX 42
III.4.2. Partie programmation des différents modules 54
III.4.2.1. Partie de stockage 55
III.4.2.2. Partie de synchronisation et acquisition 58
III.4.2.3. Partie de transmission 61
III.5. Construction de l'IHM 64
III.5.1. Préparation de la carte RaspberryPi 64
III.5.2. Principe de fonctionnement de l'IHM 66
III.5.3.1. Différents nœuds utilisés 68
III.5.2. Programmation de module LoRa 70
III.5.3. Résultat de l'IHM 71
III.6. Réalisation de PCB et du boitier du DataLogger 71
III.6.1. PCB 71
III.6.2. Conception et réalisation d'un boitier pour le DataLogger 73
III.7. Test et résultat final 73
III.7.1. Capteurs utilisés pour le test 73
III.7.2.Visualisation des résultats sur l'IHM 74
III.8. Conclusion 80
Conclusion Générale 81

Annexes

Annexe I

A.I.1. Nombre de canaux des DataLoggers AI-1
A.I.2. Calcul de la période d'enregistrement d'un DataLogger AI-1
A.I.3. Connexion des modules d'extension AI-2
A.I.4. Connecteur borne à vis AI-2

Annexe II

A.II.1. Principe de la modulation LoRa AII-1
A.II.1.1. Propriétés spéciales AII-1
A.II.1.3. Limites AII-2

A.II.2. Schéma bloc de l'alimentation de la carte STM32F407VGT6 ..... AII-2

Annexe III

A.III.1. Configuration de DMA AIII-1
A.III.2. La fonction de conversion de date/temps en forme UNIX AIII-2
A.III.3. Configuration du date/temps AIII-3
A.III.4. Configuration des taux d'échantillonnage : AIII-4
A.III.5. Fonctions de déclenchement de Timer et de DMA AIII-4
A.III.6. Fonctions callbacks AIII-4
A.III.7. Le fichier I2Cexemple AIII-5
A.III.8. Configuration et programmation de la carte STM32Nucleo64 ..... AIII-6

Liste des figures

Chapitre I : Généralités sur les DataLoggers

Figure I.1. Exemple illustratif de différents types et formes de DataLoggers [3]..... 3
Figure I.2. Eléments de base d'un data-logger types et formes de DataLoggers [3] 3
Figure I.2. Eléments de base d'un data-logger..... 4

Chapitre II : Conception d'un DataLogger Personnalisé

Figure II.1. Schéma descriptif du système conçu du datalogger---------- 13
Figure II.2. Schéma bloc du datalogger ---------- 13
Figure II.3. Schéma bloc du microcontrôleur STM32F407VGT6 [1]---------- 14
Figure II.4. Illustration de la carte STM32F407VGT6 [2] ---------- 16
Figure.II.5. Illustration du module ST-LINK V2 et ses pins [3]---------- 17
Figure II.6. Schéma électrique de l'interface SDIO SD card [5] ---------- 17
Figure.II.7. Pins de la carte SD [6] ---------- 18
Figure II.8. Eléments constituants de bloc d'acquisition des données ---------- 18
Figure II.9. TPC ½-IP68 [7] ---------- 19
Figure II.10. LM35 [8]---------- 19
Figure II.11. MQ135 [9]---------- 19
Figure II.12. ACS712[10] ---------- 19
Figure II.13. capteur d'humidité de sol [11] ---------- 19
Figure II.14. TH02 [12]---------- 19
Figure II.15. sonde de température d'ambiante RS485 [13]---------- 20
Figure II.16. capteur de température avec tige et entête avec protocole CAN [14]---------- 20
Figure II.17. capteur de debit d'eau YF-S401 [15] ---------- 20
Figure II.18. Schéma de câblage des capteurs avec les entrées analogiques de la carte STM32F407VGT6---------- 21
Figure II.19. Exemple d'utilisation d'un transmetteur de boule 4-20mA avec le capteur de température PT100 et son branchement à l'entrée 4-20mA de datalogger [16] [17] ---------- 22
Figure II.20. Schéma électronique de module de conversion 4-20Ma à tension [20]---------- 23
Figure II.21. Schéma électrique du calibrage ---------- 24
Figure II.22. Trame de données d'une transmission série UART [22] ---------- 25
Figure II.23. Emplacement de plusieurs nœuds sur le bus RS-485---------- 25
Figure II.24. Schéma électrique de Convertisseur de TTL to RS-485 [24] ---------- 26
Figure II.25. Schéma de câblage des capteurs au bus RS485 et de module TTL /RS485 à la carte ---------- 26
Figure II.27. Schéma de câblage de la carte STM32 avec un capteur sur le bus CAN --------- 28
Figure II.26. Schéma électrique de transmetteur CAN MCP2551 [26] ---------- 28
Figure II.28. Principe fondamental d'un transfert de données avec I2C ---------- 29
Figure II.29. Trame de données I2C [27] ---------- 29
Figure II.30. Schéma de câblage de la carte STM32 avec 2 capteurs utilisant le bus I2C ----- 29

Figure II.31. Pins du câble USB [28] et sschéma électrique de l'interface USB de la carte STM32 [29]- ---------- 30
Figure II.32. Illustration de module Ra-02 et d'antenne IPEX employé avec lui [31] ---------- 31
Figure II.33. Schéma électrique de module Lora [32] ---------- 32
Figure II.34. Schéma simplifie de l'alimentation électrique de DataLogger---------- 33
Figure II.35. Estimation de la consommation énergétique de DataLogger---------- 34
Figure II.36. Raspberry Pi 3 et ses modules d'E/S [35] ---------- 35
Figure II.37. Schéma fonctionnel de l'IHM ---------- 36

Chapitre III : Implémentation et Mise en Œuvre du Système de DataLogger

Figure III.1. Illustration de l'interface du logiciel Raspberry Pi Imager---------- 38
Figure III.2. Organigramme de fonctionnement principal du DataLogger----------41
Figure III.3. Les pins sélectionnés sur le MCU ----------42
Figure III.4. Configuration de GPIO sur cubeMX----------45
Figure III.5. Paramètres de configuration de l'ADC1 sur cubeMX ----------44
Figure III.6. Paramètre de configuration de l'ADC2 sur cubeMX ----------46
Figure III.7. Arbre d'horloge du système----------47
Figure III.8. Configuration de l'I2C1 sur cubeMX----------48
Figure III.9. Configuration de SPI1 sur cubeMX ----------48
Figure III.10. Configuration de CAN1 sur cubeMX----------49
Figure III.11. Configuration del'UART1 sur cubeMX ----------49
Figure III.12. Configuration de USB OTG-FS sur cubeMX ----------50
Figure III.13. Configuration de SDIO sur cubeMX----------50
Figure III.14. Configuration de RTC sur cubeMX----------51
Figure III.15. Configuration du TIMER1 sur cubeMX ----------51
Figure III.16. Configuration du TIMER2 sur cubeMX ----------52
Figure III.17. Configuration du TIMER6 sur cubeMX ----------52
Figure III.18. Configuration de FATFS sur cubeMX ----------53
Figure III.19. Configuration de classe de l'USB sur cubeMX----------53
Figure III.20. Configuration de cubeMX pour génération de code sous cubeIDE ----------54
Figure III.21. Les fichiers de projets sur cubeIDE----------55
Figure III.22. Le fichier header SD.h ----------56
Figure III.23. Correction de fichier SDIO.c ----------56
Figure III.24. Résultat de test de la carte SD ----------57
Figure III.25. Programme d'initialisation de la carte SD ----------57
Figure III.26. Test de l'EEPROM ----------58
Figure III.27. Déclaration des variables nécessaire pour l'échantillonnage et l'acquisition des données----------58
Figure III.28. Organigramme de fonctionnement de processus de synchronisation d'acquisition et d'enregistrement des données----------60
Figure III.29. Code d'acquisition et d'enregistrement des données venus des différentes entrées ----------61
Figure III.30. Le programme utilisé pour la transmission et la réception utilisant LoRa ------63
Figure III.31. Configuration de l'OS sur Raspberry Pi Imager----------64

Figure III.32. VNC viewer ---- 65
Figure III.33. Bureau de Raspberry Pi ---- 65
Figure III.34. Node-red sur RaspberryPi ---- 66
Figure III.35. Logigramme de l'IHM ---- 66
Figure III.36. Flow d'acquisition ---- 67
Figure III.37. Flow de visualisation des données ---- 68
Figure III.38. Les composantes de l'IHM ---- 71
Figure III.39. Schéma de câblage ---- 72
Figure III.40. Illustration de PCB ---- 72
Figure III.41. Résultat de la conception de PCB ---- 72
Figure III.42. Résultat de boitier ---- 73
Figure III.43. Illustration de câblage ---- 74
Figure III.44. Affichage des valeurs acquis par les canaux analogiques sur l'IHM ---- 75
Figure III.45. Graphe d'évolution des valeurs acquises par les canaux analogiques ---- 76
Figure III.46. Affichage de sur des temps d'activation de buzzer acquis par la pine digitale sur l'IHM ---- 76
Figure III.47. Affichage des valeurs acquises par le canal7 (compteur d'impulsion) ---- 77
Figure III.48. Graphe d'évolution de débit d'eau dans le temps ---- 77
Figure III.49. Affichage des valeurs acquis de port I2C sur l'IHM ---- 78
Figure III.50. Graphe d'évolution de température et d'humidité dans le temps ---- 78
Figure III.51. Affichage des valeurs acquises de port série sur l'IHM ---- 79
Figure III.52. Graphe d'évolution de la température dans le temps acquise par les ports séries 79

Annexe I

Figure A.I.3.1. Connexion des modules d'extension ---- AI-2

Figure A.I.4.1. Photo des connecteurs borne à vis ---- AI-2

Annexe II

Figure A.II.1. Modulation LoRa ---- AII-1
Figure A.II.2. Sensibilité de signal LoRa aux bruits ---- AII-1
Figure A.II.3. Schéma bloc de l'alimentation de la carte STM32F407VGT6 ---- AII-2

Annexe III

Figure A.III.1. Paramètre de DMA de l'ADC1 sur cubeMX ---- AIII-1
Figure A.III.2. Paramètre de DMA de l'ADC2 sur cubeMX ---- AIII-1
Figure A.III. 3. Configuration DMA pour UART1 et UART2 ---- AIII-2
Figure A.III.4. Configuration DMA pour SDIO Tx et Rx sur cubeMX ---- AIII-2
Figure A.III.2.1 Le code de la fonction get_time() ---- AIII-3
Figure A.III.3.1. Code de configuration de la date et le temps ---- AIII-3
Figure A.III.4.1. Code de traitement de la réception des taux d'échantillonnage ---- AIII-4
Figure A.III.5.1. Code d''initialisation des TIMER et DMA ---- AIII-4
Figure A.III.6.1. Fonctions Callbacks ---- AIII-5
Figure A.III.8.1. Configuration de MCU F446RE sur cubeMX ---- AIII-6
Figure A.III.8.2. Illustration de câblage ---- AIII-8

Figure A.III.8.3. Résultat de réception des données par DataLogger ---------- AIII-8

Liste des tableaux

Chapitre II : Conception d'un DataLogger Personnalisé

Tableau I.1. Classification des DataLoggers---------- 7
Tableau I.2. Comparaison entre les DataLoggers et les autres les systèmes DAQ---------- 9

Chapitre II : Conception d'un DataLogger Personnalisé

Tableau II.1 : Classification des types des cartes SD [4]---------- 17
Tableau II.2 : Exemples de capteurs supportés par le DataLogger---------- 19
Tableau II.3 : Table de connectivité ---------- 21
Tableau II.4 : Differentes configurations de ce module ---------- 24
Tableau II.5 : Comparaison des standards de communication série [23] ---------- 25
Tableau II.6 : Connectivité des pins de module avec celles de la carte ---------- 27
Tableau II.7 : Spécification de module Ra-02 [30] ---------- 30
Tableau II.8 : Les pins de brochage du module Ra-02 à la carte STM32f407VGT6 ---------- 31
Tableau II.9: Pines de branchement du module LoRa Ra-02 avec la carte Raspberry---------- 34

Chapitre III : Implémentation et Mise en Œuvre du Système de DataLogger

Tableau III.1. Configuration du MCU ----------42

ADC Analogue to Digital Converter

A/N Analogique/Numérique

AC Alternatif Current

RS-232 Recommended standard 232

°C degré Celesius

CO2 Dioxyde de carbone

CPU Central Processing Unit

CSS Chirp Spread Spectrum

CR Coding Rate

DAS Data Acquisition System

DC Direct Current E/S Entrée/Sortie Hz Hertz

IHM Interface Homme Machine

ISM Industriel, Scientifique et Médical

IoT Internet Of Thing

Ko Kilooctet

LCD Liquid-Crystal Display

LoRa Long Range

LPWANs Low Power Wide Area Network

LoRaWAN Long Range Wide Area Network

mA milliAmpère

MAC Medium Access Control

MCU Microcontroleur

M2M Machine to Machine

Mo Mégactet

PRT Platinum Resistance Thermometers

PC Peersonal Computer

sub-GHz sub-GigaHertz SF Spreading Factor USB Universal Serial Bus

UART Universal Asynchrone Receiver Transmitter

V Volts

Wi-Fi Wireless Fidelity

XXe vingtième

Introduction Générale

Introduction Générale

Dans un monde de plus en plus connecté, l'Internet des Objets (IoT) joue un rôle de plus en plus prépondérant en transformant la manière dont nous interagissons avec notre environnement. L'IoT ouvre la porte à une multitude d'applications innovantes, allant de la gestion intelligente de l'énergie à la surveillance environnementale en passant par la santé etc.

Au cœur de cette révolution technologique se trouve la nécessité de collecter, stocker et transmettre efficacement des données provenant de capteurs dispersés sur de vastes territoires.

La collecte de données est un outil fondamental pour la gestion, la prise de décision, l'innovation et la compréhension du monde qui nous entoure. Elle permet de contrôler, d'analyser et d'agir en fonction des informations objectives, ce qui est essentiel pour l'efficacité, la sécurité, la conformité aux réglementations et l'amélioration continue dans de nombreux domaines.

Cette importance a créé un impératif pour les entreprises et les chercheurs, qui cherchent constamment à recueillir des informations précises et pertinentes pour maintenir leur compétitivité et prendre des décisions éclairées.

C'est dans ce contexte que s'inscrit notre projet de fin d'études, qui se vise la "Conception et Réalisation d'un Datalogger Connecté via LoRa.

Le domaine de l'IoT est en constante évolution, et les solutions de communication sans fil jouent un rôle essentiel dans la transmission de données depuis des capteurs distants vers des serveurs de collecte et d'analyse. Parmi ces technologies, LoRa (Long Range) émerge comme une option particulièrement prometteuse. LoRa offre une combinaison unique de longue portée, de faible consommation d'énergie et de coûts de déploiement abordables, ce qui en fait un choix idéal pour les applications d'IoT nécessitant une connectivité à grande échelle.

Le principal objectif de cette étude est de concevoir, développer et évaluer un DataLogger basé sur la technologie LoRa, capable de collecter des données à partir de capteurs reliés à ses déférentes entrées. Les sous objectifs sont :

- ✓ La conception matérielle et logicielle d'un DataLogger avec plusieurs entrées.
- ✓ Le développement d'une solution de communication avec le DataLogger via LoRa à partir d'une interface.

Cette étude s'appuiera sur une combinaison de connaissances en électronique, en informatique et en réseaux de communication pour créer une solution intégrée et efficace.

Ce mémoire est organisé en trois chapitres qui guideront le lecteur à travers l'ensemble du processus de conception et de réalisation du notre solution.

Le premier chapitre contient une étude théorique sur les DataLoggers, ses types et domaine d'application etc.

Dans le deuxième chapitre on vous présente l'aspect matériel de notre projet ainsi les différents protocoles de communication utilisés pour se communiquer avec les capteurs.

Le dernier chapitre contient l'aspect logiciel de DataLogger et de l'IHM (Interface Human Machine) ainsi les résultats obtenus.

Chapitre 1

Généralités sur les Data-Loggers

Chapitre 1
Généralités sur les Data-Loggers

I.1. Introduction

Les enregistreurs de données existent depuis des décennies, les premiers data-logger ont été développés pour des applications industrielles et militaires au milieu du XXe siècle. Ils ont été volumineux et encombrants, et utilisaient généralement des magnétophones analogiques pour stocker les données. Actuellement, les enregistreurs de données sont devenant moins encombrés et offrant une grande précision ainsi une souplesse d'utilisation. Ils sont utilisés largement dans différents domaines tels que la surveillance environnementale, l'industrie manufacturière,la recherche scientifique, la surveillance de la santé, etc.

Dans ce chapitre, nous allons donner certains détails sur les enregistreurs de données, leurs définitions, leurs caractéristiques, leurs principes de fonctionnements, les différents types d'enregistreursde données disponibles, ainsi que les applications courantes de cette technologie. Nous allons également parler sur la différence entre eux et les systèmes d'acquisitions classiques.

I.2. Définition de DataLogger

L'enregistreur de données (également appelé data logger), est un appareil électronique autonome faisant partie des outils d'acquisition des données DAQ (Data Acquisition), qui permet la collection, le stockage, le transfert et l'analyse des données sur une longue durée (pouvant couvrir plusieurs mois) [1,2].



Figure I.1. Exemple illustratif de différents types et formes de DataLoggers [3]

I.3. Schéma de principe et composition de base d'un data logger

Le data logger est un système d'acquisition de données numériques, et lorsqu'on parle d'un système DAQ (Data Acquisition) nous faisons référence au processus consistant à prendre des mesures de phénomènes physiques et à les enregistrer d'une certaine manière afin de les

analyser.

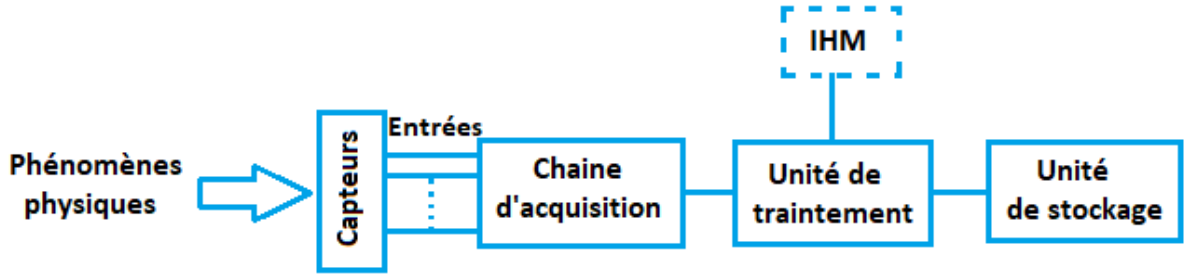


Figure I.3. Eléments de base d'un data-logger

Les enregistreurs de données se composent de quatre éléments essentiels qui forment l'ensemble de la chaîne de mesure des phénomènes physiques et leurs enregistrements cités dans l'ordre comme suit [2,4] :

I.3.1. Capteurs

Ils sont employés avec l'enregistreur de données, pour convertir les phénomènes physiques en signaux électrique tels qu'une tension ou un courant.

Selon le type de sortie du capteur on distingue : les capteurs analogiques (sortie tension ou courant) et les capteurs numériques (exemple : codeur incrémentaux, codeurs absolus, etc).

On trouve aussi les capteurs intelligents dont leurs éléments de traitement numérique de données sont intégrés, ce qui lui donne des capacités d'interprétation autonomes. Ils peuvent être connectés aux dispositifs de traitement directement à travers des interfaces de communication comme le bus I2C (Inter-Integrated Circuit), le bus CAN (Controller Area Network) ou encore le port de communication série l'UART (Universal Asynchrone Receiver Transmitter)**,** etc [5].

I.3.2. Conditionnement du signal

Les conditionneurs de signaux ont pour fonction de prendre la sortie des capteurs analogiques et de la préparer pour être échantillonnée numériquement. La partie conditionnement se compose d'une partie d'amplification, linéarisation, filtrage et conversion en format plus adapté avec le convertisseur analogique /numérique ou l'ADC (Analog to Digital Converter).

I.3.3. Convertisseur A/D

Les signaux analogiques doivent être convertis en signaux numériques à l'aide du convertisseur A/D (ou bien ADC Analog to Digital Converter) afin qu'ils soient correctement acheminés et traités par le reste des équipements du data logger.

L'ADC est un circuit qui convertit une tension analogique en données numériques en utilisant un processus appelé quantification. Cela est accompli en échantillonnant l'entrée analogique à chaque déclenchement de conversion et en transférant le code binaire du niveau de tension déterminé dans le registre de sortie. L'ADC est l'opposé d'un DAC, qui convertit les nombres numériques en tension analogique.

I.3.4. Unité de traitement

L'unité de traitement peut être un micro-processeur ou un microcontrôleur. Elle a la tâche principale d'effectuer les traitements nécessaires sur le signal acquis, et gérer tous les autres éléments du data logger.

I.3.5. Unité de stockage

Enfin, les données seront stockées dans l'unité de stockage, de type mémoire non volatile, elle peut être intégré ou externe (carte SD (Secure Data), EEPROM (Electrically Erasable Programmable Read-Only Memory), Flash memory disque dur, clé USB, etc).

Remarque : Selon l'utilisation prévue et le coût du data logger, on peut trouver d'autres éléments intégrés dans son architecture. Comme illustré dans la figure 1.2, on trouve une interface IHM comme option.

I.4. Caractéristiques des DataLoggers

Les caractéristiques des enregistreurs de données dépendent du type de data logger, de l'application et des spécifications techniques requises. Cependant, il y'a des caractéristiques communes entre eux résumées comme suit [6] :

- ✓ **Taille** : la taille d'un data logger est déterminée par son nombre d'entrées et de sorties (E/S). Ceux qui ont un petit facteur de forme sont moins encombrés ; l'utilisation encore d'un petit boîtier les rend peu coûteux et discrets.
- ✓ **Capacité de stockage :** varie selon la capacité de la mémoire utilisée. Elle est souvent mesurée en Kilo-octets (Ko), Méga-octets (Mo), ou en Giga-octets (Go).
- ✓ **Nombre/Type de canaux :** chaque enregistreur de donnée a son propre nombre et type d'entrée (simples ou multi-canaux), pour connecter des capteurs. Ces entrées varient en fonction de la sortie du capteur qui leur est connecté (détaillés dans la section 5) [7].
- ✓ **Taux d'échantillonnage :** fait référence à la fréquence à laquelle le data logger est programmé ou configuré pour lire et enregistrer les données issues de ses différentes entrées. La plupart des data loggers ont de faibles taux d'échantillonnage, c'est la raison pour laquelle ils peuvent enregistrer pour de longues périodes [8].
- ✓ **Durée d'enregistrement/de fonctionnement :** c'est la durée pendant laquelle un data logger peut enregistrer des données avant que la capacité de sa mémoire ne soit saturée et nécessite un téléchargement ou un rechargement. Elle dépend de la capacité de la mémoire utilisée, taux d'échantillonnage et du système d'alimentation utilisé [8,9].

I.5. Types des entrées des DataLoggers

Les types des entrées d'un data logger présentent une caractéristique importante, qui définit l'application du data logger en question, plus un data logger possède une multitude d'entrées, plus il est bon, et adaptable avec une variété de capteurs, il peut donc être destiné à divers applications.

Les types de terminaux suivants sont couramment utilisés pour connecter les enregistreurs de donnée aux capteurs [6,7] :

a. **Entrées analogiques :** peuvent être des entrées de tension ou de courant, et sont souvent utilisées avec l'un des éléments suivants : capteurs à pont, sorties de courant (0-20 mA (milliampère), 4-20mA, 0-40mA), thermomètres à résistance de platine (PRT : Platinum Resistance Thermometers), potentiomètres, transducteurs de pression, ponts résistifs, jauges de contrainte, thermistances, thermocouples, thermopiles, fils vibrants, courants de tension avec résistances shunt, sorties de tension (compris les mesures asymétriques et différentielles).

b. **Entrées numériques :** on trouve :

   - Compteurs d'impulsions : sont souvent utilisés avec l'un des éléments suivants : anémomètres, fermetures par contact, débitmètres, pluviomètres à auget basculant.
   - Ceux qui sont configurés en tant qu'entrées logique et exécutent des fonctions telles que la détection d'état.

c. **Ports série :** ports RS-485 (RS : Recommended Standard), I2C, CAN et UART pour connecter les capteurs séries intelligents.

Remarque : Le panneau de câblage d'un enregistreur de données comporte égale0ment des bornes permettant de connecter des alimentations, des périphériques de communication et des dispositifs de contrôle externes.

I.6. Fonctionnalités annexes et options points fort des DataLoggers

Actuellement, le marché des DataLoggers est compétitif, avec de nombreux fabricants (tel que Testo, Vaisala, Onset, Omega, MadgeTech, etc.) proposant une large gamme de produits avec des fonctionnalités variées pour répondre aux besoins des utilisateurs, citant quelques-unes de ces fonctionnalités :

✓ **Capacité d'expansion :** certains data loggers possèdent des modules d'extension, tel qu'àtravers eux le data logger aura une extension dans le nombre et le type d'E/S, au-delà de ce qu'il peut fournir sur son panneau de câblage [6].

✓ **Traitement des données à bord :** Certains enregistreurs de données fournissent un traitement de données issues du capteur, tel que le calcul des minimums, maximums, moyennes ou d'autres valeurs statistiques journalières, ce qui permet d'obtenir les informations nécessaires en moins grand nombre, donc une quantité réduite de données à stocker et à transmettre, ce qui donnera une durée d'enregistrement plus long en termes de capacité de mémoire [6].

✓ **Interface d'utilisation :** Certains enregistreurs de données sont équipés d'une interface IHM (Interface Homme Machine), à partir de laquelle l'utilisateur pourra configurer le taux d'échantillonnage du data logger, ainsi la visualisation et l'analyse des données. En se connectant à eux via des interfaces de communication, qui peuvent être une connexion câblée tel qu'Ethernet, USB (Universal Serial Bus) ou UART, etc,ou bien une connexion sans fil comme Wi-Fi (Wireless Fidelity), Bluetooth, ZigBee, LoRa (Long Range), NFC (Near Field Communication), cellulaire (M2M : Machine To Machine), etc [8].

✓ **Fonctionnement en temps réel :** Certains data loggers offrent une capacité de surveillance en temps réel ; l'affichage peut être assuré sous forme d'un écran LCD (Liquid-Crystal Display) intégré qui affiche les valeurs numériques mesurées en temps réel, ou bien sur un ordinateur à travers une connexion filaire ou sans fil [8].

I.7. Types des DataLoggers

Il existe plusieurs types du data loggers disponibles sur le marché, possédant une variété de fonctionnalité et caractéristiques comme cité dans les sections précédentes, d'où nous résumons dans le Tableau 1.1 les différents types des data loggers et leurs classifications selon différents critères et selon leur utilisation [10,11].

Tableau I.1. Classification des DataLoggers.

Classification	**Dataloggers**
Par déploiement	Autonome (mode standalone)
	Connecté au PC (Personnal Computer)
	Cloud
	GNSS/GPS (Global Navigation Satellite Système/Global Positioning System)
Par canal	Canal unique
	Multicanal
Par paramètre d'entrée	Température
	Pression
	Humidité
	Tension
	Autres paramètres d'entrée
Par industrie de l'utilisateur final	Pétrole et gaz
	Industrie manufacturière
	Automobile
	Énergie et services publiques
	Autres industries d'utilisateurs finaux
Par interface de communication	USB
	Ethernet
	Sans-fil (WI-FI, LoRa, Bluetooth...etc)

I.8. Domaine d'application des DataLoggers

Les DataLogger sont employés dans plusieurs domaines, notamment l'agriculture, l'industrie, l'environnement, la recherche scientifique et la surveillance des bâtiments, etc. tels que [10,11] :

- ✓ Dans le domaine de la surveillance environnementale pour mesurer des paramètres tels que la température, l'humidité, la pression atmosphérique, le CO2 (**dioxyde de Carbon),** la qualité de l'air, la lumière, le bruit, etc.
- ✓ Dans les chaînes d'approvisionnement alimentaire, pharmaceutique et de produits périssables. Ils sont également utilisés pour la surveillance de la température des entrepôts de stockage, des laboratoires, des musées, etc.
- ✓ Dans les applications de contrôle de processus industriels pour la surveillance, l'analyse et la régulation des processus industriels.
- ✓ Dans les applications de recherche scientifique pour mesurer les paramètres physiologiques tels que la température corporelle, la pression artérielle, la fréquence cardiaque, etc.
- ✓ Les boîtes noires des avions, appelées enregistreurs de données de vol. Ils sont essentiels pour déterminer les causes des accidents et diagnostiquer d'autres problèmes des compagnies aériennes.
- ✓ La surveillance de l'agriculture permettant aux agriculteurs de collecter en permanence des données sur la température, l'humidité du sol, l'humidité et d'autres facteurs clés afin d'améliorer le rendement, la survie et la qualité des cultures.

I.9. DataLoggers versus les systèmes DAQ classiques

Les terminologies "data logging" et "data acquisition" sont souvent utilisées indifféremment, mais elles ont des contextes historiques différents. Un data logger est un système d'acquisition de données, mais un système d'acquisition de données n'est pas nécessairement un data logger. Ces derniers ont généralement des taux d'échantillonnage plus lents que celui des systèmes d'acquisition de données [12]. Le Tableau 1.2 présente une brève comparaison entres eux [13].

Tableau I.2. Comparaison entre les DataLoggers et les autres les systèmes DAQ

Paramètre de comparaison	**DATALOGGER**	**SYSTÈME DAQ**
Taux d'échantillonnage	Très faible à moyen	Très élevé
Types d'entrée	Limité à quelques types de cœurs	Une très large gamme de types
Nombre de canaux	4-32 est commun. 50-100+ disponibles.	Habituellement 8 à 16 ; 32-100+ disponibles

Stockage des données	NVRAM (Non-Volatile Random Access Memory) interne	SSD (Solid-State Drive) interne
Temps d'enregistrement	Long	Court à moyen
Interface utilisateur	Habituellement, aucun affichage numérique ou rudimentaire.	Très robuste, avec une interface utilisateur graphique étendue
Surveillance en temps réel	Certains fournissent ceci	Tous fournissent ceci
Système d'alimentation	Habituellement alimenté par batterie	Habituellement, alimenté en externe AC / DC (Alternatif Current/Direct Current)
Coût	Très faible	Moyen à élevé

I.11. Interface de communication

Les interfaces de communication d'un data logger permettent de récupérer les données enregistrées et de les transférer vers un ordinateur ou un autre appareil, pour l'affichage et l'analyse des données. Il existe plusieurs types d'interfaces de communication pour les dataloggers, y compris les interfaces filaires et sans fil. Les interfaces filaires comprennent les ports USB, Ethernet, RS-232 et RS-485, qui permettent de connecter le data logger à un ordinateur ou à un autre périphérique via un câble.

Les interfaces sans fil, telles quele Wi-Fi, le Bluetooth, le Zigbee, LoRa permettent une communication à distance entre le data logger et d'autres dispositifs compatibles [14]. En résumé, les interfaces de communicationdes data loggers varient en fonction des besoins de l'utilisateur et de l'application spécifique,et de coût. Dans notre cas nous nous intéressons à la technologie de communication LoRa, détaillés ci-après.

I.11.1. Technologie de communication LoRa

LoRa (abréviation de "long range"), représente une technologie propriétaire de modulation d'ondes radio par étalement de spectre, dérivées de la technologie d'étalement du spectre chirp (CSS : Chirp Spread Spectrum). Il se concentre sur la communication bidirectionnelle sécurisée qui offre une longue portée et une faible consommation d'énergie, ce qui permet une communication à longue distance avec un faible débit de donnée [15]. LoRa a été créée en 2012, pour les LPWANs (Low Power Wide Area Network) utilisant les bandes ISM (Industriel, Scientifique et Médical) qui correspond aux bandes de fréquences sub-GHz (Géga Hertz), telles que 433 MHz (Méga Hertz), 868 MHz (Europe)et 915 MHz (États-Unis et Brésil) [16]).

Sa spécification est divisée en deux parties : la couche physique (PHY), brevetée par la société Semtech [17], et les couches supérieures (MAC : Medium Access Control), fondées par LoRa Alliance [18]. La couche PHY est caractérisée par les paramètres système suivants : débit de code (CR : Coding Rate), technique de modulation, facteur d'étalement (SF : Spreading

Factor), bande passante (BLoRa), la fréquence de la porteuse et la puissance de transmission [19]. LoRa est particulièrement adaptée pour les applications IoT qui nécessitent une longue portée, une consommation d'énergie faible et une couverture étendue. Elle peut être utilisée dans de nombreuses applications, telles quela gestion des déchets, la surveillance de la qualité de l'air, les villes intelligentes, l'agricultureintelligente, les réseaux d'eau intelligents, la logistique et bien plus encore.

I.11.2. Différence entre LoRa et LoRaWAN

Le protocole LoRa est un type de modulation de signal radiofréquence utilisé pour la communication entre les dispositifs LoRa ou entre un dispositif et une passerelle. En revanche, lorsqu'on parle de l'ensemble du processus de communication, c'est-à-dire du dispositif jusqu'au serveur LoRaWAN (Long Range Wide Area Network), on utilise le terme "protocole LoRaWAN".

LoRaWAN est une extension du protocole LoRa qui permet de connecter les dispositifs de manière sécurisée à un serveur pour transmettre des données à l'utilisateur final [20]. En résumé :

- ✓ **La couche physique LoRa :** correspond à la technique de modulation (CSS) et au format de la trame utilisé pour transmettre des données entre un émetteur et un récepteur.
- ✓ **Le protocole LoRaWAN :** est l'ensemble de l'architecture du réseau, qui inclut les dispositifs, les passerelles et les serveurs, ainsi que le format de trame permettant une transmission sécurisée de données entre un dispositif LoRaWAN et un serveur LoRaWAN.

I.12. Conclusion

En conclusion, nous avons exploré les généralités sur les dataloggers, ce qui nous permet à bien comprendre leur comportement et fonctionnement. Lesdataloggers continuent à évoluer avec l'avancement des technologies de l'information, offrant ainsi des fonctionnalités et des avantages de plus en plus développés.

Il est important de noter que l'utilisation des dataloggers nécessite une compréhension approfondie de leur fonctionnement et des considérations importantes à prendre en compte, telles que les caractéristiques des capteurs, les conditions environnementales, les méthodes de stockage et de traitement des données, ainsi que les normes et réglementations applicables.

Dans le chapitre suivant, nous allons explorer ces informations pour concevoir un datalogger personnalisé qui est l'objectif même de notre travail.

Chapitre 2

Conception d'un Data-Logger personnalisé

Chapitre II
Conception d'un DataLogger Personnalisé

II.1. Introduction

Dans ce chapitre, nous aborderons les différentes étapes de conception du datalogger, depuis la sélection des composants matériels jusqu'à la mise en œuvre des fonctionnalités logicielles nécessaires à la collecte, au stockage et à la récupération des données.

Nous explorerons également les considérations de conception importantes telles que la consommation d'énergie.

II.2. Objectif du travail

L'objectif de travail est de construire un Datalogger de type multicanaux qui peut supporter une variété de capteurs pour la mesure de température, de pression, de débit de gaz ou d'eau, etc, donc il peut être utilisé dans différents domaines, nous proposons en plus une architecture matérielle et logicielle facile à exploiter pour une extension future dans le nombre et le types de canaux du DataLogger, selon le besoin des utilisateurs.

II.3. Description générale du système

D'après l'étude que nous avons faite en se basant sur les informations détaillées dans le premier chapitre sur les Datalogger, nous avons sélectionné les fonctionnalités et les caractéristiques ainsi que l'architecture globale de notre système, comme décrit dont la figure II.1.

Les fonctionnalités et les caractéristiques de notre Datalogger sont les suivantes :

- **Fonctionnalités :**
  - Acquisition de données
  - Stockage de données
  - Transfère des données à une interface IHM via USB ou LoRa

- **Caractéristiques :**
  - 11 canaux du type :
    - Analogique (tension 0-3.3V),
    - Analogique (courant 4-20mA),
    - Numérique (compteur d'impulsion), logique (digital), port série (I2C, UART, RS-485, CAN),
  - Un taux d'échantillonnage configurable pour chaque canal, par défaut 1 seconde,
  - Stockage de données sur une carte SD,
  - Période d'enregistrement configurable selon les taux d'échantillonnage et la capacité de la carte SD utilisé,
  - Interface de communication USB,

- Interface de communication LoRa,
- Une IHM pour la configuration de Datalogger ainsi l'affichage des données,
- Horodatage des données enregistrées,
- Alimentation : par batterie.

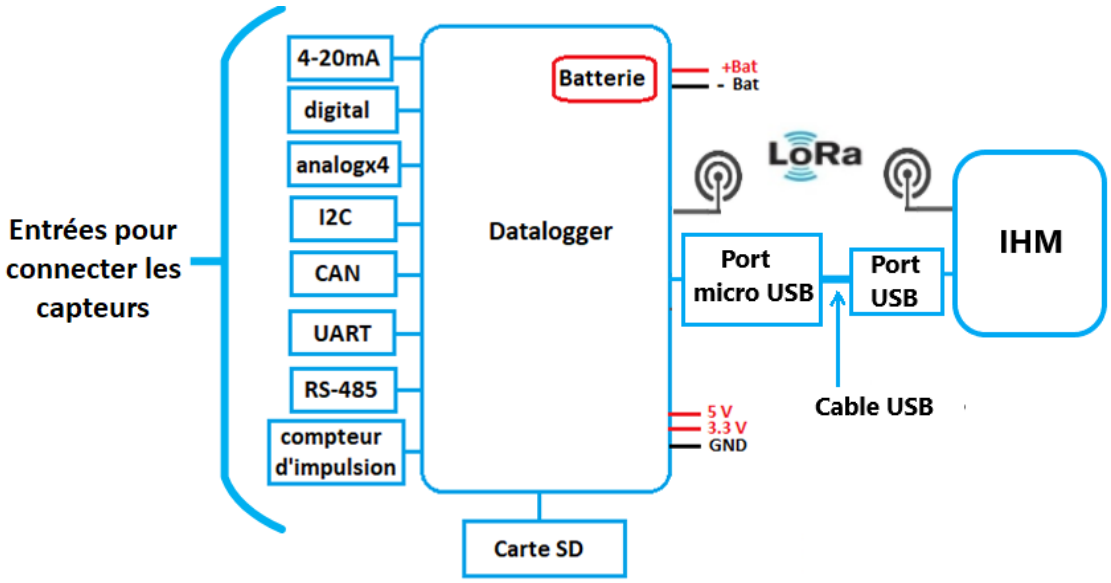


Figure II.1. Schéma descriptif du système conçu du datalogger

Le système est constitué de deux grandes parties, la partie du Datalogger et la partie IHM, dans ce qui suit nous allons décrire l'aspect matériel de chaque partie.

II.4. Description matérielle de Datalogger

L'architecture du Datalogger est organisée en 5 principaux blocs illustrés par la figure II.2:

1. Unité de traitement
2. Unité de stockage
3. Unité d'acquisition de données
4. Transmission de données,
5. Alimentation.

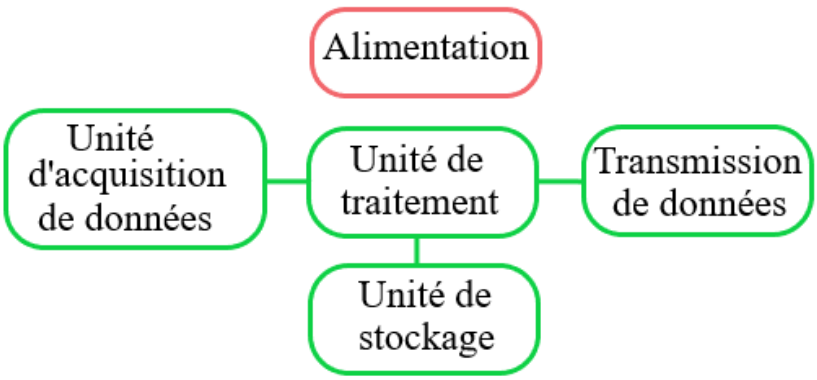


Figure II.2. Schéma bloc du datalogger

II.4.1. Unité de traitement

L'unité de traitement est basée sur un microcontrôleur haute performance STM32F4 muni d'un processeur ARM Cortex-M4 32 bits, avec unité à virgule flottante fonctionnant à 168 MHz.

Les principales caractéristiques de l'unité de traitement et de calcul sont (Voir Figure II.3) :

- ✓ Division câblée,
- ✓ Accès mots de 8, 16, 32 ou 64 bits,
- ✓ Instructions de traitement des champs de bits,
- ✓ Instructions MAC rapides et saturation,
- ✓ Opérations SIMD (Single Instruction Multiple Data),
- ✓ Virgule flottante.

Le STM32F407 supporte une alimentation allant de 1.8V à 3.6V. Il s'agit de l'alimentation des périphériques et de la mémoire. Par contre, le cœur a une alimentation plus basse. Un régulateur interne (inclus dans le boîtier) qui convertit la tension externe en 1.2VDC (Voltage Direct Current). Ce régulateur alimente le cœur du microprocesseur.

Par ailleurs, le microcontrôleur intègre un circuit supervisant la mise sous tension et la coupure d'alimentation (POR = Power-ON Reset, PDR = Power Down Reset).

Ce circuit s'assure que le cœur est toujours convenablement alimenté lors d'exécution d'instructions. Le circuit de monitoring de voltage offre aussi des services de détection de faute d'alimentation (PVD = Programmable Voltage Detector) et de reset lors de chutes brèves de tension (BOR = Brown Out Reset) [21].

STM32F407 dispose d'une connectivité riche en périphériques avec [21] :

- ✓ 1M B Flash,
- ✓ 192+4 Kbytes de SRAM include 64-Kbyte de CCM (Core Coupled Memory) data RAM,
- ✓ GPIO (82) avec une capacité d'interruption,
- ✓ 12-bit ADC (3) avec 16 canaux,
- ✓ 12-bit DAC (1) avec 2 canaux,
- ✓ RTC,
- ✓ Timers (14),
- ✓ I2C (3) interfaces (SMBus/PMBus),
- ✓ I2S (2),
- ✓ USART (4),
- ✓ SPI (3),
- ✓ USB 2.0 full-speed,
- ✓ USB 2.0 OTG,
- ✓ CAN (2),
- ✓ SDIO interface,

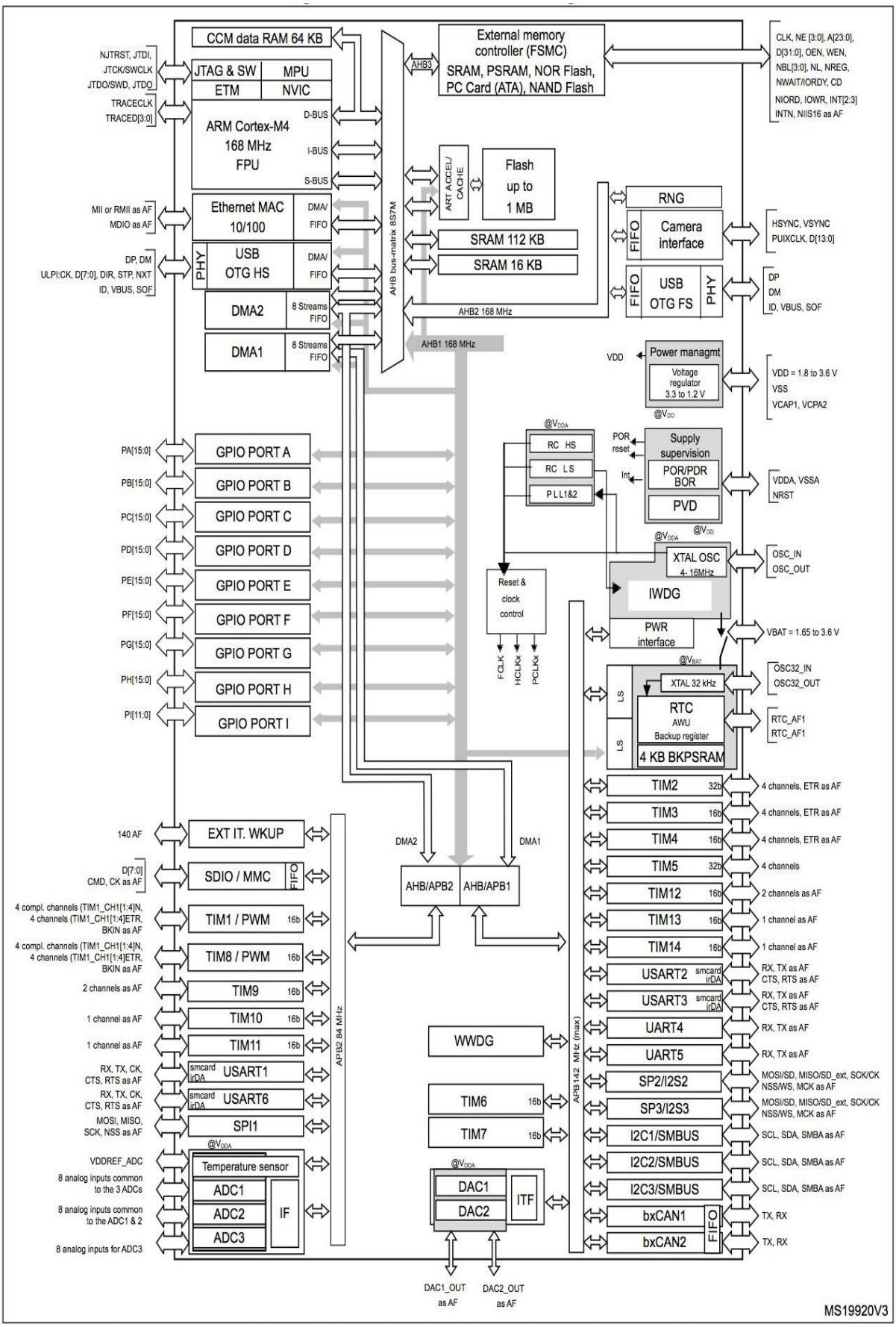


Figure II.3. Schéma bloc du microcontrôleur STM32F407VGT6 [21]

II.4.1.1 Plateforme de développement matérielle STM32F407VGT6

La carte de développement STM32F407VGT6 fournie une plateforme de développement complète pour le prototypage rapide avec une facilité d'utilisation et grande flexibilité (voir la figure II.4).

Cette carte dispose d'une variété de fonctionnalités, avec les caractéristiques suivantes [21] :

- ✓ Un microcontrôleur STM32F407VG puissant et polyvalent, qui intègre un processeur ARM Cortex-M4 cadencé à 168 MHz,
- ✓ Tension de fonctionnement : 3.6-5V,
- ✓ 2 oscillateurs à cristal (quartz), l'un de 8 MHz et l'autre de 32.768KHz, qui peuvent être utilisés pour piloter l'horloge interne RTC (Real Time Clock),
- ✓ Interface de SWD (Serial Wire Debug) et JTAG (Joint Test Action Group) pour debugging,
- ✓ Accès au 140 ports d'E/S du microcontrôleur,
- ✓ Batterie pour le module RTC,
- ✓ Connecteur USB,
- ✓ Support pour écran TFT CD,
- ✓ Dimensions : 40.89mm x 68.59mm.

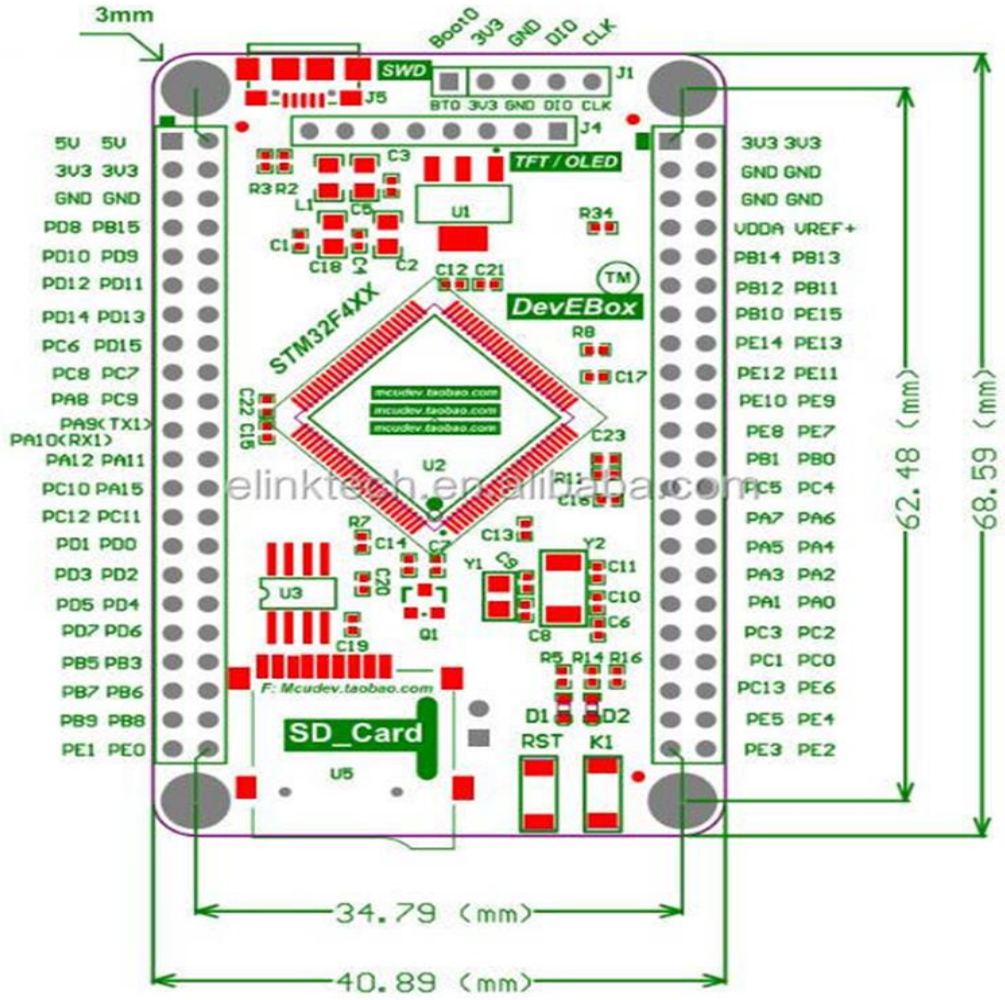


Figure II.4. Illustration de la carte STM32F407VGT6 [21]

II.4.1.2 Sonde de programmation ST-LINK

Pour injecter le programme dans la carte, on utilise son interface SWD, utilisant le module ST-LINK V2. Cette sonde de programmation et de débogage est équipée d'un connecteur IDC à 10 broches. Le brochage de ce connecteur est indiqué ci-dessous dans la figure II.5.



Figure.II.5. Illustration du module ST-LINK V2 et ses pins [22]

II.4.2. Unité de stockage

L'unité de stockage est une carte SD, une carte SD (Secure Data) est une carte mémoire amovible utilisée pour stocker des données numériques. Elle a été créée en 2000 par une alliance entre Panasonic, SanDisk et Toshiba. Les cartes SD sont largement utilisées les systèmes embarqués, les appareils électroniques tels que les appareils photo numériques, les smartphones, etc. le tableau II.1 suivant résume la classification des cartes SD selon leur capacité et forme [23].

Tableau II.1 : Classification des types des cartes SD [23]

Norme	Capacité		Forme	Taille (en millimètres)
SD	2Go		Carte SD	24 × 32 × 2,1
SDHC	2 à 32 Go		Carte mini SD	20 × 21,5 × 1,4
SDXC	32 Go à 2 To		Carte micro SD	15 × 11 × 1
SDUC	2à 128 To			

La carte SD peut être directement exploitée via nôtre carte de développement qui intègre un support pour ce type de mémoire, le support est relié au microcontrôleur via l'interface SDIO, comme indiqué sur le schéma de câblage qui suit en figure II.6 :

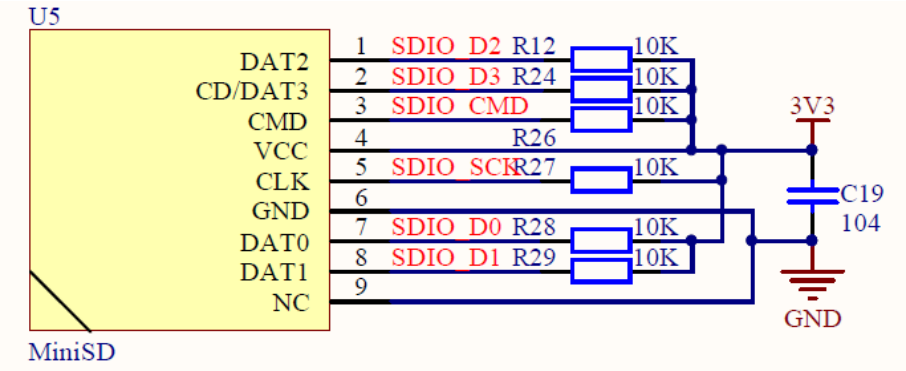
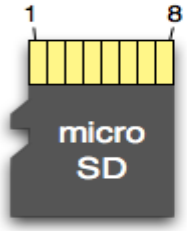


Figure II.6. Schéma électrique de l'interface SDIO SD card [24]

Les broches de la carte Micro SD sont indiquées dans la figure II.7 ci-dessous :



Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

Figure.II.7. Pins de la carte SD [25]

II.4.3. Unité d'acquisition de données

L'unité d'acquisition représente tous les éléments qui permettent au Datalogger d'interagir avec le milieu physique, il est constitué de 3 trois éléments (Voir la figure II.8) :

1. Capteurs
2. Conditionnement de signal
3. Interfaces d'entrées



Figure II.8. Eléments constituants de bloc d'acquisition des données

II.4.3.1 Capteurs

Un capteur est un dispositif permettant de transformer une grandeur physique observée en une grandeur physique utilisable. Il sert à collecter toutes sortes d'informations terrain : tensions électriques, vibrations, températures, pressions, sons, consommation de lubrifiants ou d'électricité, etc. Le capteur traditionnel ne fait que collecter l'information qui est ensuite interprétée par un être humain. Le capteur dit « intelligent » intègre des éléments de traitement numérique de la donnée, qui lui confère des capacités d'interprétation autonomes.

Les capteurs intelligents se composent ainsi de trois éléments : un ou plusieurs capteurs « classiques », un microprocesseur et une interface de communication. Ils sont par exemple capables d'identifier des signaux faibles annonçant un problème technique, de réaliser des auto-diagnostiques ou encore d'engager des actions en fonction de règles prédéfinies.

Afin de répondre à un large éventail de types de capteurs nous avons opté pour une architecture basée sur une utilisation sur bus, cette configuration permet de ratcher plusieurs capteurs de type numérique (capteur intelligent) sans pour autant augmenter le nombre de file, mais nous avons aussi prévue de raccorder directement des capteurs délivrant des grandeurs de type analogiques pour la mesure de courant ou de tension et des capteurs de type tout ou rien (1/0).

Le tableau II.2 suivant décrit quelques exemples de capteurs pouvant être pris en charge par notre DataLogger.

Tableau II.2 : Exemples de capteurs supportés par le DataLogger

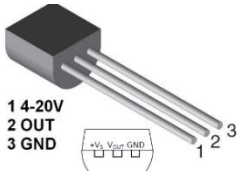
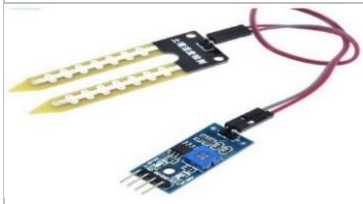
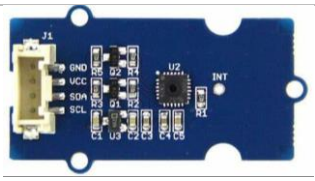
Le capteur	Sortie	Description	Caractéristique
 **Figure II.9 :** TPC ½-IP68 capteur de pression [26]	4-20mA	Permet de mesurer la pression en eau usée, eau potable ou eau brute. La prise en charge est assurée par un raccord G ¼ ou ½ pouce mâle.	- Plage de mesure : 0 - 0,1 bar à 0 – 25 bar -Alimentation : 7 à 38 V avec limitation à 1A
 **Figure II.10 :** LM35 capteur de température [27]	Analogique	C'est un capteur passif qui nécessitent un apport d'énergie externe pour fonctionne, permet la mesure de la température.	-Plage de mesure : -55°C à +150°C. -Alimentation :4-20V -Précision : ±0,5°C
 **Figure II.11 :** MQ135 capteur de gaz [28]	Analogique /numérique	Il s'agit d'un capteur de gaz de type semi-conducteur à oxyde métallique (MOS). La détection est basée sur le changement de résistance du matériau de détection lorsque le gaz entre en contact avec le matériau	-Plage de mesure : Détection de 200 à 10000ppm -Alimentation : 5V -Résistance de détection : 10 kΩ - 60 kΩ
 **Figure II.12 :** ACS712 capteur de courant [29]	Analogique / Numérique	Il permet la détection de courant AC ou DC. Son principe de fonctionnement basé sur l'effet Hall.	-Plage de mesure : -30A à +30A - Alimentation : 5V - Sensibilité : 66mV/A - Consommation : 10mA -Erreur : 1.5% à 25°C
 **Figure II.13 :** YL-96 capteur d'humidité de sol [30]	Analogique/ Numérique	Ce module de capteur d'humidité du sol permet de détecter des changements d'humidité. Avec réglage de seuil.	-Plage de mesure : 0 à 1023 analogique et 0 ou 1 numérique -Alimentation : 3.3V à 5V
 **Figure II.14:** TH02 capteur de température et d'humidité [31]	I2C	Il est basé sur le capteur TH02 qui mesure la température et l'humidité dans des plages usuelles.	-Plage de mesure : température : 0°C à 70°C, humidité : 0 à 80 -Alimentation : 3.3V à 5 V -Précision : ±0,5°C

 **Figure II.15 :** sonde de température ambiante RS485 [32]	RS-485	Il est conçu pour mesurer les concentrations de dioxyde de carbone (CO2), la température (T) de l'air, l'humidité relative de l'air (H), la pression atmosphérique (P) et les concentrations de composés organiques volatils (COV).	-Plage de mesure : T : 0 à 45 ° C (± 0,5 ° C) H : 0 à 85% (± 3%) CO2 : 400 à 5000 ppm (± 100 ppm) COV : 0 à 500 (± 15%) P : 300 à 1100 hPa (± 0,6 hPa) -Alimentation : 15 à 30 VDC
 **Figure II.16** : capteur de température avec tige et entête avec protocole CAN [33]	CAN	Permet la mesure de température, il comprend une tête en plastique et tube en acier inoxydable dans lequel est placé le capteur, et un convertisseur avec un bornier, auquel est connecter le câble d'alimentation et de sortie.	-Plage de mesure : -50 à 150 ° C -Alimentation : 15 à 30 VDC Précision électronique : ± 0,2 ° C -Type de capteur / précision Pt 100 / ± (0,3 ° C + 0,0005 | t |) -Dimension de la tige : 70, 120, 180, 240mm / 6 ± 0,2 mm
 **Figure II.17 :** capteur de débit d'eau YF-S401 [34]	5V TTL	Il destiné pour la mesure de débit d'eau, et il est constitué d'un rotor et d'un capteur à effet hall. Lorsque l'eau passe dans le capteur, la rotation du rotor génère des impulsions en fonction du débit. La fréquence de sortie varie en fonction de la vitesse du rotor	-Alimentation : 5 à 18 V DC -Plage de mesure : de 1 à 30 litres / minute -Température de fonctionnement : -25 à 80 -Précision : ± 2% -Pression maximale de l'eau : 2,0 MPa -Caractéristiques d'impulsion de débit : Fréquence (Hz) = 7.5 * Débit (L / min)

Remarque :

On ne dispose pas de tous les capteurs cités au-dessous en raison de leurs indisponibilités sur le marché, d'autres solutions de test seront décrites dans le chapitre suivant, ainsi notre objectif est de construire un DataLogger avec des interfaces aptes à connecter différents types de capteur, que l'utilisateur va choisir en fonction de ces besoins.

II.4.3.2 Interfaces d'entrées et conditionnement du signal

1. **Entrées analogiques**

Nous disposons de 4 entrées analogiques pour connecter des capteurs de sortie analogique. Ce type d'entrée nécessitera un module d'ADC (Analog to Digital Converter) pour qu'il soit traité par le microcontrôleur. Le microcontrôleur choisis intègre 3 modules d'ADC et chacun a 16 canaux, avec une résolution allant jusqu'à 12 bits, et une tension d'entrée allant de 0 à 3.6V. Il prend en charge plusieurs modes de fonctionnement, notamment le mode polling, le mode DMA et le mode interruption, voir l'exemple illustré par la figure II.18.

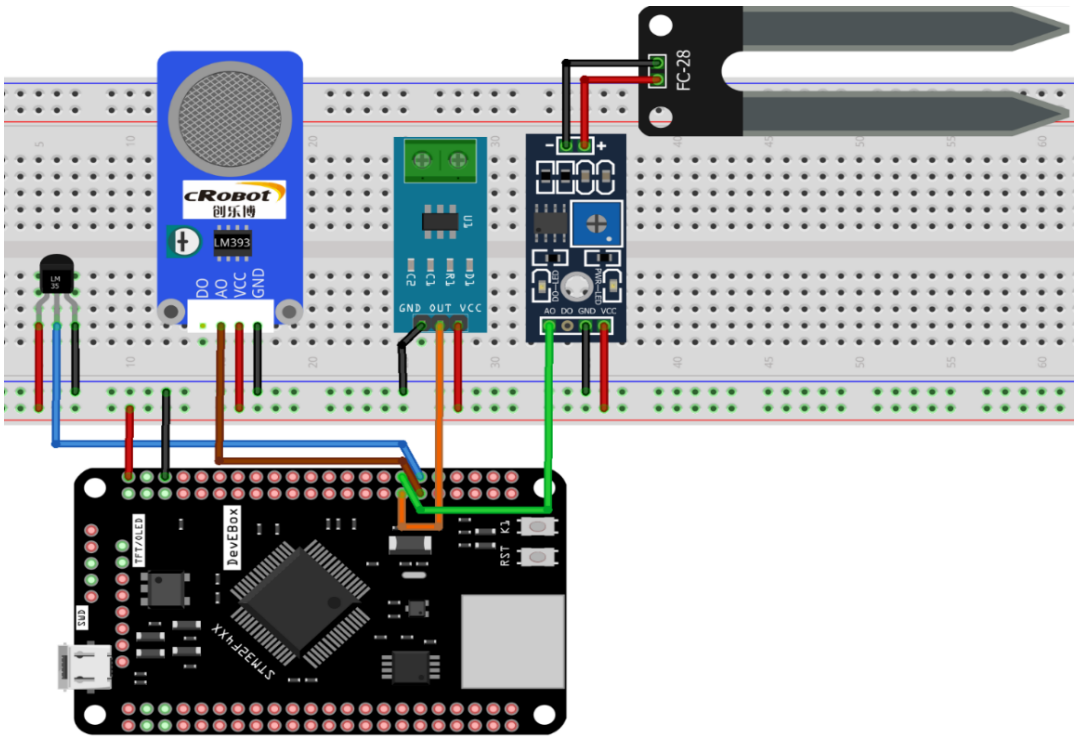


Figure II.18. Schéma de câblage des capteurs avec les entrées analogiques de la carte STM32F407VGT6

Pour notre application spécifique, nous avons choisi d'utiliser l'ADC1 avec 4 canaux (IN10, IN11, IN12, IN13) en mode DMA, qui permet un transfert rapide de données entre les périphériques et la mémoire. Cela permet d'optimiser les performances en termes de vitesse de transfert de données et libéré l'unité de traitement pour autres taches.

Le tableau II.3 suivant présent les pins de la carte correspond aux canaux sélectionnés ainsi les pins des capteurs utilisés.

Tableau II.3 : Table de connectivité

Canal	Capteur	Pin sur la carte	Pin de capteur
IN10	LM35	PC0	Pin 2 Vout
IN11	MQ2	PC1	A0
IN12	ACS712	PC2	Pin 2 Vout
IN13	Capteur d'humidité de sol	PC3	A0

2. Entrée 4-20mA et le convertisseur 4-20mA à tension DC :

Les boucle de courant 4-20 mA sont largement utilisées au niveau des industries de process (agro-alimentaire, pétrochimie, pharmaceutique etc..). Il est utilisé pour transmettre des signaux analogiques provenant de capteurs analogiques (capteur de pression, température, débit, niveau,

vitesse etc..) à des signaux courant 4-20mA, on trouve des capteurs ou le transmetteur est intégré avec lui comme l'exemple de capteur de pression qu'on a cité déjà, et d'autres utilisant des transmetteurs de boucle avec le capteur comme les PT100 qui sont des capteurs de température de type RTD utilisés souvent dans l'industrie avec des boucles de courant 4-20mA, la (figure II.19) présente une illustration de cet exemple [35].

Le 4-20 mA, nous rencontrons aussi dans l'industrie l'échelle 0-20 mA, cependant le 4-20mA reste le plus performant. En effet, le fait que la valeur basse de la boucle soit à 4 mA ou lieu de 0 mA à plusieurs avantages [36] :

- ✓ Détection de la rupture de fil lorsqu'on est en dessous de 4 mA. Cela facilite ainsi les opérations de diagnostic.
- ✓ La boucle de courant 4-20 mA peut transmettre des données numériques HART via les mêmes câbles sans aucune interférence entre les deux.
- ✓ Un signal de courant, en général, est plus immunisé contre les bruits électriques que tout autre signal de tension (0-10 VDC, 1-5 VDC) et peut également fonctionner sur de longues distances.
- ✓ En utilisant un multimètre standard, nous pouvons détecter les défaillances ou vérifier facilement la boucle 4-20 mA. En outre, il n'y a aucun risque personnel si nous touchons le câble car la tension est de 24 VCC et que le seuil de courant dangereux pour le cœur est de 30 mA.
- ✓ La facilité de conversion de signal de la boucle de courant en une plage de tension pour le connecter à un contrôleur par exemple.

Vu l'utilité des boucles de courant 4-20mA, nous avons fait une entrée pour lire ce type de signal. Pour construire cette entrée avec la carte stm32f407vgt6, nous avons besoin d'un convertisseur courant vers une tension de 0-3.6V, car notre carte ne peut traiter que des signaux analogiques de tension, donc le même traitement qu'on a décrit pour les entrées analogiques de tension sera fait par la carte sur le signal, après sa conversion. Pour cette application nous allons utiliser l'ADC2 avec le canal IN0 qui relier sur la carte avec la pine PA2 comme indiqué sur la figure II.20.

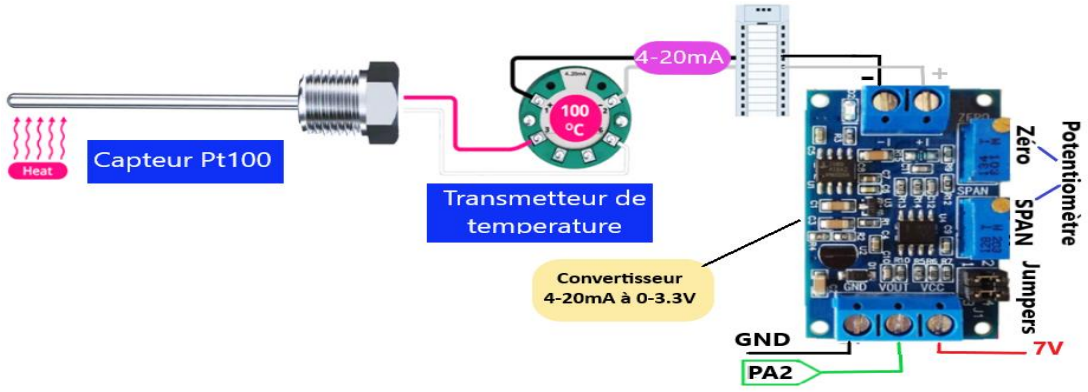


Figure II.19. Exemple d'utilisation d'un transmetteur de boule 4-20mA avec le capteur de température PT100 et son branchement à l'entrée 4-20mA de datalogger [37]

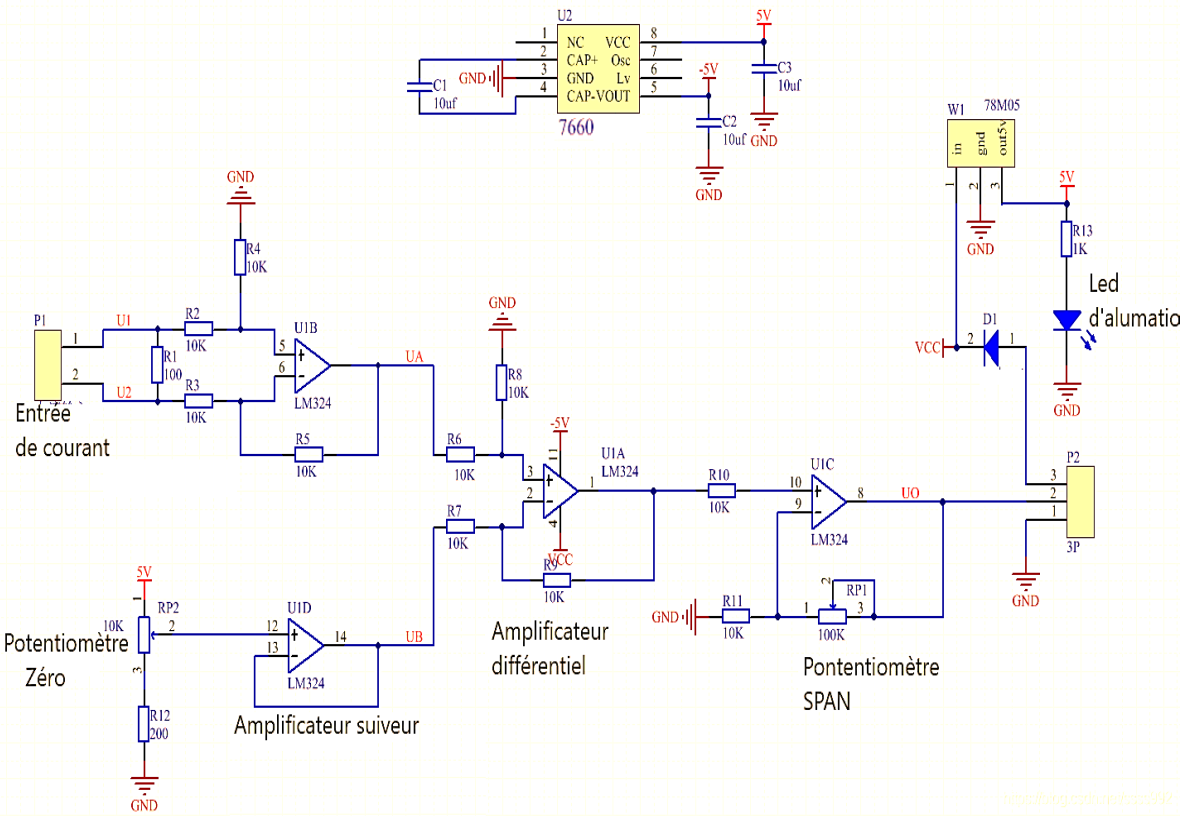


Figure II.20. Schéma électronique de module de conversion 4-20Ma à tension [38]

❖ Description de module de conversion 4-20mA à tension

Ce module est un circuit qui peut recevoir un signal variable de 0 ou 4 à 20mA et de le convertir en un signal de tension de 0 à 15V réglable. De cette façon, il est possible de le connecter directement à l'entrée analogique de microcontrôleur. Dans le circuit, il y a un potentiomètre qui fait le réglage de la tension de sortie en fonction de besoins, pouvant varier de 0 à 15V, son schéma électrique est illustré dans la figure II.21.

❖ Mode d'emploi

1. Tension d'alimentation : 7– 36 V (si la sortie est de 10V, la tension d'alimentation doit être supérieure à 12V).
2. Après la mise sous tension, l'indicateur D2 doit être allumé, sinon on vérifie la connexion de la ligne.
3. Lorsque l'entrée de courant est le minimum (4mA ou 0mA), on règle le potentiomètre « Zéro » de sorte que la sortie de sortie soit le minimum (0.0V ou autre tension).
4. Lorsque l'entrée de courant est maximale (20mA), on règle le potentiomètre « SPAN » de sorte que la tension de sortie soit maximale (3.3V ou 5V ou 10V, la sortie peut être aussi basse que 2.5V lorsque l'entrée est 4-20mA).

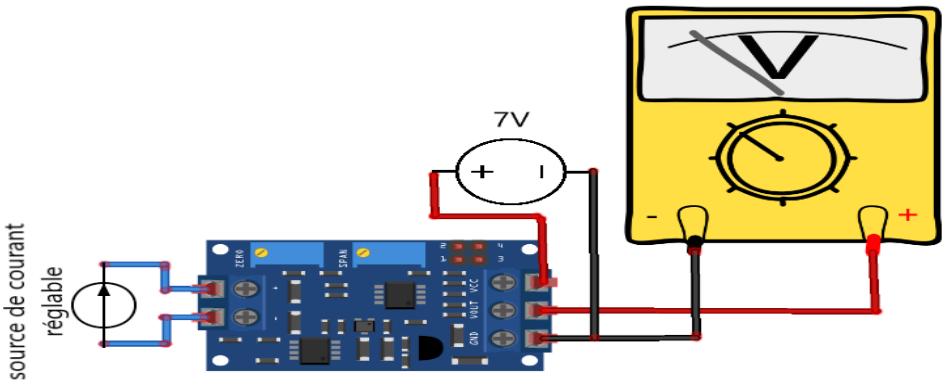


Figure II.21. Schéma électrique du calibrage

Le tableau II.4, suivant illustre les différentes configurations de ce module.

Tableau II.4 : Différentes configurations de ce module

Plage de courant d'entrée	Plage de tension de sortie	Broche			
		1	2	3	4
4-20mA	0-2.5 V	ON	ON	ON	ON
	0-3.3 V	OFF	OFF	OFF	OFF
	0-5 V	OFF	OFF	ON	ON
	0-10 V	ON	ON	OFF	OFF
0-20mA	0-3.3 V	ON	ON	ON	ON
	0-5 V	OFF	OFF	ON	ON
	0-10 V	ON	ON	OFF	OFF

3. Entrée UART

UART permet à deux éléments de communiquer ensemble, via une liaison série filaire, bidirectionnelle, basiquement, il utilise une liaison série de trois fils, dont deux fils de communication (Tx pour envoyer des données, et Rx pour recevoir des données) et un fil pour la masse, pour garantir l'équipotentialité de ces signaux.

Le branchement de deux éléments communicants avec l'UART se fait en croisement des fils Tx1 avec Rx2 et Tx2 avec Rx1, afin que les données ne circulent que dans un seul sens, au niveau de chaque fil, et la vitesse d'envoi de ces données est normalisée, les valeurs standard les plus utilisées sont : 1200, 2400, 4800, 9600, 115200 bauds (bit par seconde). Une trame UART est constituée des bits suivants [39] (Voir la figure II.22) :

- ✓ Un bit de start toujours à 0 : servant à la synchronisation du récepteur
- ✓ Données : la taille des données est comprise entre 5 et 9 bits. Bits envoyés du LSB (bit de poids faible) au MSB (bit de poids fort).
- ✓ Parité : Paire ou Impaire (optionnel)
- ✓ Fin : Un bit de stop, toujours à 1. La durée de celui-ci varie entre 1, 1.5 et 2 (A l'utilisateur de choisir).

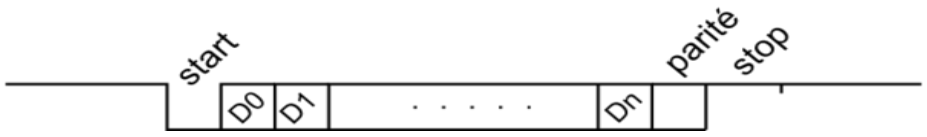


Figure II.22. Trame de données d'une transmission série UART [39]

Vu la simplicité et l'efficacité de ce bus de communication, il est utilisé souvent dans les capteurs intelligents série, pour cette raison le datalogger est équipé d'une entrée UART. Ce type de communication est souvent utilisé pour les systèmes de surveillance à distance.

La carte de développement STM32f407VGT6 à 6 interfaces UART avec différente vitesse d'envoi, pour notre application nous allons sélectionner UART1 avec une vitesse 115200 bauds dont Tx est relie à la pin PA9 et Rx à la pin PA10 de la carte.

4. Entrée RS-485 et convertisseur TTL to RS-485

RS-485 est une norme de communication définie par l'Electronic Industry Alliance (EIA) et la Télécommunications Industry Association (TIA). C'est une spécification industrielle qui définit les caractéristiques électriques des transmetteurs et des émetteurs destinés à être utilisés dans des systèmes de communication multipoints en mode symétrique. Elle permet de mettre plusieurs appareils RS485 sur le même bus qui est une ligne électrique, pratiquement une paire torsadée, par des variations de tension en mode différentiel, avec une topologie « daisy chain » ce qui permet à plusieurs nœuds de se communiquer les uns avec autres, environ 32 équipements. Cela signifie que le câble du bus va directement d'esclave en esclave, comme indiqué dans la figure II.23 [40].



Figure II.23. Emplacement de plusieurs nœuds sur le bus RS-485

Le tableau suivant montre une comparaison entre les 3 standards de communication série :

Tableau II.5 : Comparaison des standards de communication série [40]

	RS-232	**RS-422**	**RS-485**
Topologie	Point to point	Point to point et multipoint	Point to point et multipoint
Mode de communication	Full duplex	Full duplex Half duplex	Full duplex Half duplex
Nombre d'appareil	1 contrôleur 1 récepteur	1 contrôleur 10 récepteurs	32 récepteurs et émetteurs
Niveau de signal	Undifférentiel	Différenciel	Différentiel
Distance maximal	60m à 2.4 kbps	1.2 km à 100 kbps	1.2 km à 100 kbps
Niveau de tension pour le niveau logique (1) ON	-15 à -3 V	1.5 à 5 V	2 à 6 V
Niveau de tension pour le niveau logique (0) OFF	+3 à +15 V	1.5 à 5 V	2 à 6V

D'après les critères de comparaison entre le protocole RS-485 et les protocole séries, nous remarquons qu'il est le plus fort par rapport aux autres en termes de nombre de récepteurs et émetteurs qu'il peut connecter à la fois. En plus, ce protocole est utilisé largement dans l'industrie et c'est la norme utilisée dans la couche physique du protocole de communication ModBus.

La carte de développement que nous allons utiliser ne dispose pas d'une interface de communication RS-485, mais de l'UART donc pour avoir une interface RS-485 prêt à l'emploi par des capteurs ou des modules d'acquisition communiquant avec ce protocole, nous allons utiliser le convertisseur TTL/RS-485 qui prend des données UART en entrée et les convertis en RS-485.

❖ **Caractéristique :**

- Alimentation : 5V
- Module de conversion : MAX485
- Conversion des signaux TTL de UART en RS485 half-duplex

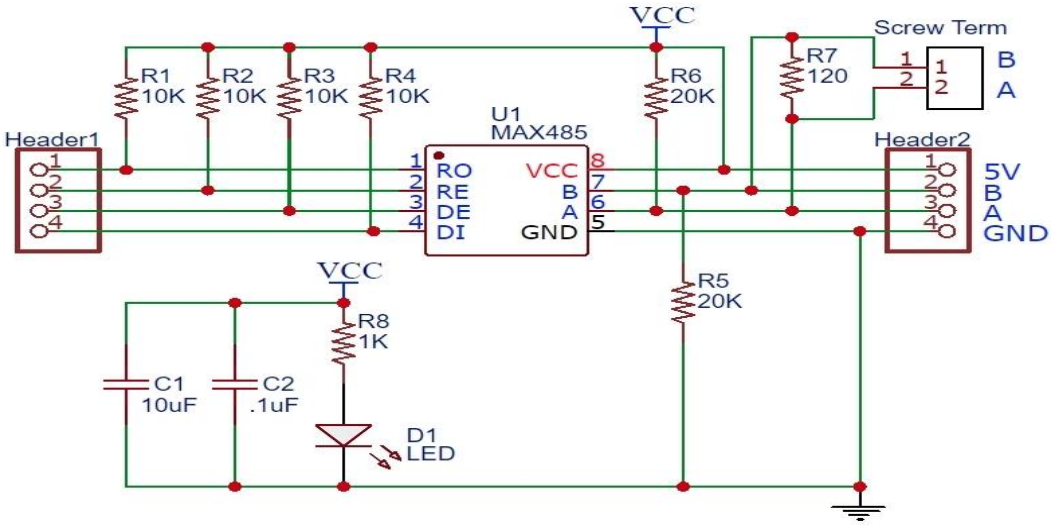


Figure II.24. Schéma électrique de Convertisseur de TTL to RS-485 [41]

❖ **Mode d'emploi**

Nous allons sectionner l'UART2 sur la carte STM32F407VGT6, dont Tx est relié à la pine PD5 et Rx à la pine PD6 ; les broches A et B sont les broches de sortie qui transportent le signal de transmission. Le branchement de ce module de conversion avec la carte illustrée par la figure II.25.

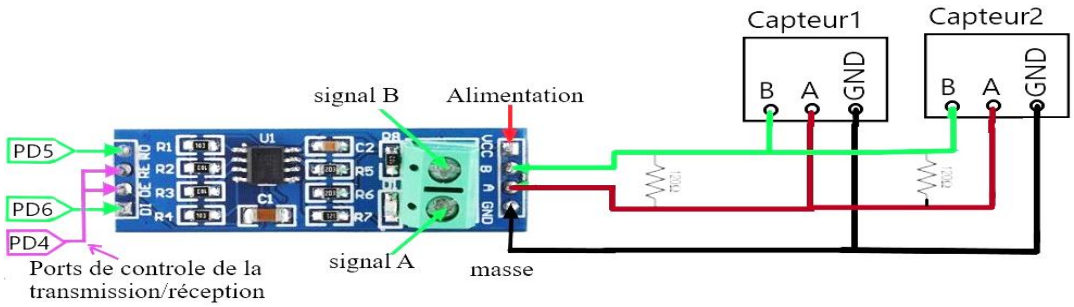


Figure II.25. Schéma de câblage des capteurs au bus RS485 et de module TTL /RS485 à la carte

Tableau II.6. Connectivité des pins de module avec celles de la carte

Pin de module	Pin sur la carte STM32F407VGT6
RO	PD5 (Rx)
DI	PD6 (Tx)
RE et DE	PD4

5. Entrée CAN et transmetteur de ligne

Le bus CAN (Controller Area Network) est un protocole de communication très efficace grâce à ses capacités de vitesse élevée, sa fiabilité à long terme et son immunité au bruit. Pour ces raisons, la communication CAN est devenue la norme dans les technologies automobiles et les environnements très bruyants.

Les dispositifs sur le bus CAN sont appelés nœuds. Tous les nœuds du bus CAN sont connectés en parallèle, ce qui signifie que chaque nœud est connecté à tous les autres nœuds du réseau. Un seul bus CAN peut comporter jusqu'à 115 nœuds à la fois, en fonction du taux de transmission des messages, mais pour la plupart des applications, il est recommandé d'avoir jusqu'à 32 dispositifs [42].

Pour communiquer avec le bus CAN, nous utilisons le circuit intégré émetteur-récepteur CAN MCP2551 avec la carte STM32F407VGT6 qui dispose de deux interface CAN, nous allons sélectionner le CAN1 qui est relié à la pin PB8 pour CAN1 Rx et la pin PB9 pour CAN1 Tx sur la carte.

Le transmetteur MCP2551 est un circuit qui agit comme une paire intermédiaire émetteur/récepteur pour connecter le STM32 au bus CAN, il a une tolérance de pannes CAN.

❖ **Caractéristiques du module MCP2551** [43] **:**

- Tension d'utilisation de 4,5V à 5.5V,
- Pente contrôlée de l'extérieur pour réduire les émissions RFI,
- Détection de défaut de masse (permanent dominant) sur entrée TXD,
- Protection contre les creux de tension et reset d'alimentation,
- Un nœud hors tension ou un creux de tension n'affectera pas le bus CAN,
- Fonctionnement veille faible courant,
- Protection contre les dommages causés par les courts-circuits (tension de batterie positive ou négative),
- Tensions transitoires élevées et la protection d'arrêt automatique thermique,
- Haute immunité au bruit grâce à l'implémentation de bus différentiel,
- Gamme de température d'utilisation de -40°C à 125°C.

Le schéma de la (figure II.27) décrit le branchement de module MCP2551 au carte STM32 en plus la façon de connecter des capteurs au bus CAN de datalogger.

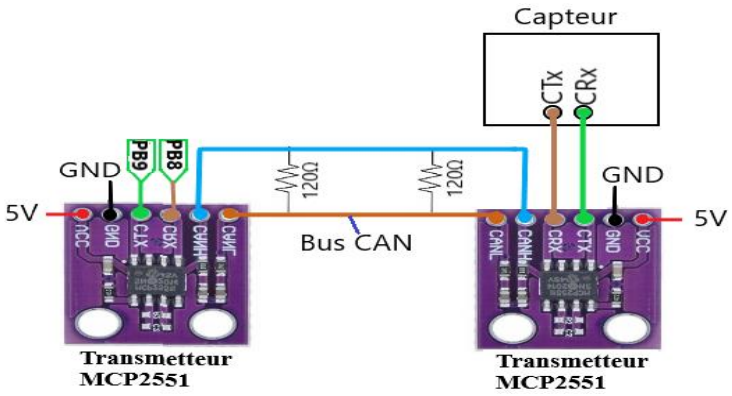


Figure II.27. Schéma de câblage de la carte STM32 avec un capteur sur le bus CAN utilisant le transmetteur MCP2551

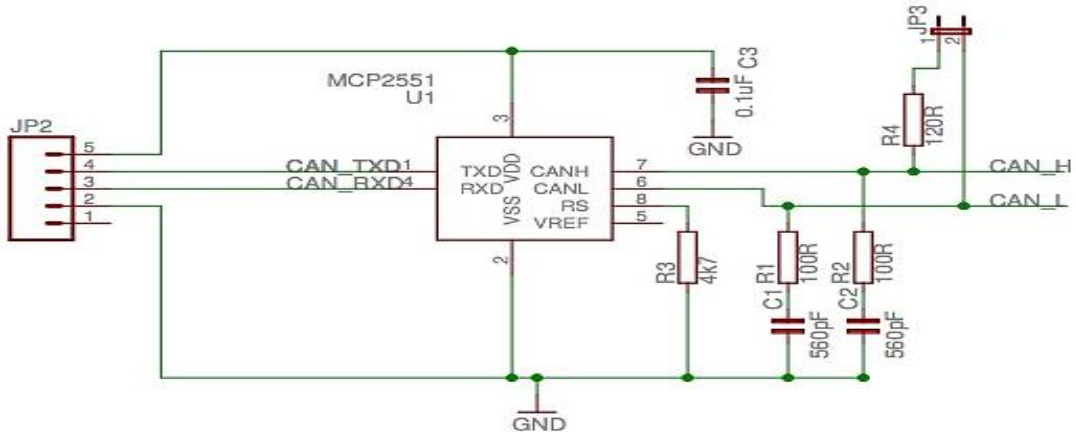


Figure II.26. Schéma électrique de transmetteur CAN MCP2551 [43]

6. Entrée numérique

On prévoie aussi des entrées numériques pour notre Datalogger, on a sélectionné une entrée pour mesurer la fréquence en comptant les impulsions entrées dans une seconde et une autre entrée logique pour la détection d'état ON d'une machine ou autre.

L'entrée d'impulsion permet le comptage d'impulsions électriques, selon la valeur maximale de registre pereload de TIMER on peut détecter la valeur maximale et minimale de la fréquence mesurée.

STM32F407VGT6 est équipé de 17 timers qui peuvent être utilisés comme des compteurs ou décompteurs à usage général avec rechargement automatique de 16 ou 32 bits, pilotés par un prédiviseur programmable à 16 bits, qui permet de diviser la fréquence de l'horloge par un facteur compris entre 1 et 65536.

Pour notre application va utiliser le TIMER1 en mode ETR pour compter les impulsions entrées et le TIMER6 pour générer une interruption chaque seconde et lire la valeur compter par le TIMER1 puis le remettre à zéro. L'entrée de signal impulsionnel sur la carte est la pin PE7.

Le TIMER1 peut compter de 0 à 65536 ce qui donne une fmax=65536 et fmin=0.

Pour l'entrée de détection d'état logique peut être réalisé facilement utilisant une GPIO input de la carte STM32, et on a sélectionné la pin PE12comme entrée logique, en mode EXTI (External Interrupt) pour détecter le moment d'activation de l'appareil relié.

7. Entrée I2C

I2C l'acronyme de "Inter-Integrated-Circuit" est un bus série synchrone bidirectionnel half-duplex, où plusieurs équipements, maîtres ou esclaves, peuvent être connectés au bus, utilisant deux fils seulement SDA (Serial Data Line : ligne de données bidirectionnelle) et SCL (Serial Clock Line : ligne d'horloge de synchronisation bidirectionnelle), avec un fil de masse. Il peut fonctionner en mode standard (débit maximal 100 kbps), en mode rapide (débit maximal 400kbps), ou bien en mode rapide plus (à 1Mbps) et en mode haut débit (à 3,4Mbps).

Un message I2C typique se compose de quelques éléments de base qui se déroulent sur le bus I2C de manière séquentielle comme indiqué dans la figure II.28, il commence par une condition de démarrage (S : START), suivi de l'adresse du dispositif esclave souhaité (7 ou 10 bits), puis d'un bit R/W pour déterminer si le maître (qui a initié la condition S pour la communication) veut lire (R/W=1) ou écrire (R/W=0) à cet esclave ayant cette adresse. Ensuite, si l'esclave existe et fonctionne correctement, il accuse réception au maître en envoyant un bit d'accusé de réception ACK, sinon, il s'agit d'un accusé de réception négatif NACK. Ensuite, l'octet de données est envoyé, suivi d'un accusé de réception de l'esclave. Enfin, le maître peut mettre fin à la communication en envoyant la séquence Stop Condition (P : STOP), la trame de données est illustrée dans la figure II.29 [44].

Plusieurs capteurs intelligents utilisent ce bus, donc nous allons mettre en place une interface I2C sur notre datalogger pour qu'il puisse communiquer avec ce type de capteur. La carte TM32F407VGT6 intègre trois interfaces I2C avec deux modes de fonctionnement standard et rapide, et il peut connecter jusqu'à 48 appareils sur le même bus.

Pour notre application nous avons sélectionné I2C1 en mode master et standard, SCL correspond à la pin PB6 et SDA correspond à la pin PB7 sur la carte. Un exemple de câblage des capteurs ayant une interface I2C avec la carte STM est illustré dans la figure II.30.

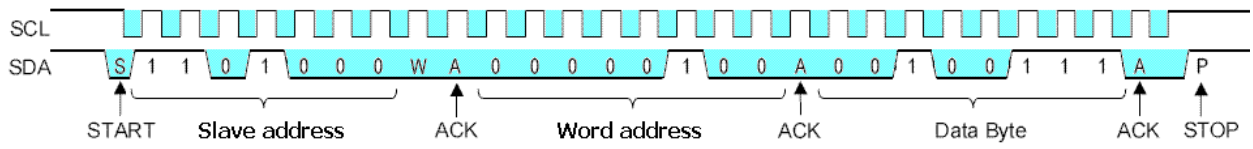


Figure II.28. Principe fondamental d'un transfert de données avec I2C

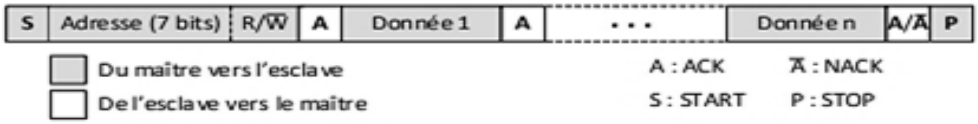


Figure II.29. Trame de données I2C [44]

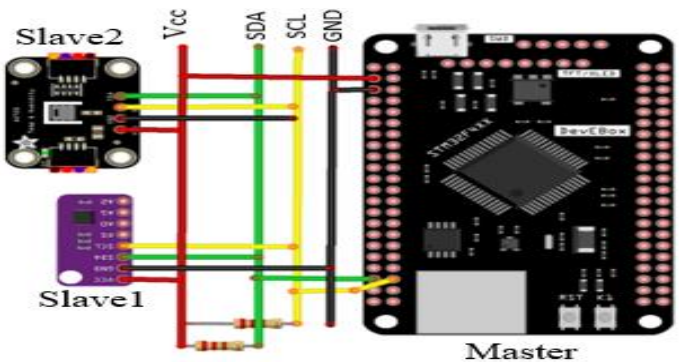


Figure II.30. Schéma de câblage de la carte STM32 avec 2 capteurs utilisant le bus I2C

II.4.4 Transmission de données

Ce bloc est constitué des modules qui permet au Datalogger de communiquer avec l'IHM, et comme nous l'avons déjà cité, le datalogger a deux façons de communication, une filaire USB et l'autre sans fil via LORA qui nécessitera un module de LoRa que nous décrivons ci-après.

II.4.4.1 Interface USB

La carte STM32f407vgt6 est équipée d'un contrôleur USB 2.0 avec un port micro USB prêt à l'utilisation, son schéma électrique est illustré dans la figure II.31, pour l'employer on aura besoin d'un câble USB illustré sur la même figure avec le schéma électrique.

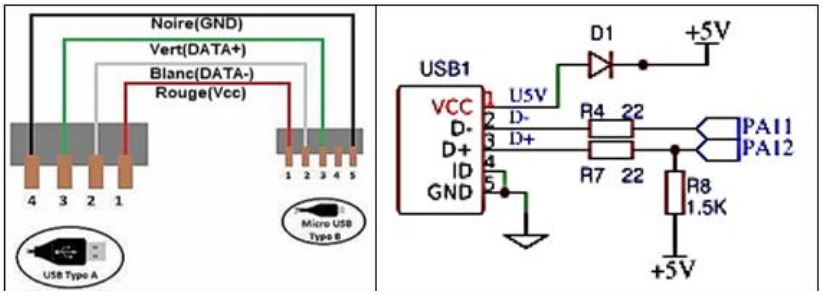


Figure II.31. Pins du câble USB [45] et schéma électrique de l'interface USB de la carte STM32 [46]

II.4.4.2 Module LoRa-02 SX1278

Le module RA-02 est un module LoRa basé sur SX1278 produit par la société Ai-Thinker. Le RA-02 dispose d'un connecteur UFL ou IPEX pour connecter une antenne externe, son avantage permet de connecter de meilleures antennes de gain et de les positionner de manière flexible. Il est blindé pour minimiser les interférences électromagnétiques. Il prend en charge de la modulation LoRa, modulation FSK (Frequency Shift Keying) et OOK (On-Off Keying) [47]. Ses caractéristiques sont résumées dans le (Tableau II.7 et figure II.32).

Tableau II.7 : Spécification de module Ra-02 [47]

Propriété	Valeur
Puce LoRa	Semtech SX1278
Colis	Module SMD-16
Taille	17 x 16 x 3,2 (±0,2) mm
Gamme de fréquences	410 ~ 525MHz
Communication Interface	SPI
Fréquence SPI	10 MHz
Débit programmable	Jusqu'à 300 kbps
Puissance d'émission maximale	20±1 dBm
Sensibilité du récepteur	-140 dBm
Tension de travail	2.7 ~ 3.6V, 3.3V typique
Courant de travail	140 mA
Courant de réception minimum	4,5 mA
Courant de veille	1,6 mA
Température de fonctionnement	-20~+70°C
Environnement de stockage	-40 ~ +125°C, <90%HR
Poids	0,45 g

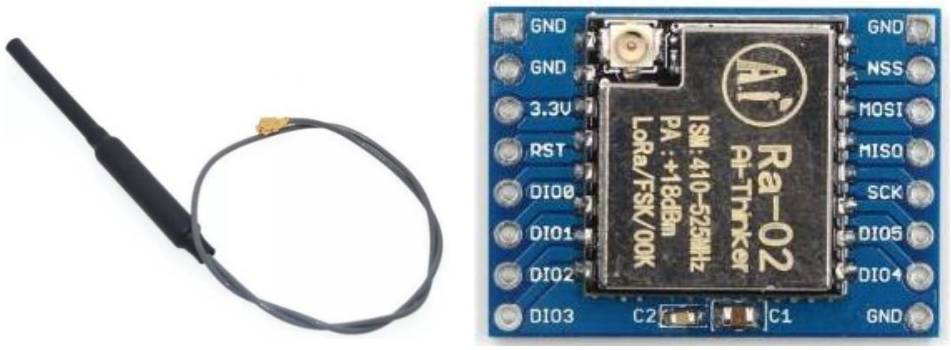


Figure II.32. Illustration de module Ra-02 et d'antenne IPEX employé avec lui [47]

Ce module LoRa est équipé d'une interface SPI, ce qui permet de le relier à la carte STM32f407vgt6 qui est équipée de 3 interfaces SPI, pour cette application nous allons employer le SPI1. Donc ce module prend les données du datalogger via l'interface SPI et les envoie vers l'IHM via une antenne IPEX, après l'opération de modulation LoRa faite par ce module et ceci dans le mode émetteur, le contraire sera effectué dans le mode récepteur.

Le tableau II.8 décrit les différentes pines du module LoRa avec son câblage correspondants aux pines de la carte STM32 et la figure II.33 décrit le schéma électrique de cette liaison.

Tableau II.8 : Les pins de brochage du module Ra-02 à la carte STM32f407VGT6

Module LoRa–02 SX1278	Description	STM32F407VGT6
GND	Ground	GND
3.3V	Alimentation 3.3V	3.3V
RESET	Réinitialisation	PC5
DIO0	IO0 numérique, configuration logicielle	PC4
DIO1	Configuration du logiciel Digital IO1	X
DIO2	Configuration du logiciel Digital IO2	X
DIO3	Configuration du logiciel Digital IO3	X
GND	Ground	X
DIO4	Configuration numérique IO 4 pièces	X
DIO5	Configuration du logiciel Digital IO5	X
SCK	Entrée d'horloge SPI	PA5
MISO	Sortie de données SPI	PA6
MOSI	Entrée de données SPI	PA7
NSS	Entrée de sélection de puce SPI	PA4
GND	Ground	X

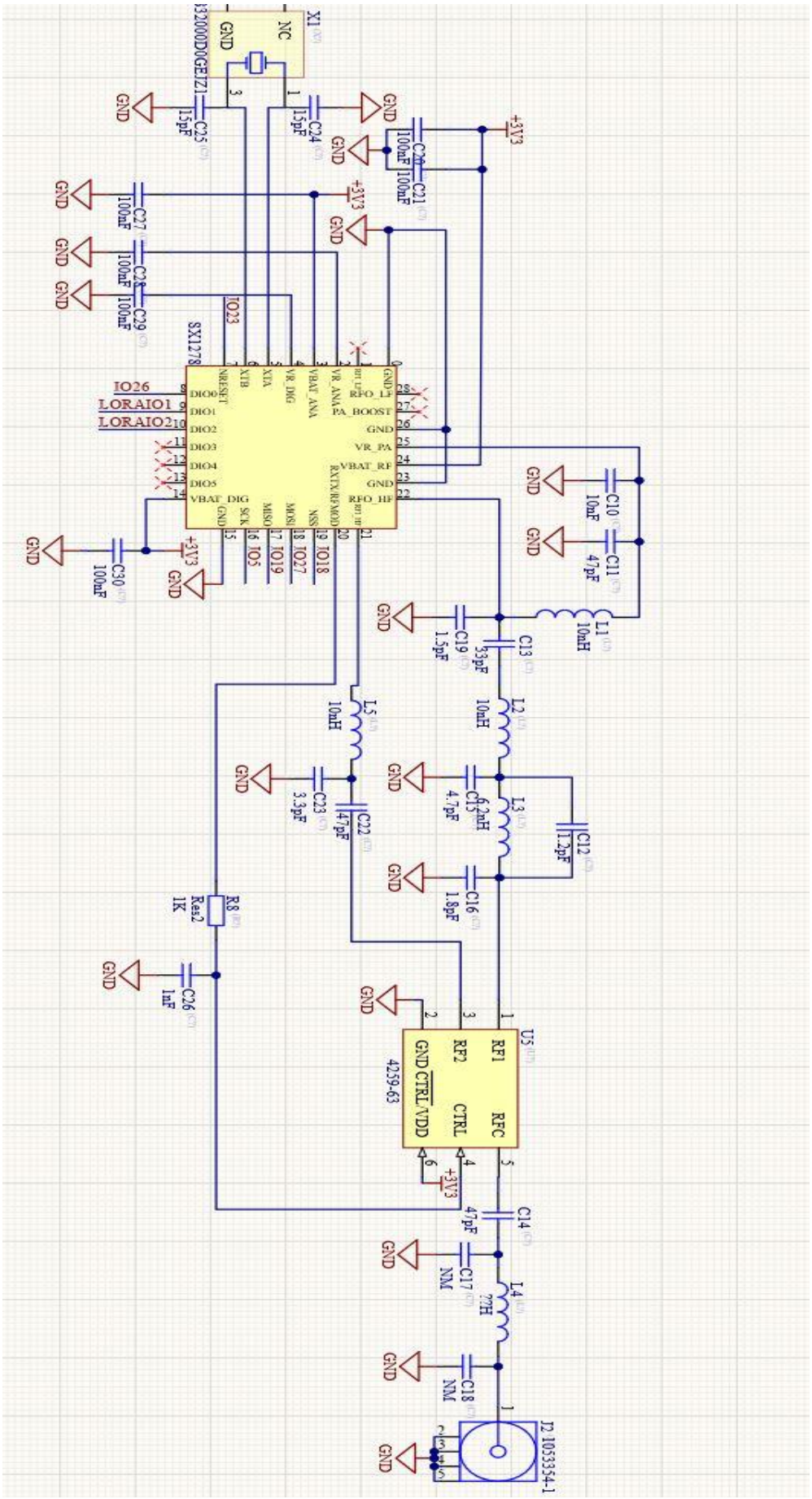


Figure II.33. Schéma électrique de module Lora [48]

II.4.5 Alimentation

La figure ci-dessous représente un schéma simplifié d'un circuit d'alimentation électrique typique, pour le DataLogger, dont 3.3volts pour alimenter la carte STM32, et le 7volts pour le module de conversion 4-20mA à tension, les autres modules peuvent être alimenté par la carte.

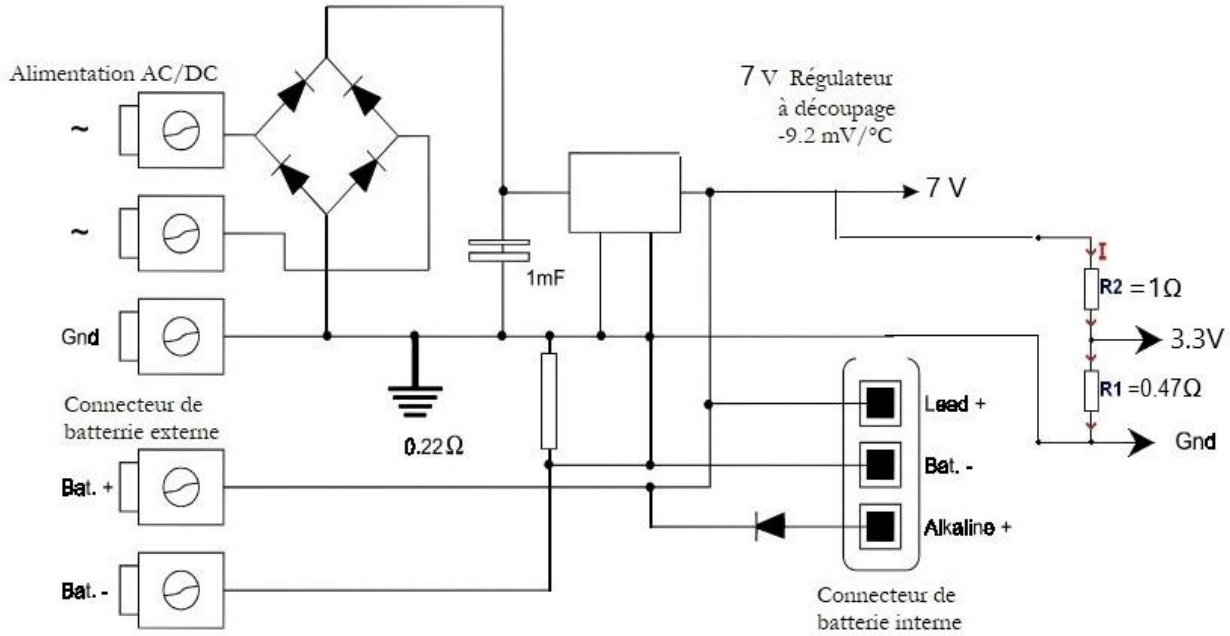


Figure II.34. Schéma simplifie de l'alimentation électrique de DataLogger

Lorsqu'une batterie alcaline non rechargeable interne et une source d'alimentation externe AC ou DC sont connectées simultanément, la tension de sortie du régulateur est augmentée pour éviter de puiser dans la batterie interne. Les batteries de gel scellées peuvent être rechargées via une source d'alimentation AC/DC ou un panneau solaire de 7 Volts, en fonction des heures d'ensoleillement et des besoins en charge. Enfin, la durée de vie maximale de la batterie dépend de divers facteurs tels que la fréquence de balayage des canaux d'entrée, le nombre de canaux analogiques et numériques connectés, la puissance d'excitation des capteurs, la complexité des calculs, etc.

❖ **Consommation énergétique estimée de système**

Utilisant cubeMX on peut estimer la consommation énergétique de système en sélectionnant le type de batterie utilisée dont ses caractéristiques :

- Alkaline (D LR20)
- Capacité : 20500 mAh
- Décharge : 0.3%/mois

Ainsi on ajoute la consommation des autres composants qu'on a ajouté sans compter les capteurs :

- Convertisseur TTL/RS485 : 0.01Ua
- Convertisseur 4-20mA/0-3.3V : 15mA
- MCP2551 : 0.01mA
- Module LoRa Ra-02 : 10.8mA

On obtient le résultat suivant :

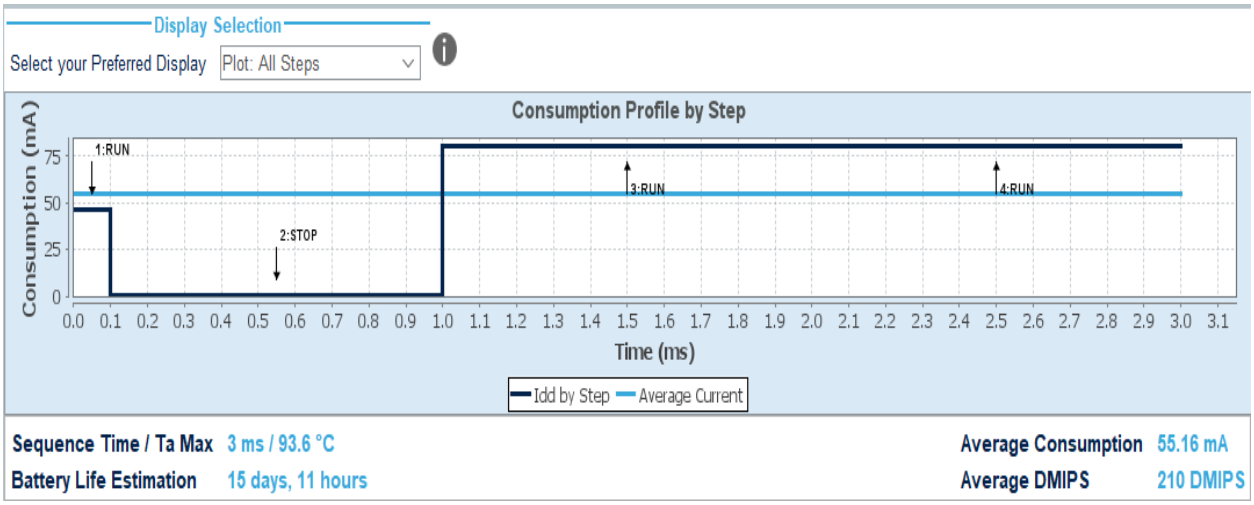


Figure II.35. Estimation de la consommation énergétique de DataLogger

L'estimation de vie de la batterie est de 15 jours et 11 heures pour ceci on recommande d'ajouter des panneaux solaires pour recharger la batterie.

II.5 Description matérielle de l'IHM (Interface Homme Machine)

Pour construire une interface de visualisation des données enregistrée par le datalogger, nous avons deux solutions, soit utiliser le PC avec un adaptateur LoRa vers USB ou bien utiliser Raspberry Pi qui équipé d'une interface SPI en plus de port USB, ce qui permet de relier le module LoRa directement avec la carte. Comme premiers tests, nous avons opté pour la deuxième solution.

II.5.1 Raspberry Pi 3

Le Raspberry Pi est un nano-ordinateur mono-carte à processeur ARM, conçu par le créateur de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi. Cet ordinateur, qui a la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique : il permet l'exécution de plusieurs variantes du système d'exploitation (OS) libre GNU/Linux-Debian et des logiciels compatibles. Mais également avec les Microsoft Windows : Windows10, loT Core et Android Pi. Il est fourni avec une carte mère seule, sans boîtier, ni alimentation, ni clavier, ni souris et ni écran dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération [50].

- ✓ Processeur : 64 bits Quad-core ARMv8 Cortex A-53 cadencé à 1,2 GHz,
- ✓ 1 GB RAM
- ✓ Wi-Fi LAN 802.11 b/g/n intégré, jusqu'à 70 Mb/s,
- ✓ Bluetooth 4.1 intégré,
- ✓ Port HDMI 1.3a,
- ✓ Port Ethernet 10/100,
- ✓ Prise casque 3.5 et composite vidéo,
- ✓ Alimentation : 5V, 2.5A,
- ✓ 4 ports USB 2.0,
- ✓ 1 port micro USB,
- ✓ Stockage : micro SD.

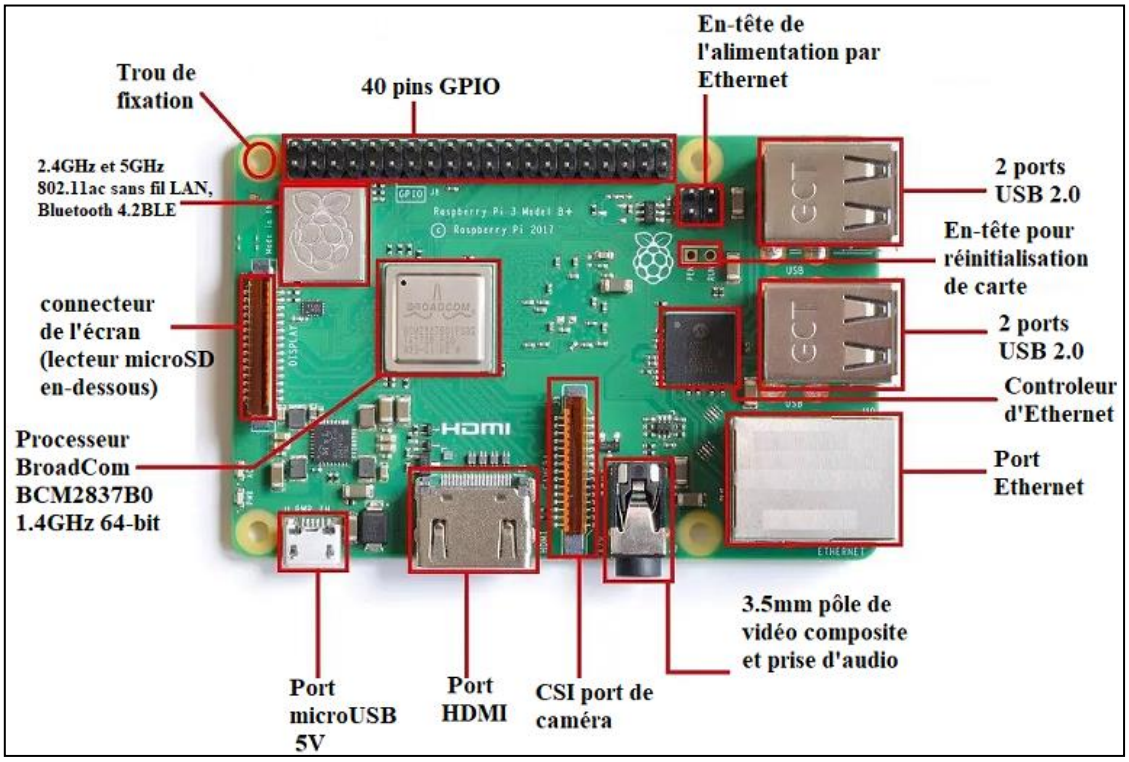


Figure II.36. Raspberry Pi 3 et ses modules d'E/S [50]

Nous avons besoin d'autres modules ou matériel pour faire fonctionner la carte Raspberry Pi 3 qui sont : un chargeur de type B qui délivre 5V pour l'alimentation de la carte, et une carte mémoire de 32Go pour le système d'exploitation.

Pour le module LoRa, nous allons utiliser le deuxième module de même type que celui décrit dans la section précédente pour réaliser une communication entre l'IHM et le datalogger.

II.5.2 Schéma fonctionnel de l'IHM

Le Ra-02 SX1278 est un module à 16 broches qui communique à l'aide de SPI sur 3.3V Logic. Le Raspberry pi fonctionne également au niveau logique 3.3V et dispose également d'un port SPI intégré et d'un régulateur 3.3V. Nous pouvons donc connecter directement le module LoRa avec le Raspberry Pi. La table de connexion est illustrée ci-dessous ainsi que le schéma de câblage illustré dans la figure II.36.

Tableau II.9 : Pines de branchement du module LoRa Ra-02 avec la carte Raspberry

Raspberry Pi 3	Module LoRa–02 SX1278
3.3V	3.3V
Ground	Ground
GPIO 10	MOSI
GPIO 9	MISO
GPIO 11	SCK
GPIO 8	Nss / Enable
GPIO 17	DIO 0
XX	DIO 1
XX	DIO 2
XX	DIO 3
GPIO 4	RST

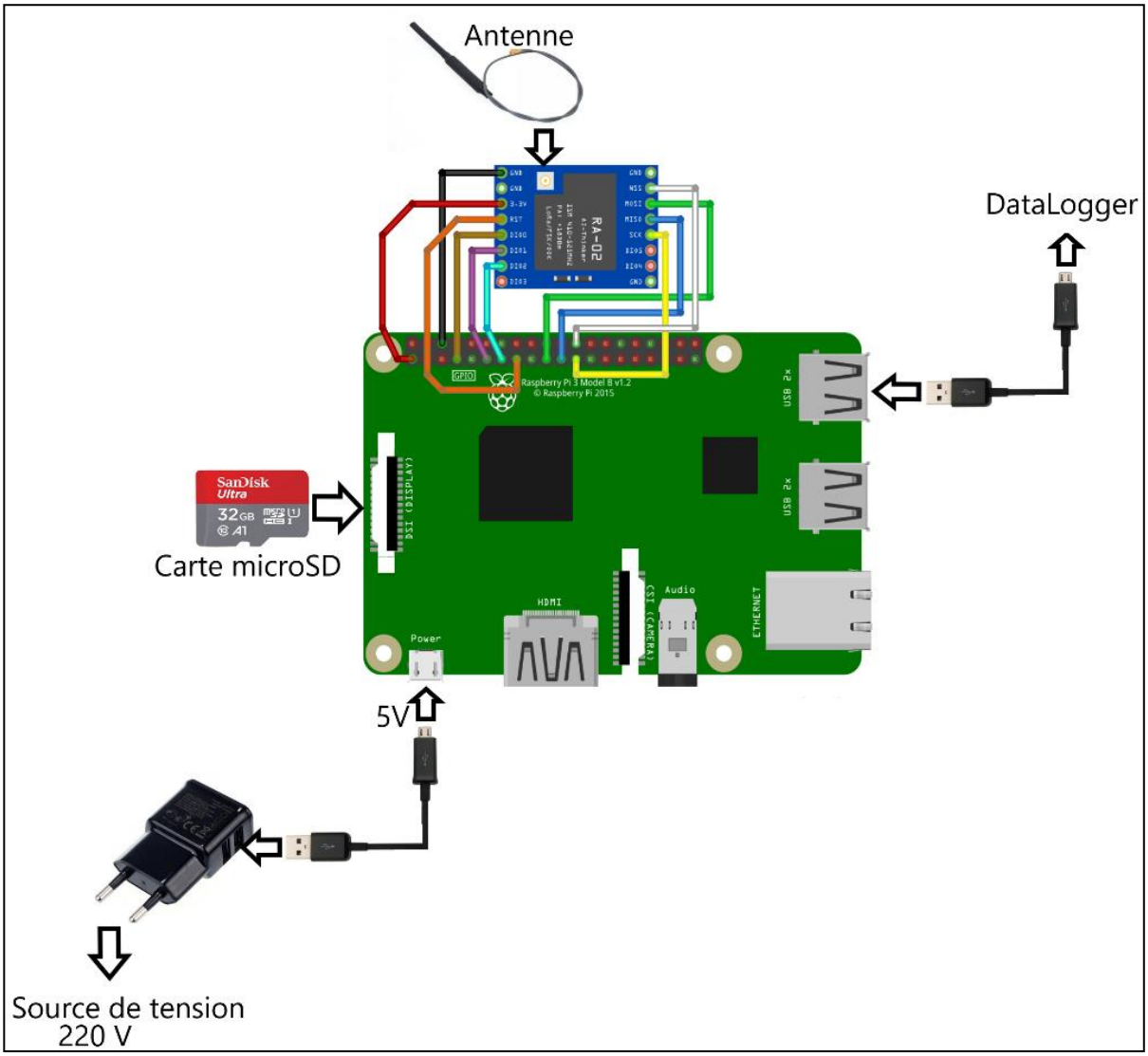


Figure II.37. Schéma fonctionnel de L'IHM

II. 6 Conclusion

Dans ce chapitre nous avons décrit les différents aspects matériels de notre système conçu, de la carte d'acquisition, à la carte de traitement STM32, aux différents bus et modules de communication, au module LoRa, puis l'interface IHM réalisée en se basant sur le module Raspberry avec ses différents modules nécessaires pour son fonctionnement.

Dans le chapitre suivant, nous allons nous focaliser sur l'aspect logiciel et les différents tests que nous avons pu réaliser pour mettre en œuvre notre système conçu de datalogger ainsi que les résultats obtenus.

Chapitre 3

Implémentation et mise en œuvre du système de Datalogger

Chapitre III

Implémentation et Mise en Œuvre du Système de DataLogger

III.1. Introduction

Après avoir décrit les exigences et l'aspect matériel de notre système, nous allons passer à sa partie implémentation et mise en œuvre, tout en passant par plusieurs étapes.

Dans ce chapitre, nous allons décrire les différentes étapes de réalisation de notre projet en commençant par une description des logiciels utilisés et l'organigramme de fonctionnement global de notre système, nous enchainons après par les étapes de programmation et de tests du datalogger, ainsi que la description des étapes de conception de notre IHM, avant de procéder enfin à l'illustration des résultats obtenus en utilisant quelques capteurs parmi ceux que nous avons déjà décrits en chapitre précédent.

III.2. Description des logiciels utilisés

La partie logicielle de notre système est constituée de deux grandes parties, la partie relative au datalogger, ses logiciels de fonctionnement, de sa programmation, et la partie concernant son interfaçage ou qu'on appelle l'IHM, chaque partie nécessite des logiciels différents, comme nous allons les décrire tous dans l'ordre ci-dessous.

Les logiciels seront installés sur un PC acer, avec un processus Intel Core i3-5005U (2.2 GHz, 3MB L3 Cache), travaillant sous un système d'exploitation Windows 10.

III.2.1. Partie datalogger

Afin de programmer le cœur du datalogger, nous avons choisi d'utiliser les logiciels : CubeMx et CubeIDE pour plusieurs raisons :

- ✓ Ce sont des logiciels offerts par le constructeur de la série des MCUs STM32,
- ✓ Open source,
- ✓ Permettent une compréhension profonde du système grâce à l'outil CubeMX qui décrit toutes les caractéristiques des modules de système et leurs liaisons avec le MCU,
- ✓ La programme des différents modules et leur configuration se fait en langage C, ce qui nous facilite le travail.

III.2.1.1. STM32CubeMX

STM32CubeMX est un outil de configuration graphique qui simplifie et automatise la configuration et la génération de code C pour l'initialisation des microcontrôleurs STM32, qui est ainsi prêt à l'emploi dans plusieurs environnements de développement, parmi eu le CubeIDE.

III.2.1.2. CubeIDE

C'est un environnement de développement intégré (IDE) professionnel, issu de l'acquisition en 2017 de l'éditeur suédois Atollic et accessible dans le cadre d'une licence ouverte. Il dispose des caractéristiques propres aux STM32 pour accélérer la mise au point de conceptions embarquées bâties sur les microcontrôleurs du fabricant franco-italien et intègre notamment l'outil STM32CubeMX qui permet de configurer le circuit cible et de gérer la mise au point d'un projet, tel que : la génération de code, la compilation de code et le débogage.

III.2.2. Partie IHM

Pour construire l'IHM avec la Raspberry Pi, on doit préparer son environnement, qui est la carte Raspberry qui nécessitera un système d'exploitation dédié aux systèmes embarqués, plusieurs choix existent, mais nous avons opté pour le Rasbian, et pour sa configuration nous aurons besoin d'autres programmes et outils que nous décrivons ici et les étapes dans une autre section. Après la préparation de l'environnement de développement, nous allons concevoir notre IHM en utilisant le serveur WEB Node-red.

III.2.2.1. Raspberry Pi Imager

Raspberry Pi Imager est un utilitaire officiel développé par la Fondation Raspberry Pi pour faciliter le processus de création d'une image du système d'exploitation pour les appareils Raspberry Pi sur une carte micro SD. Il simplifie la configuration d'un Raspberry Pi en fournissant une interface conviviale pour sélectionner et créer des images de différents systèmes d'exploitation sur la carte micro SD. La figure III.1 montre l'interface graphique du Raspberry Pi Imager.

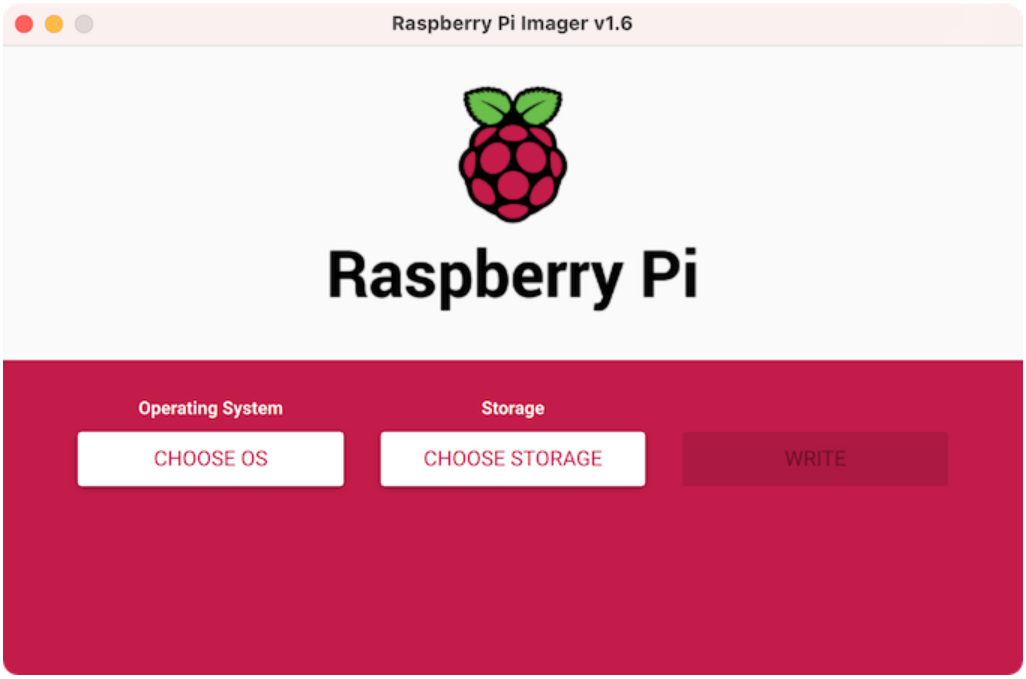


Figure III.1. Illustration de l'interface du logiciel Raspberry Pi Imager

III.2.2.2. CMD Windows

L'invite de commandes Windows, souvent abrégée en "CMD", est une interface en ligne de commande intégrée dans le système d'exploitation Windows. Elle permet aux utilisateurs d'interagir avec le système d'exploitation en saisissant des commandes texte. L'invite de commandes est un outil puissant qui permet d'effectuer une variété de tâches système, de manipuler des fichiers et des dossiers, d'exécuter des programmes et de configurer des paramètres système.

III.2.2.3. VNC viewer

VNC Viewer, abréviation de "Virtual Network Computing Viewer", est un logiciel qui permet de visualiser et de contrôler à distance un ordinateur ou un appareil à partir d'un autre ordinateur. Il fait partie du système VNC, qui est une technologie permettant l'accès et le contrôle à distance d'un ordinateur par le biais d'une interface graphique.

III.2.2.4. Node-red

Node-RED est une plateforme de développement visuelle open source qui permet de créer des applications et des flux de données en utilisant une interface graphique. La plateforme est basée sur Node.js, ce qui lui permet de s'exécuter facilement sur différents systèmes d'exploitation, on peut y accéder à partir d'une adresse IP locale.

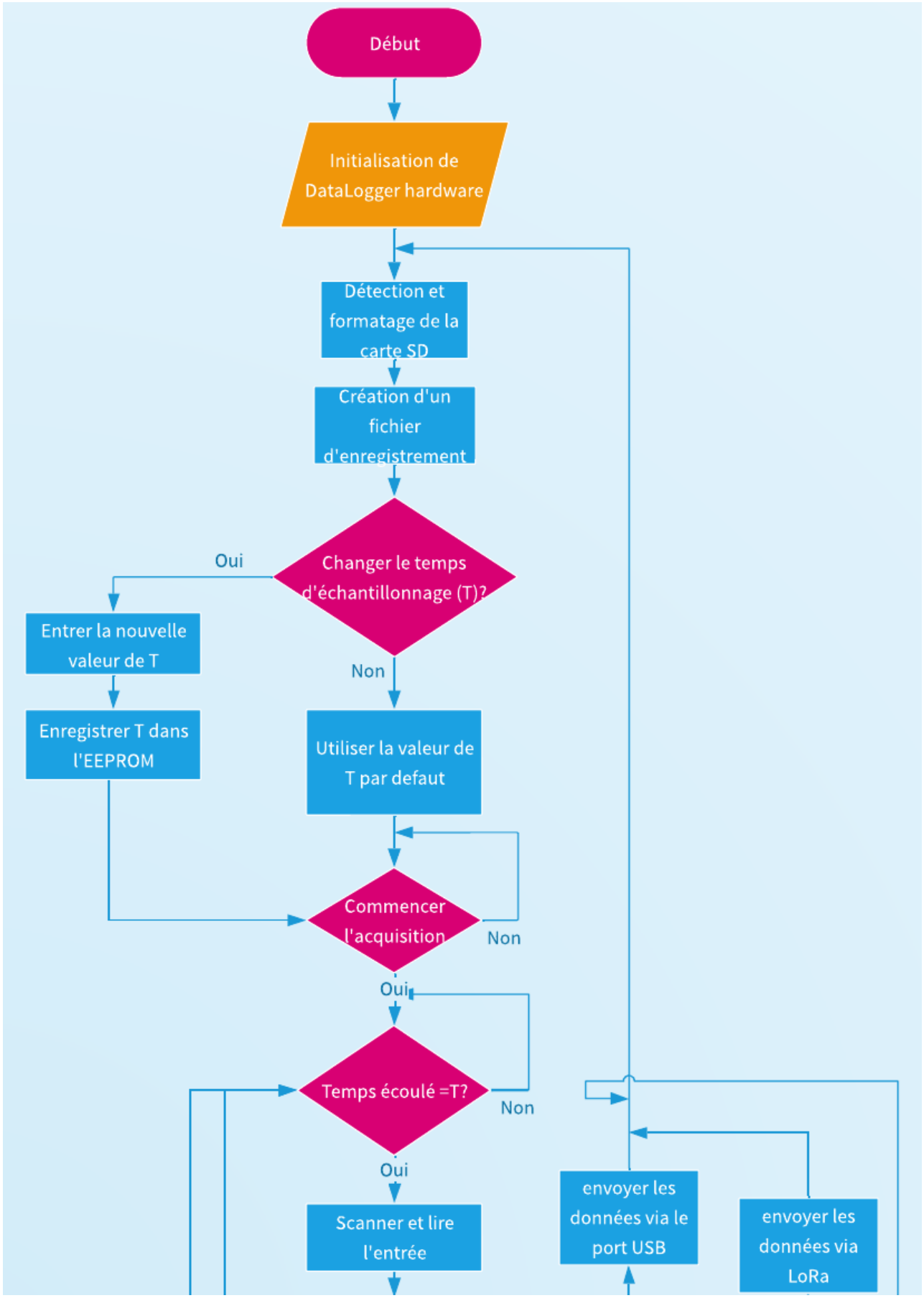
Il utilise un paradigme de "glisser-déposer" pour créer des nœuds (nodes) et les connecter afin de former un flux de données. Chaque nœud représente une fonction ou une action spécifique, telle que la lecture de données à partir d'une source, la transformation des données, leur envoi à une destination ou l'exécution d'une opération spécifique. Ces nœuds peuvent être combinés pour construire des flux de données complexes et des applications IoT (Internet des objets).

III.3. Principe de fonctionnement du DataLogger

La fonction principale du datalogger est de scanner ses entrées selon un taux d'échantillonnage bien défini, qu'on configure depuis l'IHM, et d'enregistrer les données acquises sur la carte SD, dans un fichier créé au début de programme. Ce fichier contient les informations suivantes : la date et le temps d'acquisition de chaque donnée, ainsi le canal correspondant.

Au cours du fonctionnement du datalogger, le système peut communiquer avec l'IHM, tel que l'envoi d'une alerte à l'utilisateur lors de la nécessité de téléchargement de données, ou bien d'autres alertes liées aux conditions de grandeurs mesurées.

L'organigramme de fonctionnent du système est illustré par la figure III.2 suivante.


Début
Initialisation de DataLogger hardware
Détection et formatage de la carte SD
Création d'un fichier d'enregistrement
Changer le temps d'échantillonnage (T)?
Oui
Non
Entrer la nouvelle valeur de T
Enregistrer T dans l'EEPROM
Utiliser la valeur de T par defaut
Commencer l'acquisition
Non
Oui
Temps écoulé =T?
Non
Oui
Scanner et lire l'entrée
envoyer les données via le port USB
envoyer les données via LoRa

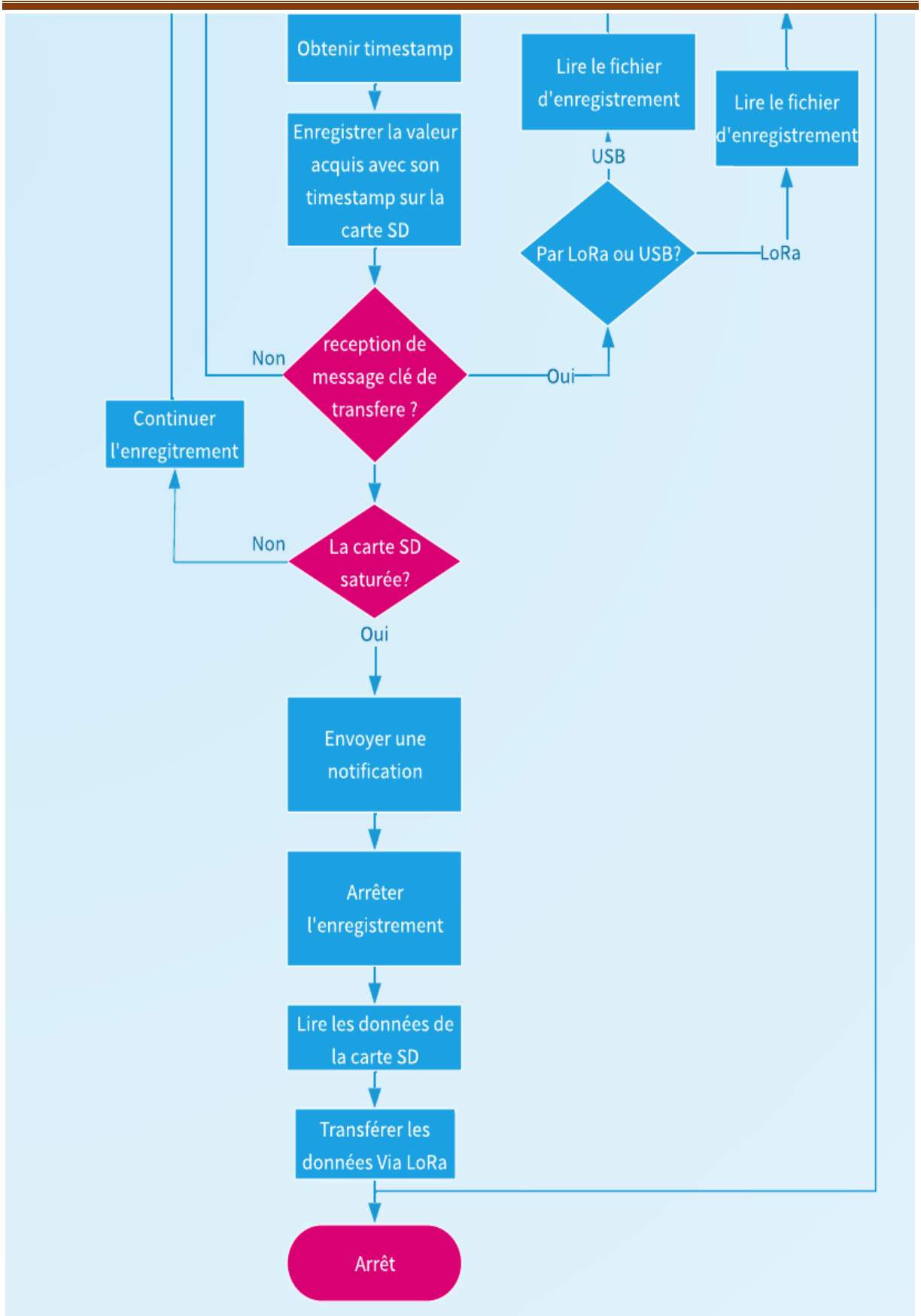


Figure III.2. Organigramme de fonctionnement principal du DataLogger

III.4. Etapes de réalisation du DataLogger

III.4.1. Configuration du MCU sur CubeMX

La première étape du travail consiste à configurer toutes les pins et les modules que nous allons utiliser sur le MCU, qui correspond à celui qui se trouve sur la carte que nous avons choisi d'utiliser STM32F407VGT6, et dont les critères de choix et explications relatives sont détaillés dans le chapitre précédent. Mais, avant tous, nous allons sélectionner les modules déjà liés sur la carte en suivant la description de la carte sur le site [51]. Le tableau suivant décrit les différentes configurations que nous avons faites en utilisant cubeMX version 6.8.1 et la figure III.3 décrit les différents pins sélectionnés sur le MCU.

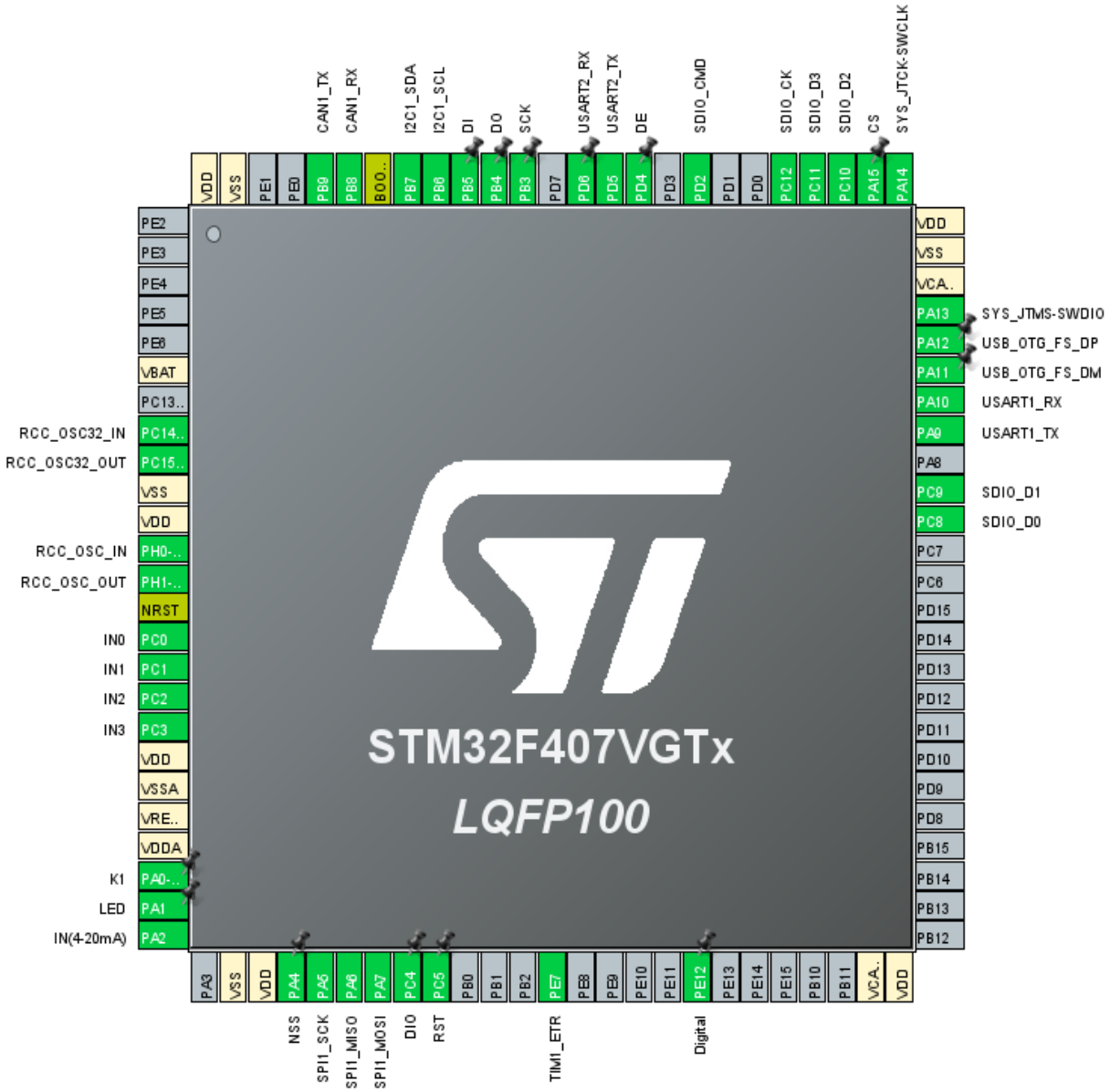


Figure III.3. Les pins sélectionnés sur le MCU

Tableau III.1. Configuration du MCU

Module		**Explication de configuration**
GPIO	Botton (K1)	Il se trouve sur la carte que nous allons utiliser, donc de préférence les activer, la LED sera utilisée lors de programmation pour détecter les erreurs, sa configuration est illustrée dans la figure III.4.
	Led (D1)	
	DE	Cette pine est utilisée pour l'activation de mode transmission dans le module RS485, sa configuration est illustrée dans la figure III.4.
	Digital pin	Détection d'état ON de la pine, lors d'activation une interruption sera générée, sa configuration est illustrée dans la figure III.4.
	NSS	Pour le module LoRa, déjà expliqué dans le chapitre II, sa configuration est illustrée dans la figure III.4.
	RST	
	DIO	
EEPROM		Intégrés sur la carte de développement via SPI3, donc on active le module SPI3 dans le mode full-duplex master, avec un débit en baud de 5Mbits/s obtenu en configurant le diviseur d'horloge (Prescalar) à 8, ainsi une taille de donnée de 8bits.
System Debug		Pour programmer le MCU via st-Link, on choisit dans la section debug une connexion série (serial wire).
ADC1/ADC2		Comme nous avons déjà expliqué dans le chapitre II, l'utilité du module ADC et le mode souhaité, à ce stade on vous montre la configuration sur cubeMX illustré sur la figure II.5 pour l'ADC1 et sur la figure II.6 pour l'ADC2, ainsi on active le mode DMA pour les deux modules en mode normal. En plus, on a sélectionné le TIMER2 et le TIMER5 pour l'ADC1 et l'ADC2 respectivement pour synchroniser l'enregistrement des données, configuré dans le champs « External Trigger source » qui synchronise le début de conversion d'un groupe de canal sélectionné, selon le taux d'échantillonnage configuré dans chaque TIMER, qui est diffèrent du taux d'échantillonnage de l'ADC « sampling Time » qu'on le met à 144 cycles pour chaque canal, ce qui permet de calculer le temps nécessaire d'une conversion utilisant la formule suivante : $\text{Tconv} = \text{Sampling time} + 12.5\ \text{cycles} = 144 + 12.4 = 156.4\text{cycles}$ On a: $ADCclk = \frac{84MHz}{2} = 21MHz \rightarrow Tconv = \frac{156.4}{21} = 7.44\mu s$
RCC (Reset and Clock Controller)		On active la source d'horloge externe, pour mettre HCLK au maximum 168MHz pour utiliser toutes les performances de notre système, la figure III.7 vous montre la configuration d'arbre d'horloge.
I2C1		La configuration de ce module est expliquée dans le chapitre II, à ce stade on applique la configuration sur cubeMX comme est illustré sur la figure III.8.
SPI1		STM32 fournie plusieurs modes d'opération le bus SPI, selon le mode de communication full-duplex ou half-duplex ou bien en tant que master ou slave et transmetteur ou receveur, pour notre cas on sélectionne le mode master pour qu'on

	puisse gérer la communication avec l'autre module et full-duplex pour l'échange d'information des deux côtés, le module LoRa permet une communication avec un débit de 10Mbits/s, donc on a configuré le diviseur de clock de façon à avoir cette valeur, ainsi la taille de donnée 8 bits, comme illustré sur la figure III.9.
CAN1	STM32 fournit plusieurs modes d'opération pour le CAN ainsi des débits différents qu'on peut les changer en modifiant le diviseur d'horloge (prescalar) et le quanta de temps des différents segments, avec la configuration qu'on a fait illustrer dans la figure III.10, ça donne un temps de 10000 ns pour un bit et un débit en bauds de 100000 bits/s.
USART1/ USART2	USART1 et USART2 sont configuré en mode asynchrone, et on active aussi le mode DMA pour en ajoutant RX, la configuration souhaitée qu'on a appliquée sur cubeMX est illustrée dans la figure III.11 pour UART1 qui est la même avec celle de l'UART2.
USB-OTG-FS (USB On The Go-Full speed)	Comme on a mentionné dans le chapitre précédent, on a besoin de connecter le MCU au PC pour transférer les données, pour ceci on utilise le module USB OTG-FS et le configurer en mode « device only », ce qui permet de basculer entre agir en tant qu'hôte et en tant que subordonné afin qu'ils puissent être connectés directement au PC et échanger les données, avec une vitesse de 12Mbits/s, sa configuration sur cubeMX est illustrée sur la figure III.12.
SDIO	Est un format de carte mémoire non-volatile développé par l'association SD Card pour les appareils portables ce qui permet la lecture des cartes SD sans l'utilisation des pilotes, sur STM32 fournit deux modes d'opération pour une carte SD 1 bit et 4 bits wide, on sélectionne 4 bits wide comme illustré sur la figure III.13, on active aussi le mode DMA et interruption.
RTC	C'est un compteur d'horloge qui garde la trace sur l'heure et la date actuelle, les cartes STM32 ont RTC intégré pour l'activer il suffit de la configurer sur cubeMX en tant qu'active, comme illustré sur la figure III.14.
TIMER1	C'est un timer à usage général, on l'active pour calculer la fréquence d'une source externe, tel que lors de changement de fréquence, il capte ce changement et l'enregistrer, sa configuration est illustrée sur la figure III.15.
TIMER2	Comme on a déjà mentionné, TIMER2 est la source de trigger pour l'ADC1 sa configuration est illustré sur la figure III.16. On a la formule suivante : $T = \frac{PSC*Preload}{Fclk}$ qui nous permet de calculer la période du timer, pour notre cas on Fclk=84MHz, tel que le source d'horloge pour le timer2, 3 et 6 est APB1, et PSC=42000, pour changer le taux d'échantillonnage de l'ADC1, il suffit de changer la valeur du registre preload qui peut aller jusqu'à **4 294 967 295**, Pour 2000 ça donne T= 1s.
TIMER5	On configure le TIMER5 comme source de trigger de l'ADC2, de la meme façon que celle de TIMER2, on peut changer la valeur de registre preload pour changer le taux d'échantillonnage de l'ADC2, qui peut aller jusqu'à **4 294 967 295.**
TIMER6	C'est un timer basique, on le programme pour générer une interruption chaque une seconde pour régler le taux d'échantillonnage des entrées séries, voir la figure III.17. La méthode de calcul et la même que celle mentionnée dans TIMER2.

FATFS	Est un module de système de fichier FAT générique pour les systèmes embarqués, donc on l'active pour générer la bibliothèque des fichiers qui nous permet d'écrire et lire de la carte SD via l'interface SDIO, voir la figure III.18 pour la configuration.
USB-DEVICE	C'est une bibliothèque de périphérique USB, propose des API pour accéder aux périphériques USB de différentes classes, voir la figure III.19 pour la configuration.

1. Module GPIO

La configuration des pines sélectionnées sur le MCU est illustrée dans la figure III.4.

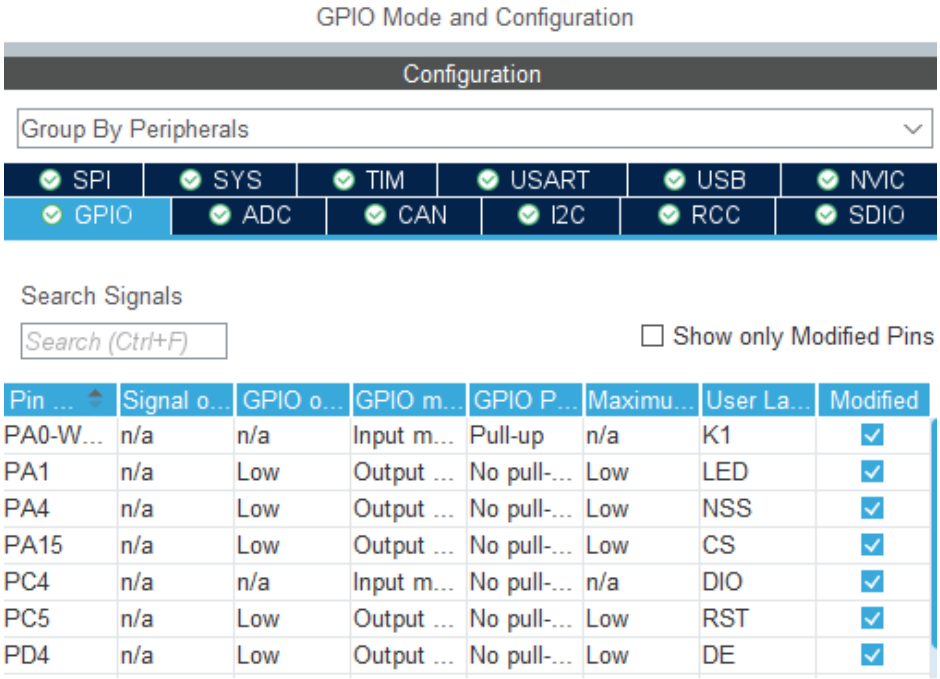


Figure III.4. Configuration de GPIO sur cubeMX

2. Module ADC1 et ADC2

La configuration de l'ADC1 et l'ADC2 sur cubeMX, dont on sélectionne la résolution, le nombre de canaux utilisés ainsi le mode de conversion, et dans les figures ci-dessous on montre bien cette configuration et les pines correspondantes :

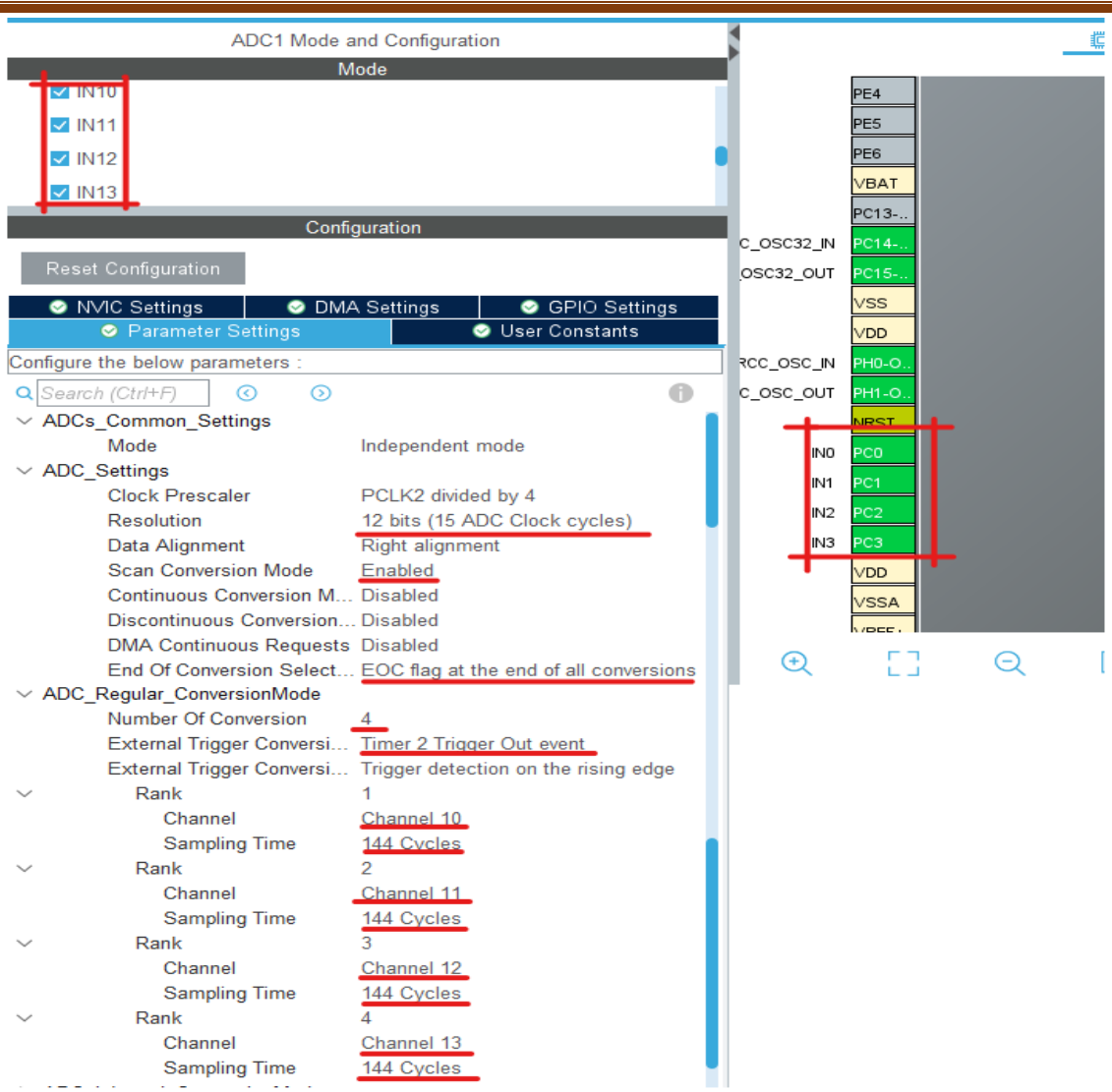


Figure III.5. Paramètres de configuration de l'ADC1 sur cubeMX

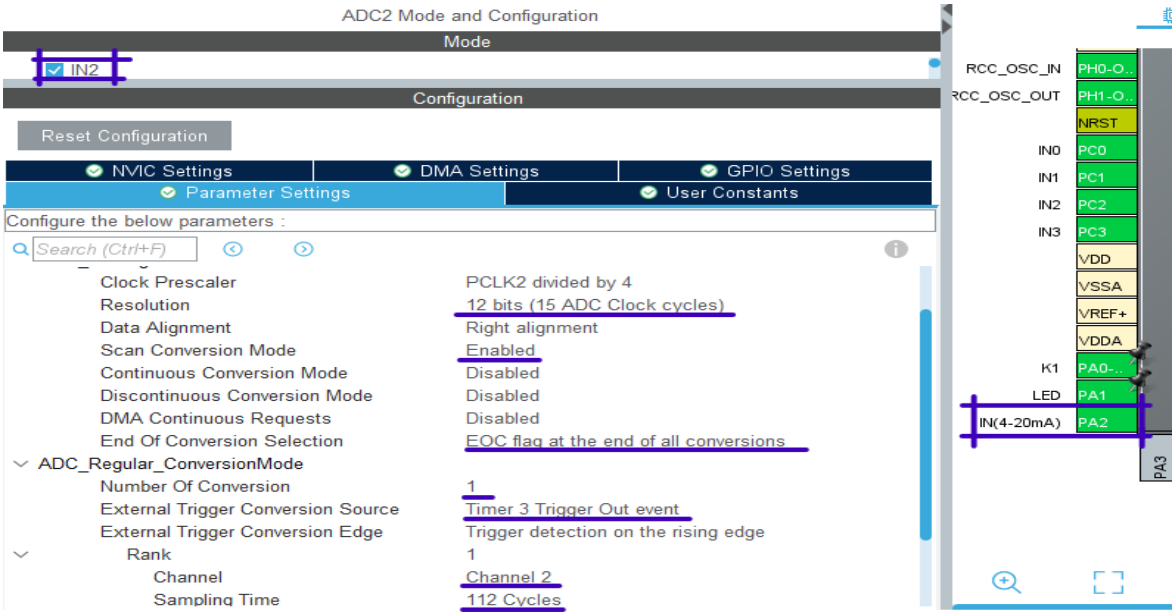


Figure III.6. Paramètre de configuration de l'ADC2 sur cubeMX

3. Arbre d'horloge

La figure suivante monte la configuration finale de l'arbre d'horloge qui montre la fréquence utilisée pour chaque module de MCU :

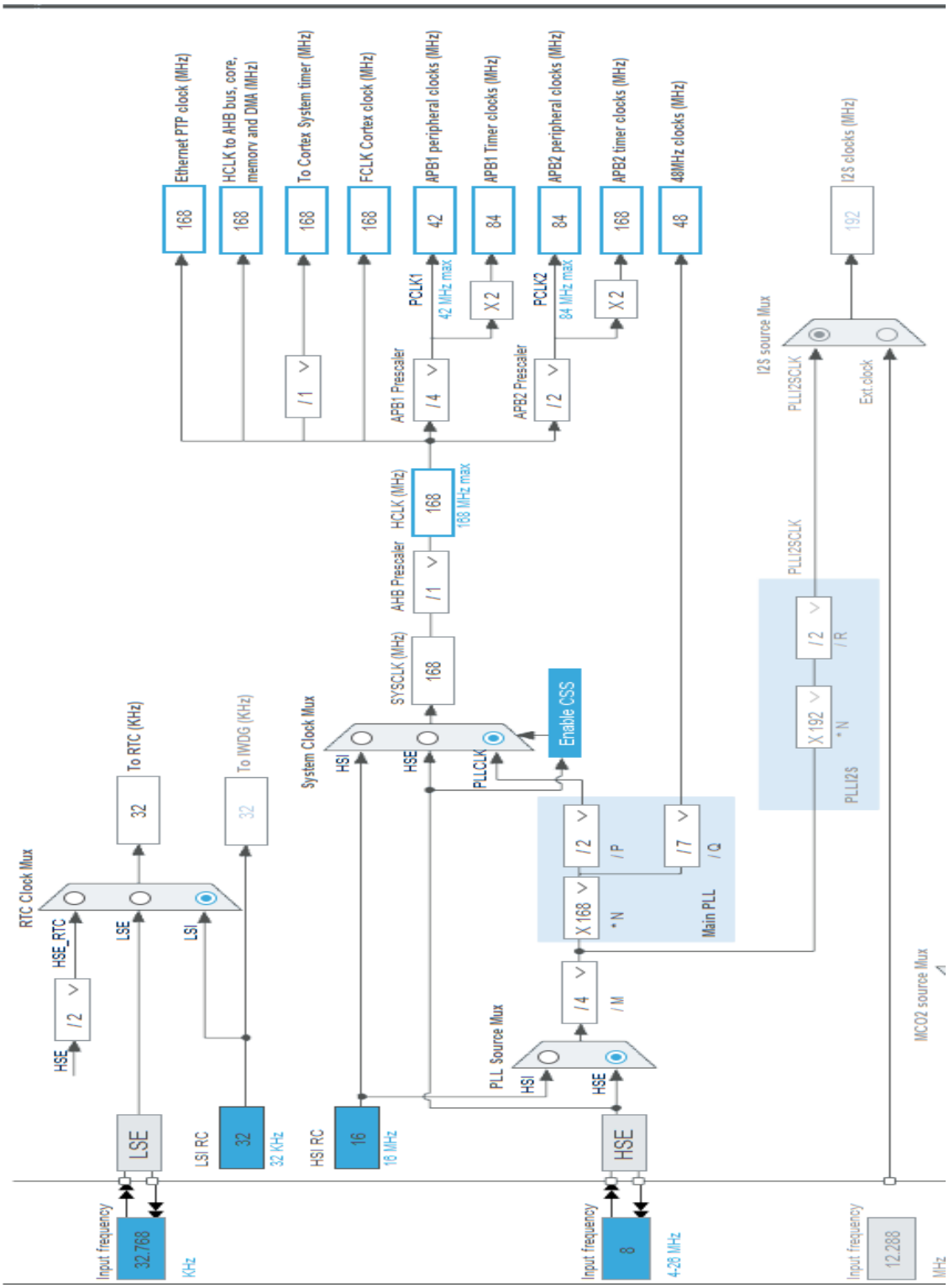


Figure III.7. Arbre d'horloge du système

4. Module I2C

La figure suivante vous montre la configuration de l'I2C dont le mode, la vitesse, et le nombre de bit d'adresse primaire utilisé ainsi les pines correspondantes à ce module.

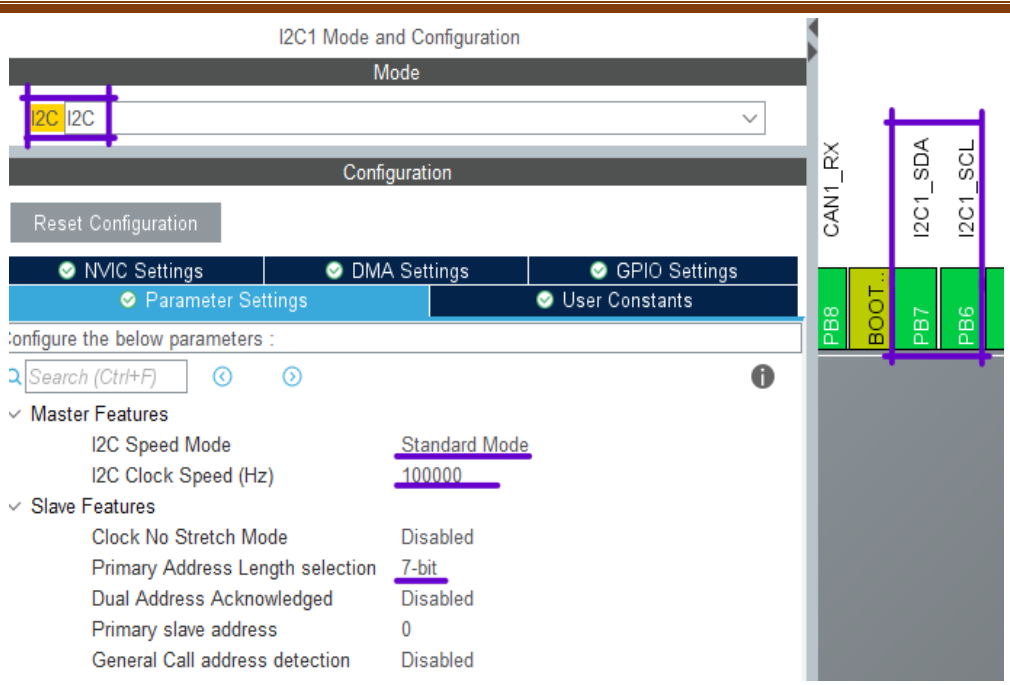


Figure III.8. Configuration de l'I2C1 sur cubeMX

5. Module SPI

La figure suivante montre la configuration de module SPI sur cubeMX, dont on a sélectionné la forme de trame, la taille de donnée, la vitesse et les pines correspondantes à ce bus.

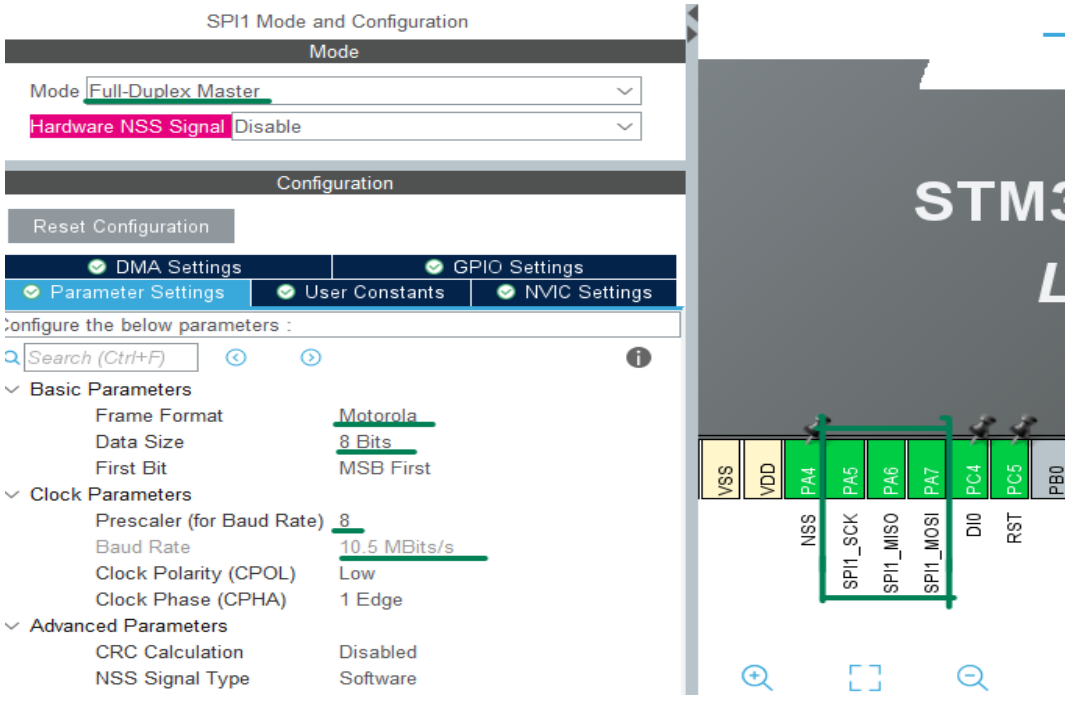


Figure III.9. Configuration de SPI1 sur cubeMX

6. Module CAN

La configuration sélectionnée sur cubeMX pour le bus CAN, ainsi les pines correspondantes à ce bus sont illustrées dans la figure suivante :

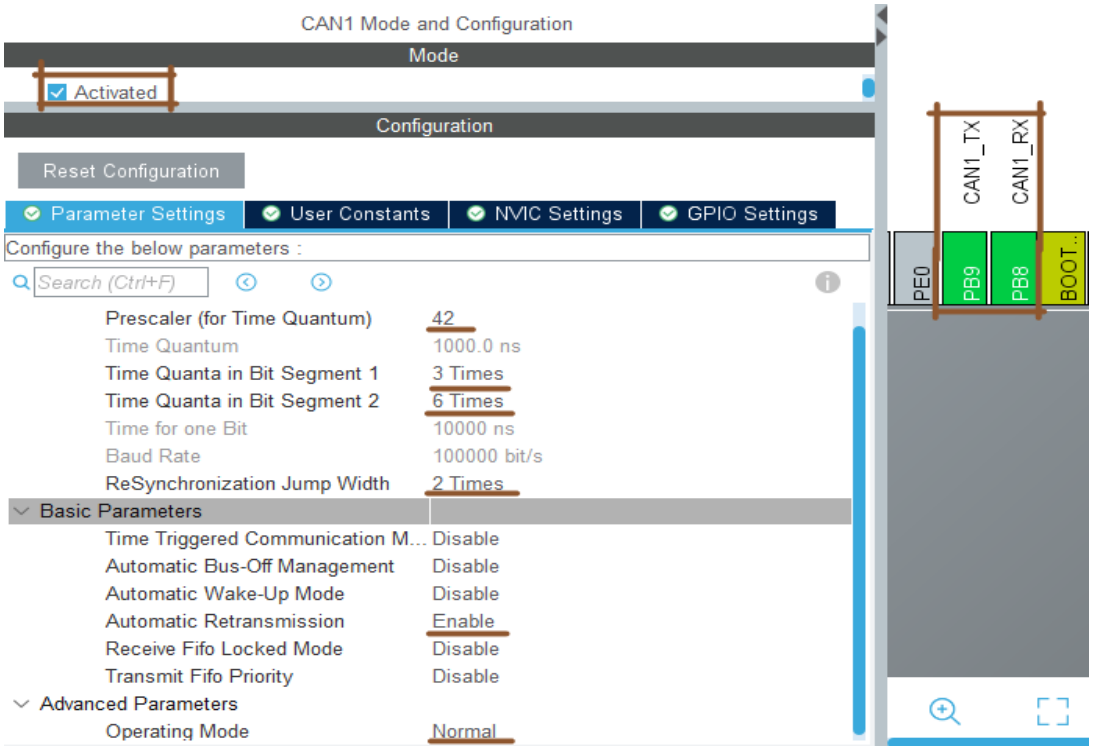
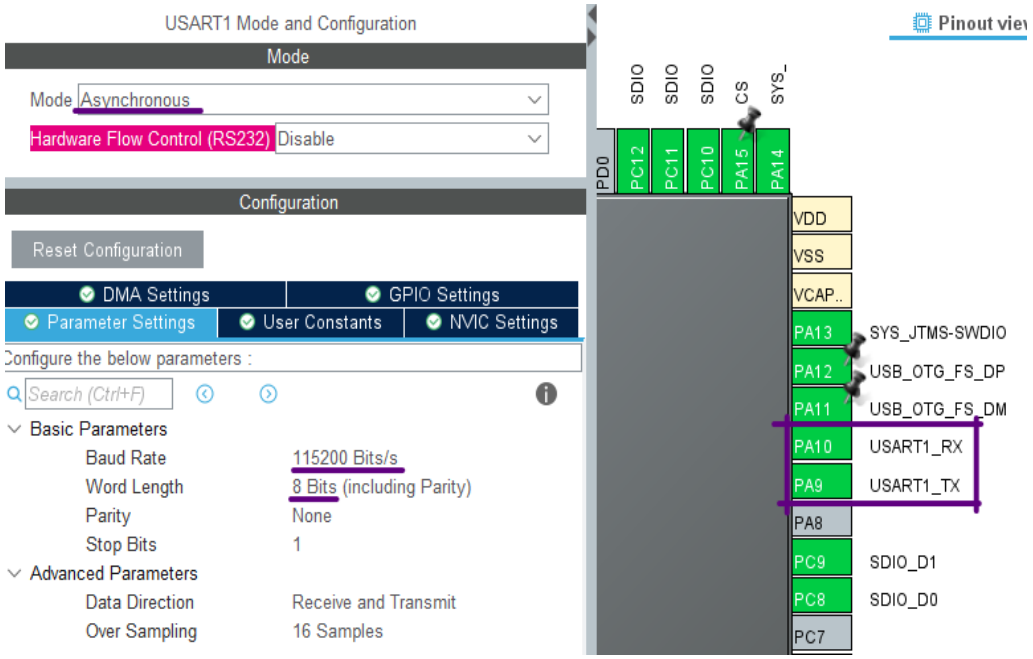


Figure III.10. Configuration de CAN1 sur cubeMX

7. Module UART

La configuration du module UART qu'on a sélectionné sur cubeMX, dont la vitesse, le mode, la taille de donnée et les pines correspondantes sont illustrés sur la figure suivante :

Figure III.11. Configuration del'UART1 sur cubeMX



8. Module USB-OTG-FS

La configuration de USB-OTG-FS est illustré dans la figure suivante :

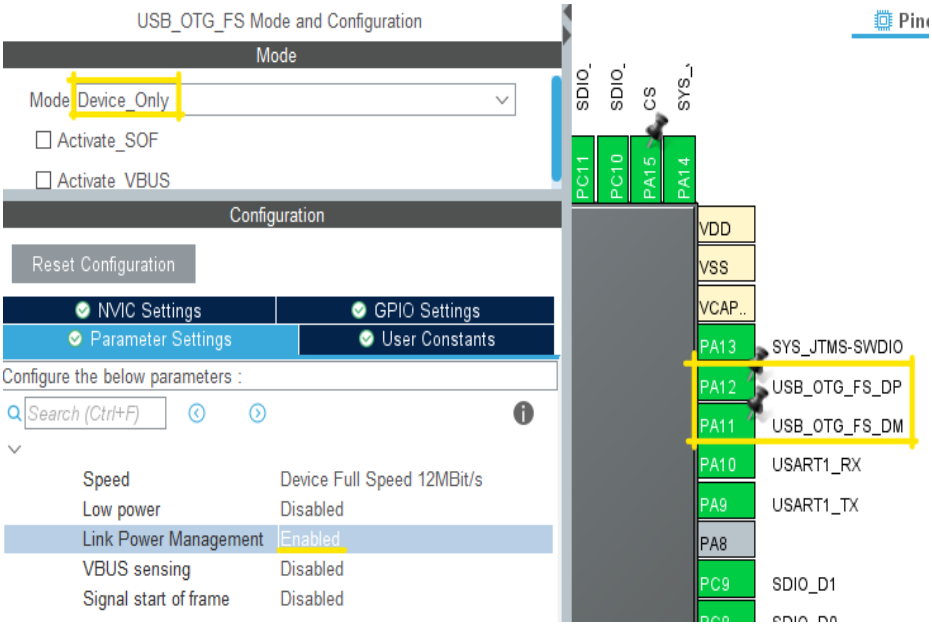


Figure III.12. Configuration de USB OTG-FS sur cubeMX

9. Module SDIO

La configuration de module SDIO, dont le mode et le facteur de division d'horloge ainsi les pines correspondantes sont illustré dans la figure suivante :

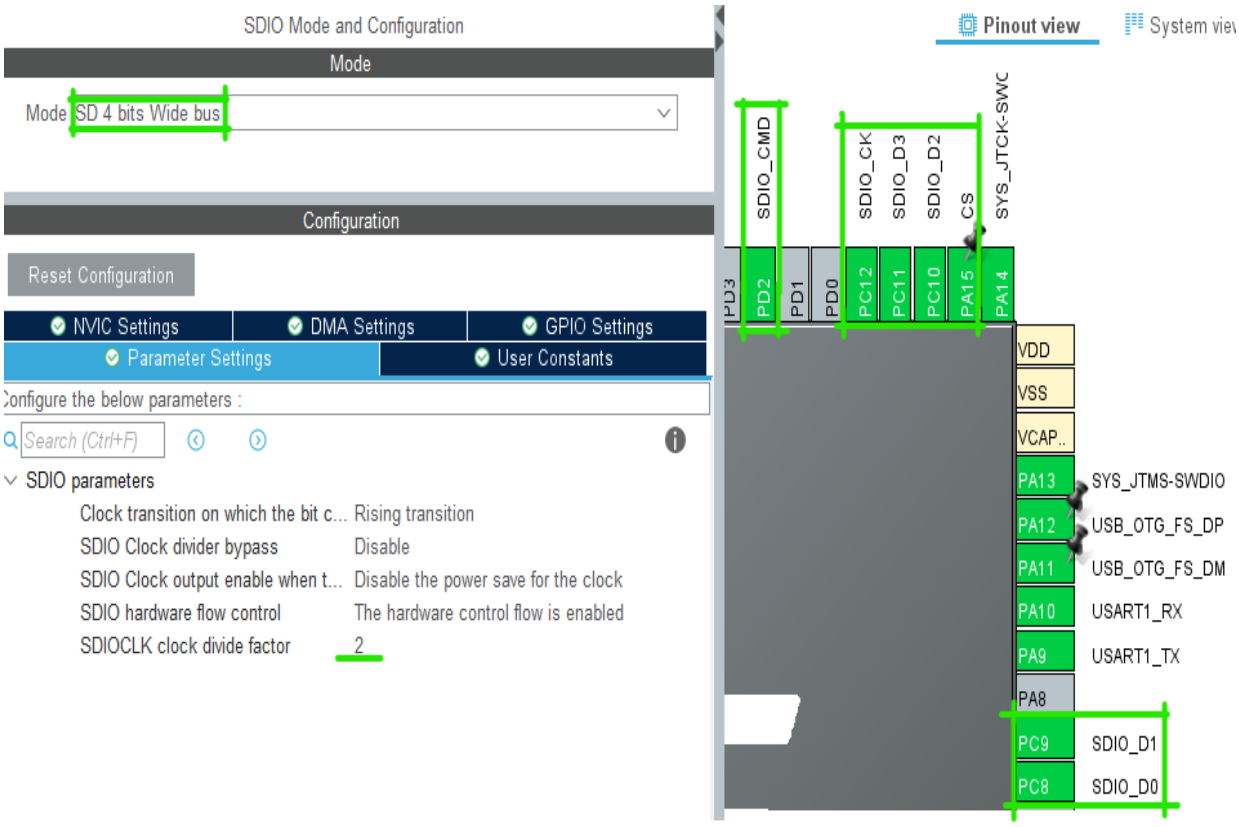


Figure III.13. Configuration de SDIO sur cubeMX

10. Module RTC

La configuration de module RTC sur cubeMX dont on l'active et on active le calendrier, ainsi la forme de l'heure et la date, comme est illustré ci-dessous :

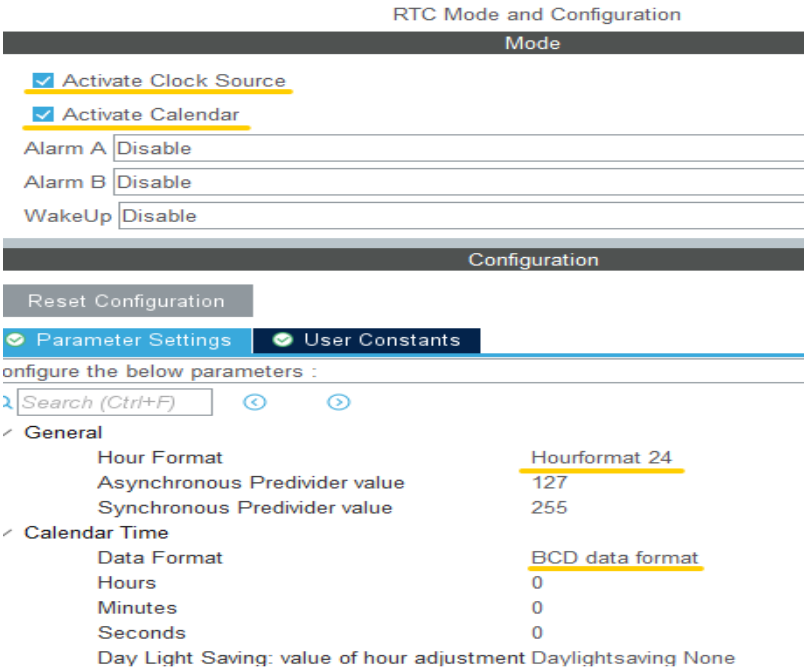


Figure III.14. Configuration de RTC sur cubeMX

11. Module TIMER

On configure le TIMER1, 2 et 6 pour l'échantillonnage de l'enregistrement des données, dont on a sélectionné le diviseur d'horloge et la période de compteur ainsi on a activé la réinitialisation automatique de TIMER, comme est illustré dans les figure ci-dessous :

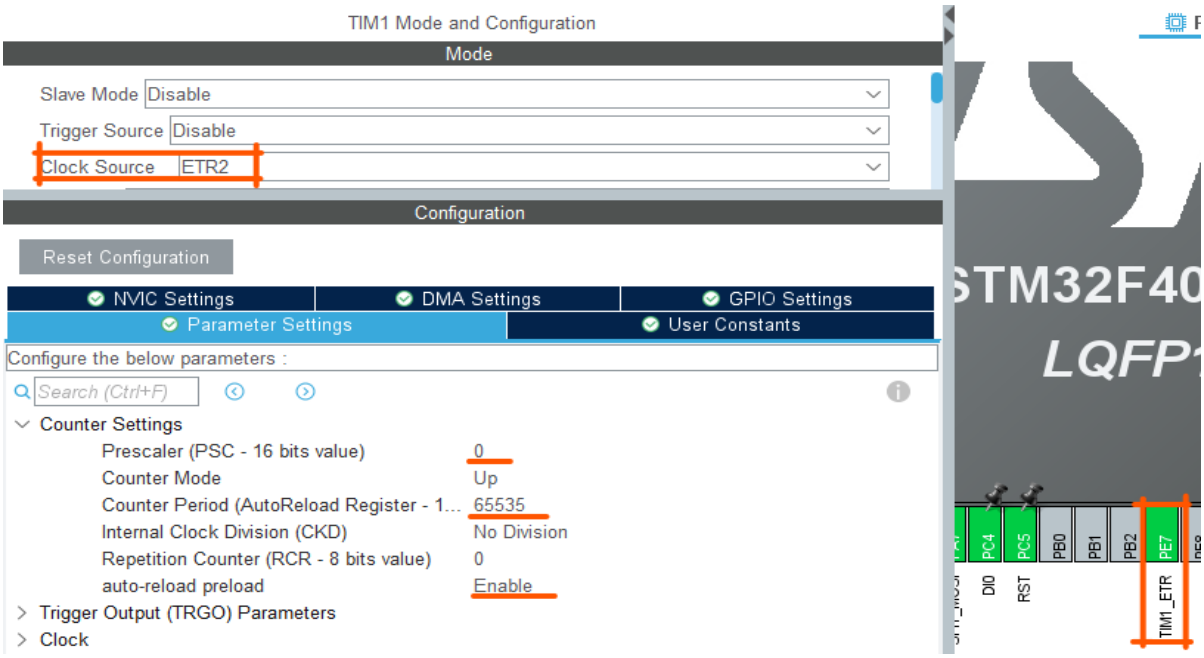


Figure III.15. Configuration du TIMER1 sur cubeMX

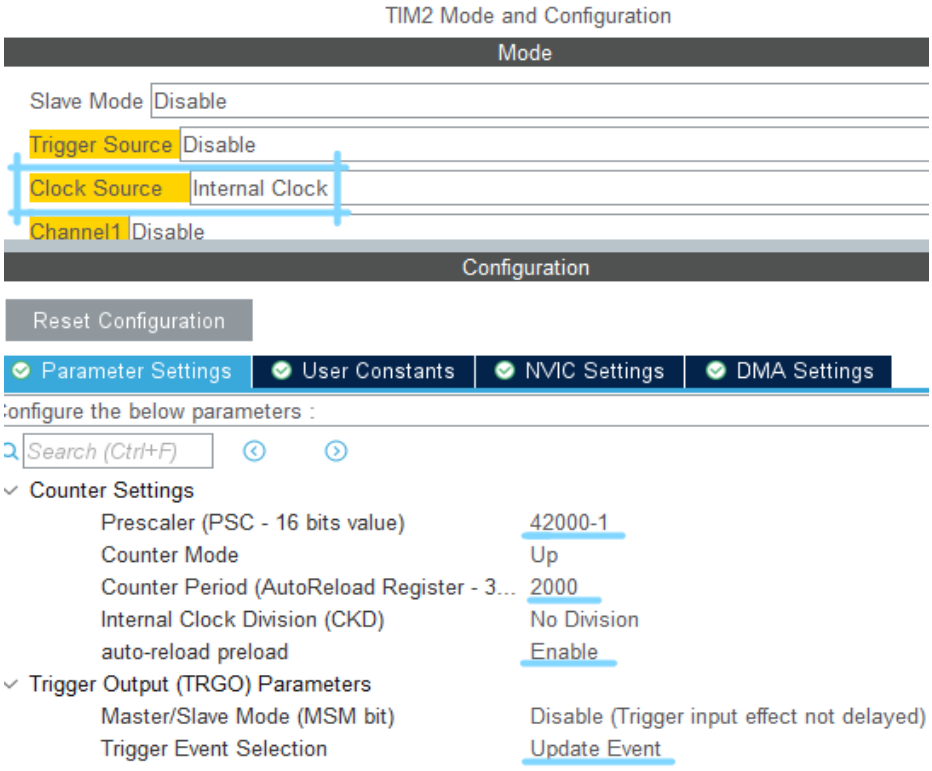


Figure III.16. Configuration du TIMER2 sur *cubeMX*

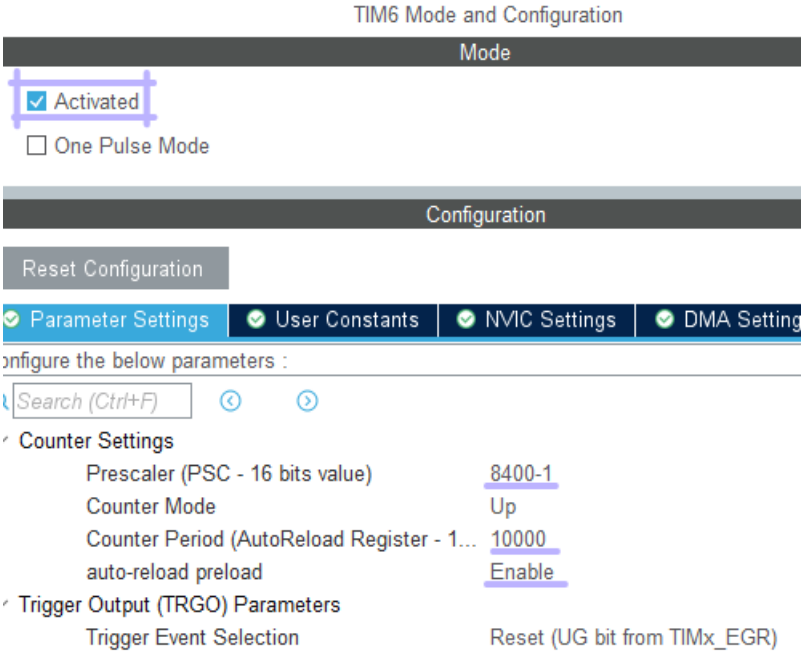


Figure III.17. Configuration du TIMER6 sur cubeMX

12. Bibliothèque FATFS

On active la bibliothèque FATFS dont on sélectionne la carte SD, comme illustré dans la figure suivante :

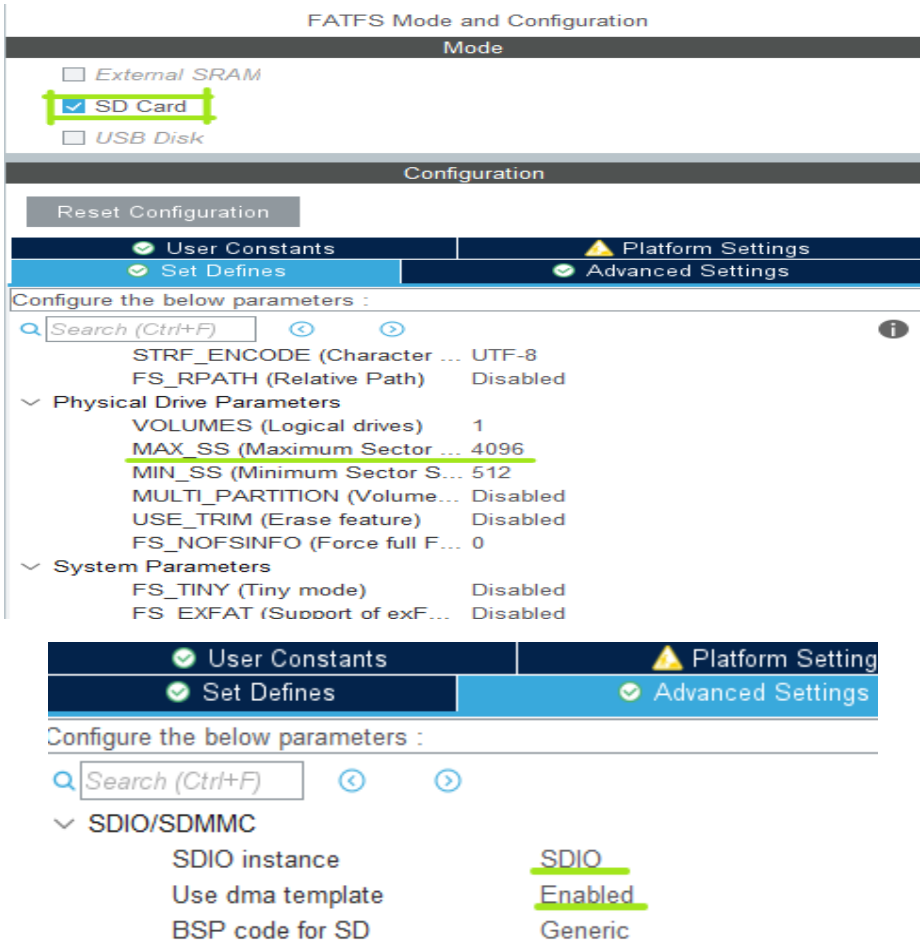


Figure III.18. Configuration de FATFS sur cubeMX

13. Bibliothèque USB_DEVICE

En activant le périphérique USB on est sensé a sélectionné sa classe et la taille de buffer utilisé pour Tx et Rx, comme suit :

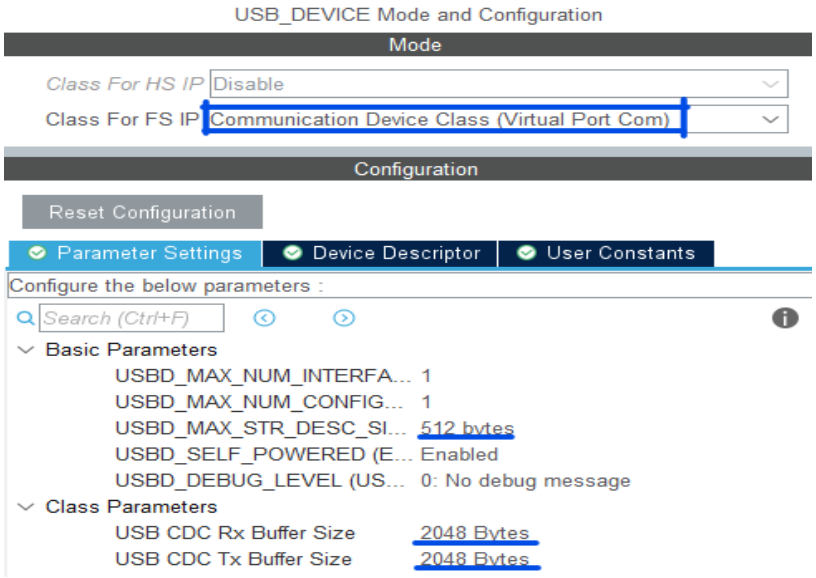


Figure III.19. Configuration de classe de l'USB sur cubeMX

Après la configuration de tous les modules, nous allons générer le code en langage C/C++ sous STM32CubeIDEn après avoir configuré les paramètres comme illustré par la figure III.26.

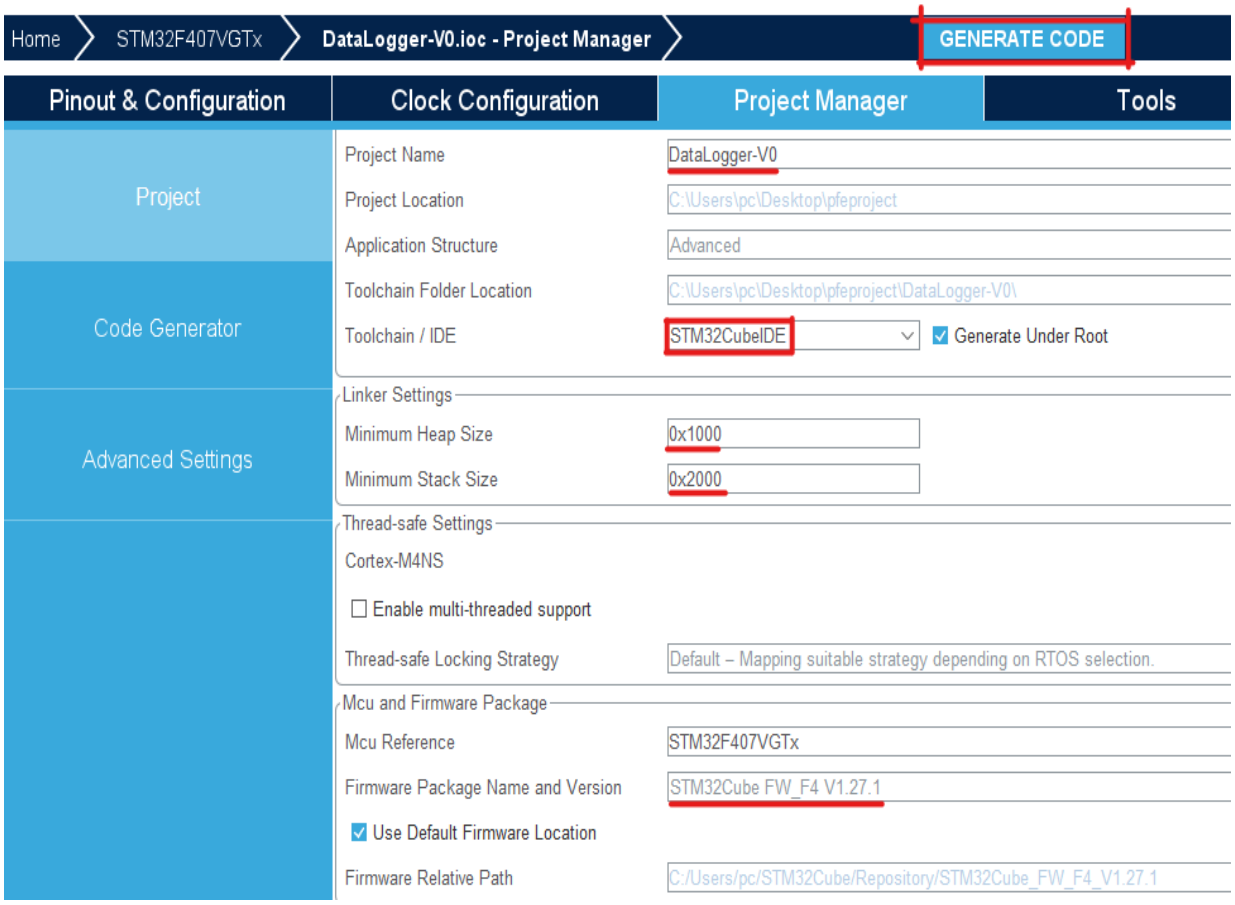


Figure III.20. Configuration de cubeMX pour génération de code sous cubeIDE

III.4.2. Partie programmation des différents modules

On ouvre le projet généré par cubeMX sur cubeIDE et on trouve les différents fichiers du système et on peut ajouter notre programme dans le fichier main.c qui se trouve dans le dossier « src » inclus dans le dossier « core » qui contient aussi « inc » contenant des fichiers headers, donc dans ces deux dossiers on peut créer des fichiers sources et headers pour notre programme, on va ajouter le programme correspond pour chaque module et tester.

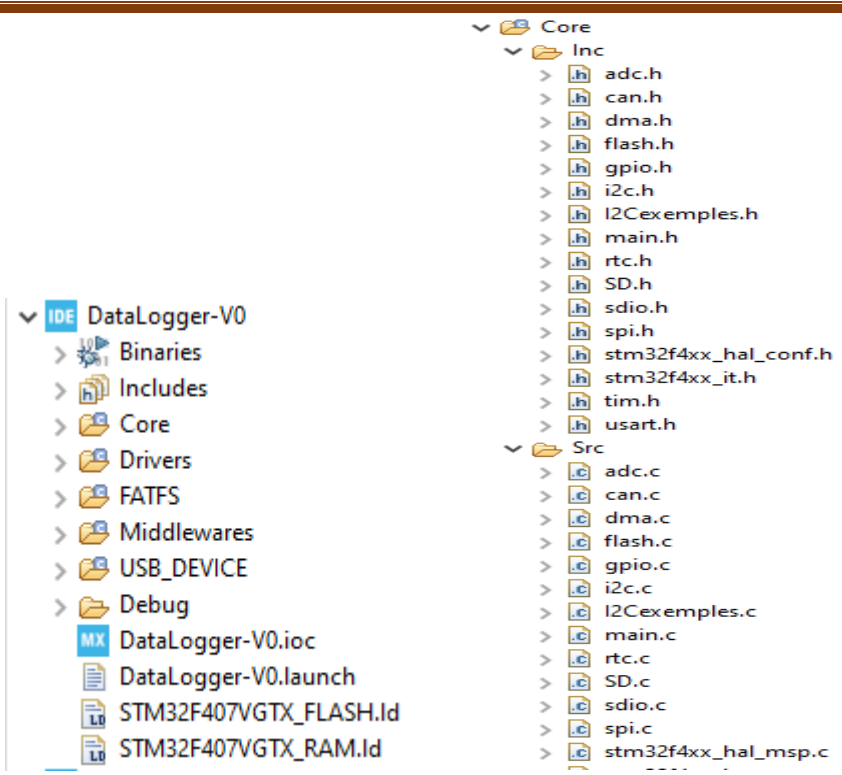


Figure III.21. Les fichiers de projets sur cubeIDE

Notre programme est constitué de trois grandes parties on les décrit ci-après.

III.4.2.1. Partie de stockage

1. Carte SD

On prépare la carte SD pour l'écriture et la lecture des données, à partir de l'interface SDIO et en utilisant les fonctions de la bibliothèque FATFS, on écrit nous propres fonctions sur un fichier source « SD.c », et on va les appeler dans le programme principal on inclut le fichier header « SD.h ». Ces fonctions sont :

✓ Mount_SD : permet l'activation de présence de la carte SD dans le système
✓ Unmount_SD : permet la désactivation de la carte SD de système
✓ Format_SD : permet le formatage de la carte SD
✓ Create_File(name) : pour la création des fichiers sur la carte SD en citant son nom
✓ Check_SD_Space : retourne l'espace de stockage libre sur la carte SD
✓ Update_File(name,data): permet l'écriture des données à un fichier en citant le nom de fichier et le buffer des données à met
✓ Read_File(name) : permet la lecture d'un fichier en donnant son nom à partir de port USB
✓ Read_Fil_LoRa : permet la lecture d'un fichier en donnant son nom à partir de module LoRa

```
/* mounts the sd card*/
void Mount_SD (const TCHAR* path);
/* unmounts the sd card*/
void Unmount_SD (const TCHAR* path);
/* Only supports removing files from home directory. Directory remover to be added soon */
FRESULT Format_SD (void);
/* creates the file, if it does not exists
 * @ name : is the path to the file*/
FRESULT Create_File (char *name);
/* checks the free space in the sd card*/
void Check_SD_Space (void);
/* updates the file. write pointer is set to the end of the file
 * @ name : is the path to the file
 */
FRESULT Update_File (char *name, char *data);
/* read data from the file
 * @ name : is the path to the file*/
FRESULT Read_File (char *name);
```

Figure III.22. Le fichier header SD.h

```
/* USER CODE END SDIO_Init 1 */
hsd.Instance = SDIO;
hsd.Init.ClockEdge = SDIO_CLOCK_EDGE_RISING;
hsd.Init.ClockBypass = SDIO_CLOCK_BYPASS_DISABLE;
hsd.Init.ClockPowerSave = SDIO_CLOCK_POWER_SAVE_DISABLE;
hsd.Init.BusWide = SDIO_BUS_WIDE_1B;
hsd.Init.HardwareFlowControl = SDIO_HARDWARE_FLOW_CONTROL_ENABLE;
hsd.Init.ClockDiv = 2;
/* USER CODE BEGIN SDIO_Init 2 */
if (HAL_SD_Init(&hsd) != HAL_OK)
   {
     Error_Handler();
   }
   if (HAL_SD_ConfigWideBusOperation(&hsd, SDIO_BUS_WIDE_4B) != HAL_OK)
   {
     Error_Handler();
   }
```

Figure III.23. Correction de fichier SDIO.c

Avant de commencer le test on fait un changement dans le fichier « SDIO.c » pour fonctionner la carte SD, car une carte SD toujours la première lecture doit être effectué on 1bit wide une fois est activé on peut utiliser le mode 4bit wide, comme illustré sur la figure III.29

Dans le fichier main.c on crée une chaine de caractère contenant le nom de fichier et une autre pour le message qu'on veut enregistrer sur le fichier, et ceci dans la case des variables le nom de fichier et un buffer contenant le message à enregistré. Le programme est comme suit :

```
//VAR SD Card
char LOGfil[]="LOGFIL.TXT";
char Buffer[30]="This is the first test";
/* USER CODE END 2 */
  Format_SD();
  Create_File(LOGfil);
  Update_File(LOGfil,Buffer);
```

On compile le programme et on debug à la carte via ST_Link, en vérifiant la carte SD en la plaçant sur le pc, et le résultat comme suit :

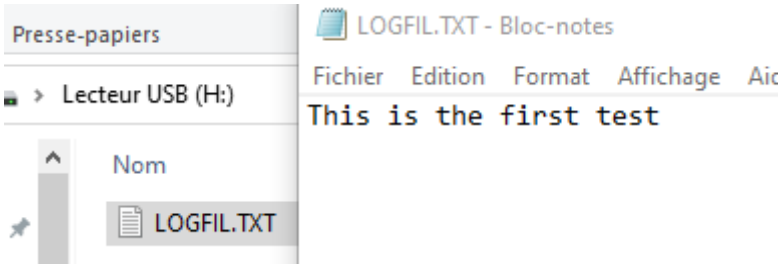


Figure III.24. Résultat de test de la carte SD

2. EEPROM

L'EEPROM est utilisé pour stocker des variables dans les cas de RST ou de perte d'alimentation. On ajoute le fichier header « flash.h » et le fichier source « flash.c » contenant les fonctions configuration et d'utilisation de l'EEPROM, qui sont :

```
//1. Suppression de secteur
void MY_FLASH_EraseSector(void);
//2.citant l'addresse de secteur
void MY_FLASH_SetSectorAddrs(uint8_t sector, uint32_t addrs);
//3. ecrire à l'EEPROM
void MY_FLASH_WriteN(uint32_t idx, void *wrBuf, uint32_t Nsize, DataTypeDef dataType);
//4. lire de l'EEPROM
void MY_FLASH_ReadN(uint32_t idx, void *rdBuf, uint32_t Nsize, DataTypeDef dataType);
```

Dans notre cas on a besoin de contrôler la partie initialisation de programme qui consiste à formater et créer le fichier d'enregistrement sur la carte SD, donc la variable « i » serve ra cette tache en changeant sa valeur entre 0 et 1. Le programme est comme suit :

```
//VAR SD Card
uint8_t i=0;
FRESULT fres;
char LOGfil[]="LOGFIL.TXT";
char BufADC1[36];
char BufADC2[20];
char BufI2C1[30];
char BufSerie[30];
char BufCAN1[30];
char BufFREQ[20];
char Bufstat[10];
/* USER CODE BEGIN 2 */
MY_FLASH_SetSectorAddrs(11, 0x080E0000);
MY_FLASH_ReadN(0,&i, 1, DATA_TYPE_8);
HAL_Delay(100);
Mount_SD("/");
//Initialization/////////////
      if(i!=1){
      Format_SD();
      Create_File(LOGfil);
      i=1;
      MY_FLASH_WriteN(0,&i, 1, DATA_TYPE_8);
      HAL_Delay(200);
       }
```

buffers pour enregistrer les données acquis par chaque entrée, sur la carte SD

Avant la boucle while

Mis dans la bucle while

Figure III.25. Programme d'initialisation de la carte SD

On a vérifié que la variable est bien stockée sur l'EEPROM en citant l'adresse de secteur dans memory map (outil fournit par cubeIDE permet de visualiser la map de la mémoire), le résultat est illustré sur la figure suivante :

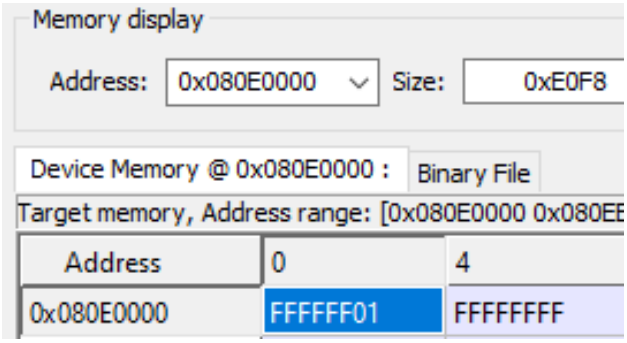


Figure III.26. Test de l'EEPROM

III.4.2.2. Partie de synchronisation et acquisition

1. Les variables

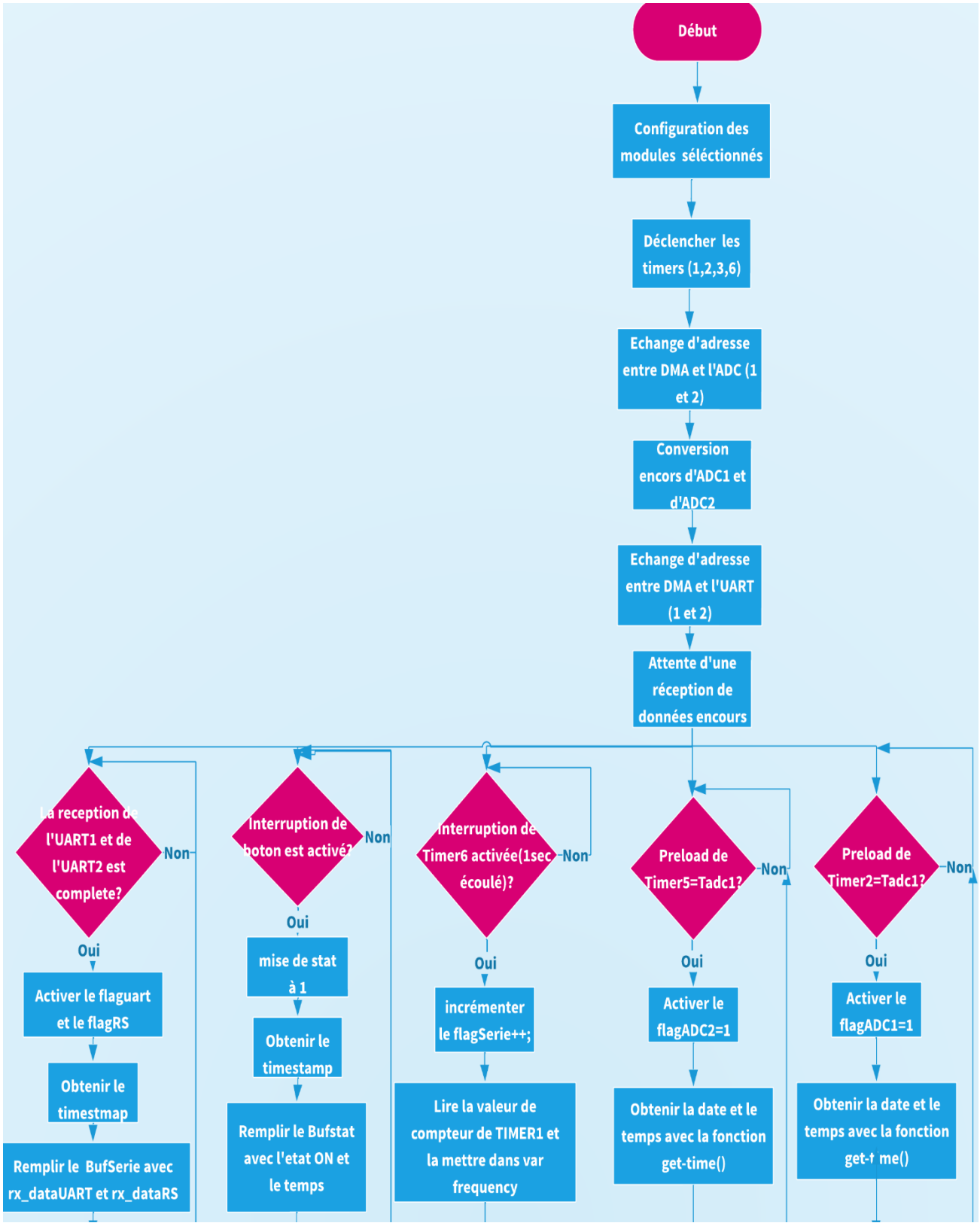
La figure ci-dessous vous montre les variables nécessaires pour l'échantillonnage et l'acquisition des données de chaque entrée.

```
//VAR de synchronization
typedef struct {
    uint32_t UnixDtTm;
} t;
t dt;
uint8_t flagADC1=0;
uint8_t flagADC2=0;
uint8_t flagSerie=0;
//VAR pour régler la date et l'heure
RTC_TimeTypeDef sTime = {0};
RTC_DateTypeDef sDate = {0};
char setDT[20]="\0";
char Dt[3]="dt";
//VAR de configuration des taux d'échantillonnage
uint8_t Tad1=2;
uint8_t Tad2=1;
uint8_t Tserie=4;
char config[5]="TT";
///VAR ADC1 4 canaux
volatile uint16_t ADC1val[4] = {0};
///VAR ADC2 1canal pour 4-20mA
volatile uint16_t ADC2val[1] = {0};
//VAR I2C1
const uint8_t TH02_address = 0x40;
uint8_t regCfg[5]={0x03,0x01,0x01,0x03,0x11};
uint16_t TM_value, HM_value;
float temp,humidity;
uint16_t timer_val;
//VAR mesure de frequence d'un signal
float frequency = 0;
float prevfreq = 0;
/////VAR pour l'obtention de l'etat de la pine Digital input////
uint8_t stat = 0;
uint8_t prevstat = 0;
/////VAR Serial port UART1 & RS485 & CAN1//////
uint8_t rx_dataUART[7];
uint8_t rx_dataRS[7];
uint8_t flaguart=0;
uint8_t flagRS=0;
```

Figure III.27. Déclaration des variables nécessaire pour l'échantillonnage et l'acquisition des données

2. Principe de fonctionnement de programme

Le fonctionnement de programme est basé sur l'organigramme décri dans la figure III.2 et l'organigramme suivant explique bien le processus de synchronisation de l'enregistrement des données, ainsi l'acquisition des données, en exploitant les variables qu'on a déclarées dans cette explication comme illustrée sur la figure suivante :



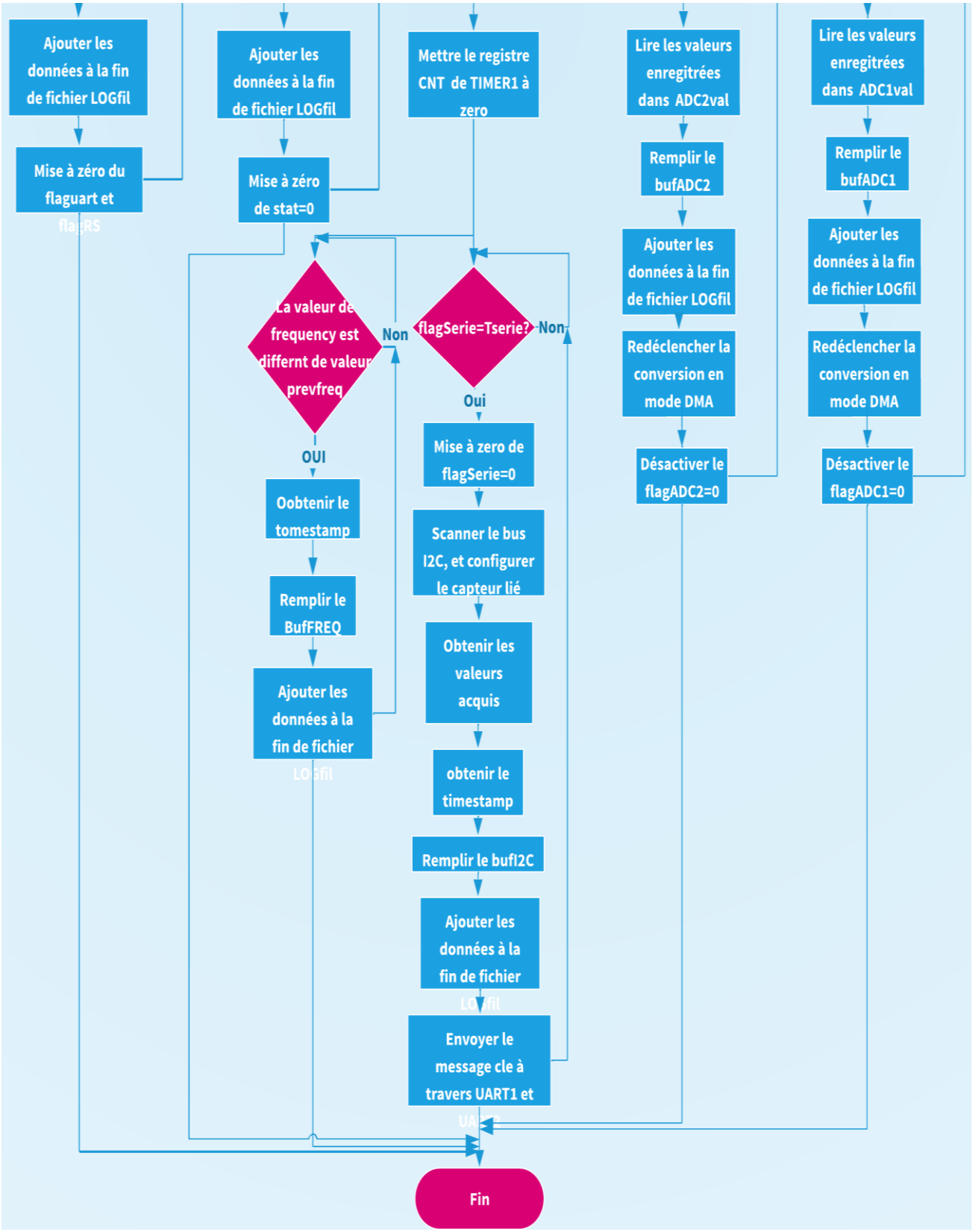


Figure III.28. Organigramme de fonctionnement de processus de synchronisation d'acquisition et d'enregistrement des données

Le code ajouté dans la boucle infinie est illustré dans la figure suivante :

```
'/Acquisition des données///////
 if (flagADC1 == 1) {
            flagADC1 = 0;
            get_time();
            ///stockage des données
            sprintf(BufADC1,"C1,%ld,%d,%d,%d,%d,\n",dt.UnixDtTm,ADC1val[0],ADC1val[1],ADC1val[2],ADC1val[3]);
            Update_File(LOGfil,BufADC1);
            HAL_ADC_Start_DMA(&hadc1,ADC1val,4);
            }
  if (flagADC2 == 1) {
            flagADC2 = 0;
            get_time();
            ///stockage des données
            sprintf(BufADC2,"C2,%ld,%d,\n",dt.UnixDtTm,ADC2val[0]);
            Update_File(LOGfil,BufADC2);
            HAL_ADC_Start_DMA(&hadc2,ADC2val,1);
        }
  if(flagSerie>=Tserie){
             flagSerie=0;
             get_time();
             TH02_TM(TH02_address,regCfg,&TM_value,&HM_value);
             temp=get_temp(TM_value);
             humidity=get_hm(HM_value);
             ///stockage des données
             sprintf(BufI2C1,"C3,%ld,%.2f,%.2f,\n",dt.UnixDtTm,temp,humidity);
             Update_File(LOGfil,BufI2C1);
             send_data(cle);
             HAL_Delay(100);
             send_dataRS(cle);
     }
  if(flaguart==1 && flagRS==1){
      flaguart=0;
      flagRS=0;
      get_time();
       ///stockage des données
      sprintf(BufSerie,"C4,%ld,%s,%s\n",dt.UnixDtTm,(char *)rx_dataUART,(char *)rx_dataRS);
      Update_File(LOGfil,BufSerie);
      memset(BufSerie,'\0',30);
      memset(rx_dataUART,'\0',7);
      memset(rx_dataRS,'\0',7);
      HAL_Delay(200);
  }
  if(frequency != prevfreq){
       get_time();
       ///stockage des données
       sprintf(BufFREQ,"C5,%ld,%.2f\n",dt.UnixDtTm,frequency);
       Update_File(LOGfil,BufFREQ);
       prevfreq=frequency;
       HAL_Delay(50);
   }
  if(stat==1){
       get_time();
       ///stockage des données
       sprintf(Bufstat,"C6,%ld,ON\n",dt.UnixDtTm);
       Update_File(LOGfil,Bufstat);
       stat=0;
       HAL_Delay(50);
   }
```

Figure III.29. Code d'acquisition et d'enregistrement des données venus des différentes entrées

III.4.2.3. Partie de transmission

Pour la transmission des données comme il est expliqué dans l'organigramme de la figure III.2 les données sont envoyées via LoRa ou bien USB selon le port de réception de mot clé « read », et si la carte SD est saturé les données seront envoyé par LoRa puisque l'USB il peut être n'est pas lié.

1. USB

On a utilisé les fonctions fournit par la bibliothèque « USB_Device » qu'on trouve dans le fichier source « usbd_cdc_if.c », pour la transmission il n'y a pas de problème on peut utiliser directement la fonction « cdc_transmit_fs », par contre dans la réception on a créé la variable « char buffer[64] » et l'ajouter dans le fichier « usbd_cdc_if.c » comme externe, puis on a changé la fonction « cdc_receive_fs » comme suit :

```
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  memset(buffer,'\0',64);
  uint8_t len = (uint8_t)*Len;
  memcpy(buffer, Buf, len);  // copy the data to the buffer
  memset(Buf,'\0',len);
  return (USBD_OK);
  /* USER CODE END 6 */
}
```

Ce changement résoudre le problème dont les caractères de l'ancien message persistent à cause de la différence dans la longueur des messages reçus.

Le code de transmission des données via USB est le suivant :

```
///Exportation des données via USB
 if(strcmp(buffer,cle) == 0){
     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
     memset(buffer,'\0',64);
     Read_File(LOGfil);
     i=0;
     HAL_Delay(300);
             }
```

2. LoRa

Pour LoRa on a utilisé la bibliothèque SX1278 qui contient deux fichiers un pour la configuration des pins de module LoRa « SX1278_hw.h» et le fichier « SX1278.h » contenant les fonctions de configuration de module en transmission et en réception.

La méthode d'utilisation de cette bibliothèque est illustrée dans la figure suivante :

```
////LoRa
SX1278_hw_t SX1278_hw;
SX1278_t SX1278;
int master;
int m=0;
int ret;
char buffer1[512];
int message;
int message_length;
int LoRa_stat=0;

if (master == 0)
       {
           ret = SX1278_LoRaRxPacket(&SX1278);
           if (ret > 0) {
               SX1278_read(&SX1278, (uint8_t*) buffer1, ret);
               printf("received : %s",buffer1);

           }
       }
///Exportation des données via LoRa
   if(strcmp(buffer1,cle)==0){
      memset(buffer1,'\0',64);
      m=1;
      SX1278_MODConfig();
      HAL_Delay(1000);
      Read_Fil_LoRa(LOGfil);
      CDC_Transmit_FS((uint8_t *)"transmited", strlen("transmited"));
      HAL_Delay(100);
      m=0;
      SX1278_MODConfig();
      HAL_Delay(100);
     i=0;
    }
void SX1278_MODConfig(void) {
    master = m;
    //initialize LoRa module
    SX1278_hw.dio0.port = DIO_GPIO_Port;
    SX1278_hw.dio0.pin = DIO_Pin;
    SX1278_hw.nss.port = NSS_GPIO_Port;
    SX1278_hw.nss.pin = NSS_Pin;
    SX1278_hw.reset.port = RST_GPIO_Port;
    SX1278_hw.reset.pin = RST_Pin;
    SX1278_hw.spi = &hspi1;
    SX1278.hw = &SX1278_hw;
    SX1278_init(&SX1278, 434000000, SX1278_POWER_17DBM, SX1278_LORA_SF_7,
    SX1278_LORA_BW_125KHZ, SX1278_LORA_CR_4_5, SX1278_LORA_CRC_EN, 10);
    if (master == 1) {
        ret = SX1278_LoRaEntryTx(&SX1278, 16, 2000);
        HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
        LoRa_stat=1;
    } else {
        ret = SX1278_LoRaEntryRx(&SX1278, 16, 2000);
        HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
        LoRa_stat=0;
    }}
```

Les variables ajoutées dans le champ des variables

Reception des données via LoRa et les données reçus seront mis dans le buffer1, mis dans la boucle while

Ajoutée dans le champ de code user4, permet l'initialisation de module LoRa (les pins, la fréquence, SF, largeur de bande, l'énergie etc), ainsi la variable master et m permettent de basculer entre le mode reception et transmission

Figure III.30. Le programme utilisé pour la transmission et la réception utilisant LoRa

Remarque 1 : Pour la fonction de transmission des données « send_LoRa » est ajoutées dans le fichier le fichier « SD.c » avec la fonction « Read_Fil_LoRa » pour pouvoir lire les données de la carte SD et les transmettre directement utilisant le buffer1.

```
void Send_LoRa (char *string){
    msg_length = strlen (string);
    ret = SX1278_LoRaEntryTx(&SX1278, msg_length, 2000);
    ret = SX1278_LoRaTxPacket(&SX1278, (uint8_t*) string,msg_length, 200);
}
```

Remarque 2 : Dans une communication entre deux modules LoRa, il fallait citer les mêmes paramètres de configuration (BW, fréquence, SF, Energie), on plus la taille de message à envoyer.

III.5. Construction de l'IHM

III.5.1. Préparation de la carte RaspberryPi

✓ **Préparation d'une image OS :** comme on a mentionné déjà on va utiliser l'outil RaspberryPi Imager, puis on place la carte SD qu'on a choisi sur le PC et la sélectionner depuis la case storage de logiciel, et on sélectionne le OS, puis on fera la configuration mentionnée sur la figure III.31 pour pouvoir faire un telnet à partir d'un autre ordinateur dans le même réseau local que le RaspberryPi pour la configurer sans avoir besoin d'un écran et d'une souris.

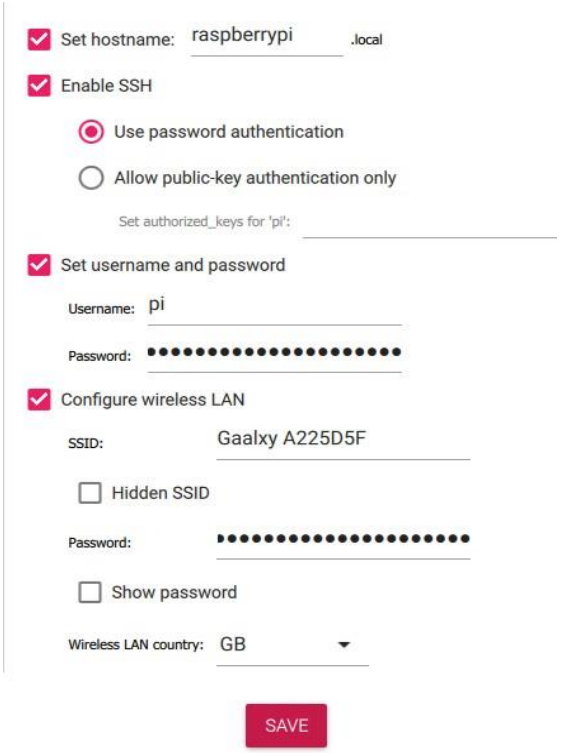


Figure III.31. Configuration de l'OS sur Raspberry Pi Imager

✓ **Configuration de la carte à par CMD Windows :** utilisant cet outil on peut directement entrer dans le terminal de système d'OS de RaspberryPi utilisant la commande « ssh pi@raspberrypi.local », et le nom d'utilisateur et mot de passe qu'on a entré lors de l'installation de l'OS, puis on pourra activer l'utilisation de l'interface VNC Viewer et accéder au bureau de Raspberry PI (le PC d'où on a effectué le telnet et d'où on ouvre VNC Viewer est connecté au même réseau wifi que celui de la carte RaspberryPi).

✓ **Accès au bureau de RaspberryPi :** à ce point on peut utiliser l'outil VNC Viewer on entrant l'adresse IP de la carte, qu'on peut l'avoir depuis la source wifi, dans ce cas on a utilisé le point d'accès mobile on peut facilement voir les appareils et leurs adresse IP, connectés à ce point d'accès comme mentionner dans la figure III.32, le bureau de mini-ordinateur est mentionné dans la figure III.33.

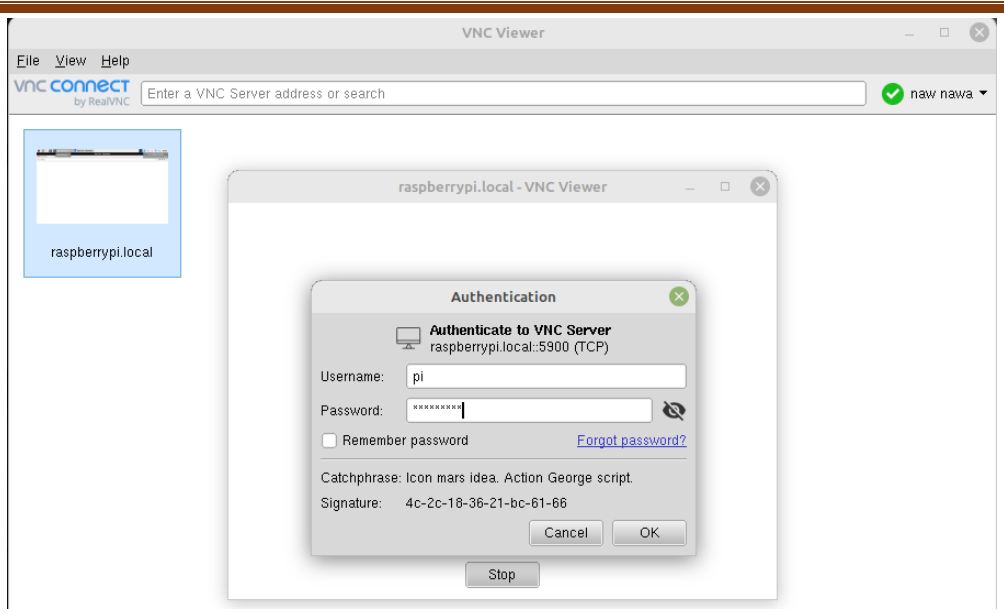


Figure III.32. VNC viewer

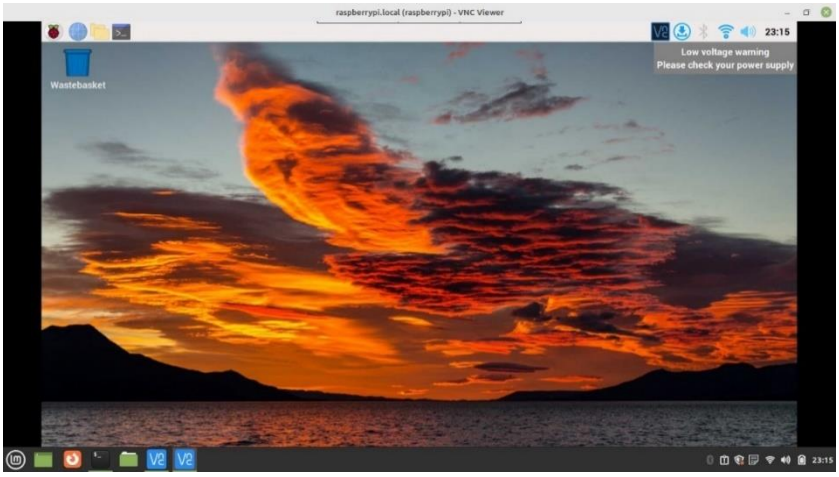


Figure III.33. Bureau de Raspberry Pi

✓ Activation du Port Série, GPIO et SPI pour la communication avec le DataLogger.

✓ **Installation de Node-red :** Dans le terminal on place les commandes suivantes :

- sudo apt_get update
- sudo apt-get upgrade
- bash<(curl-sLhttps://raw.githubusercontent.com/node-red/linux installers/master/deb/update-nodejs-and-nodered)

✓ Après on peut démarrer ce serveur en saisant sur le terminal « node-red ».

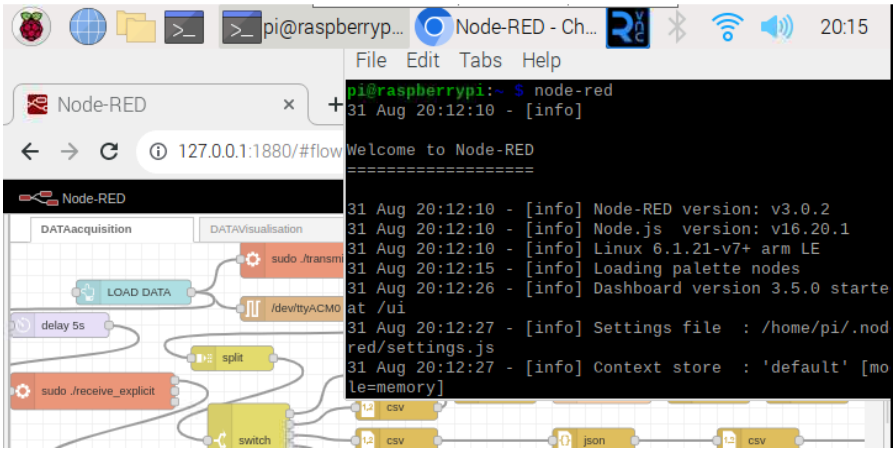


Figure III.34. Node-red sur RaspberryPi

III.5.2. Principe de fonctionnement de l'IHM

L'organigramme ci-dessus explique le principe de fonctionnement l'IHM, pour réaliser ça plusieurs nœuds sur node-red sont utilisés.

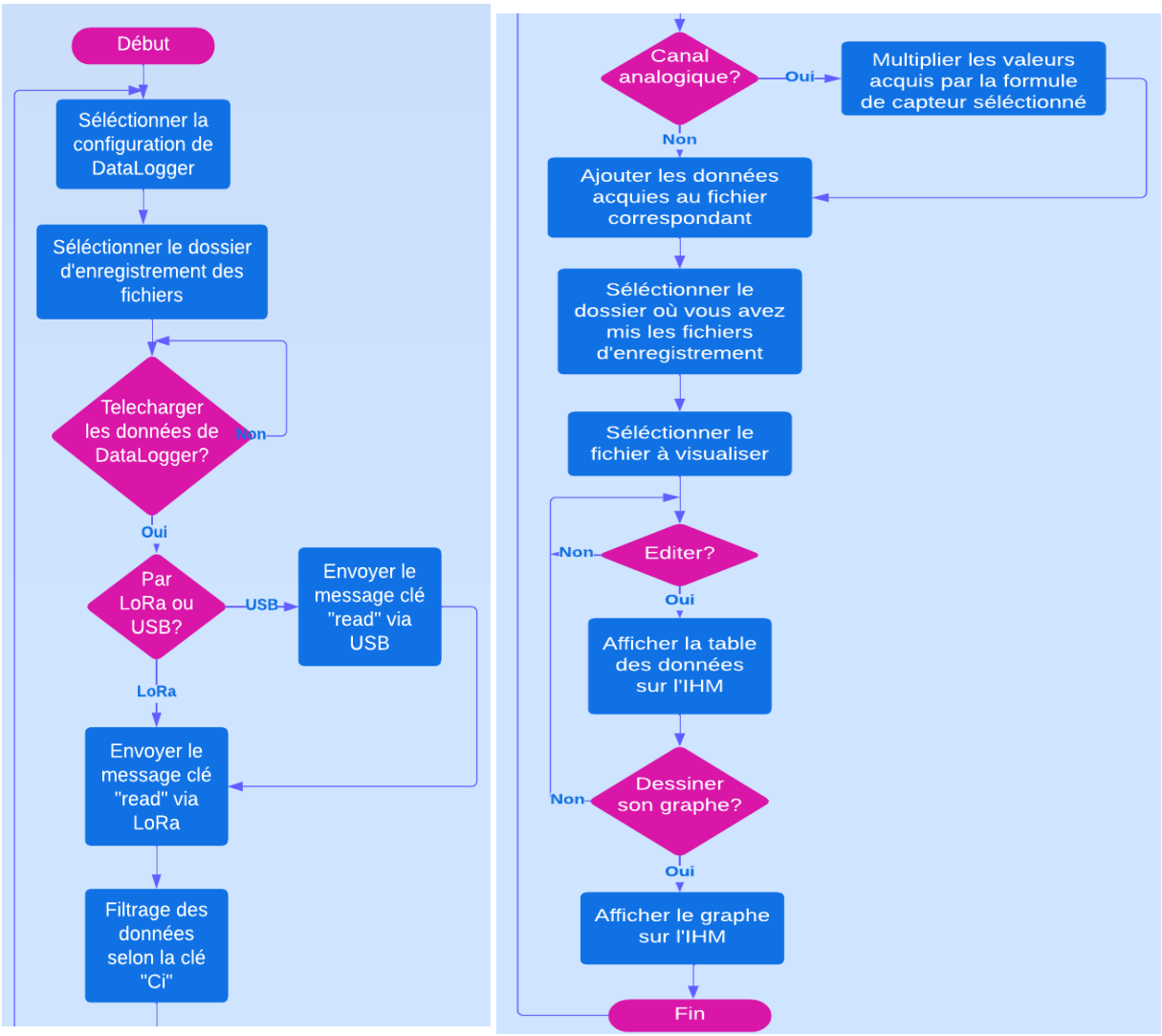


Figure III.35. Logigramme de l'IHM

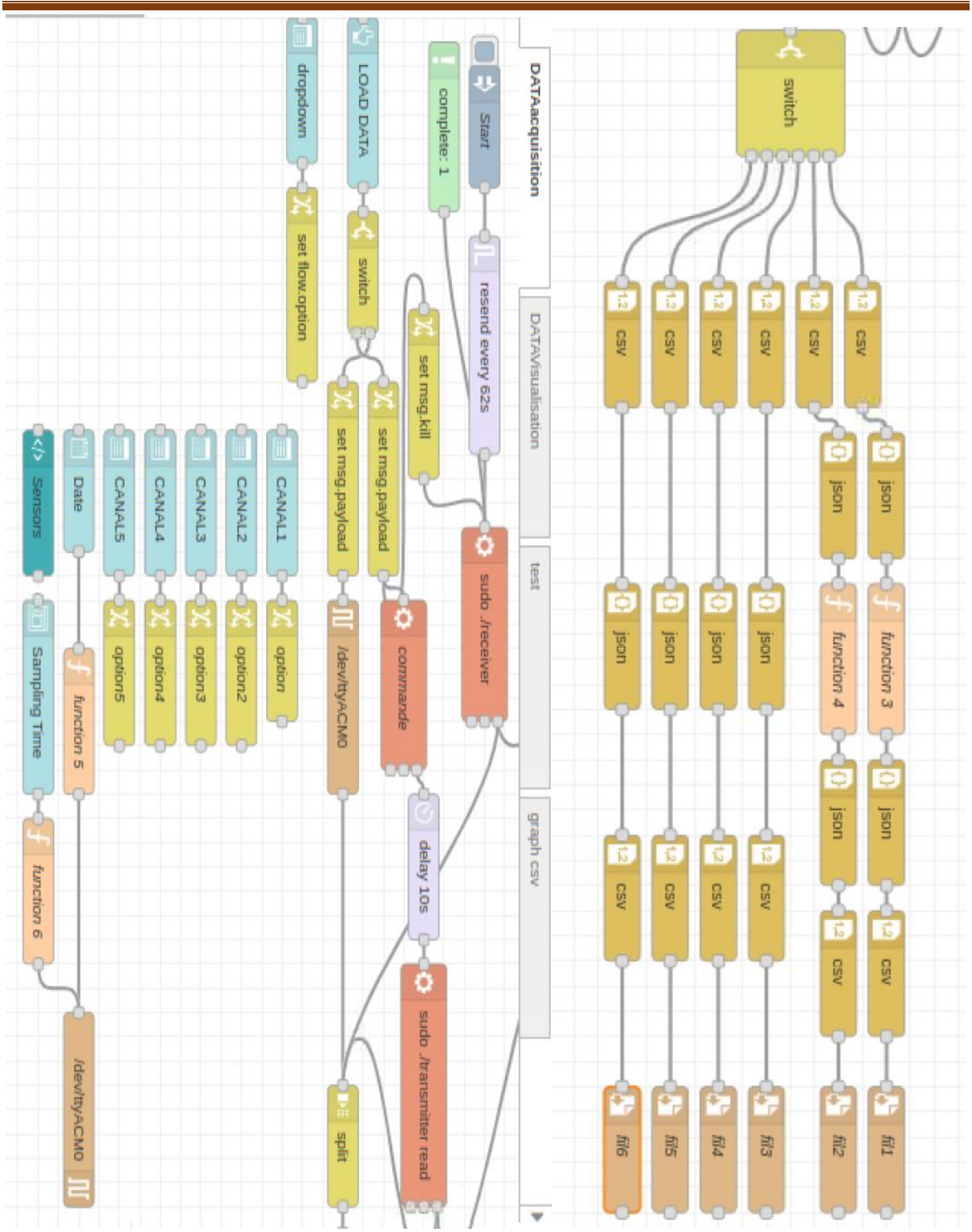


Figure III.36. Flow d'acquisition

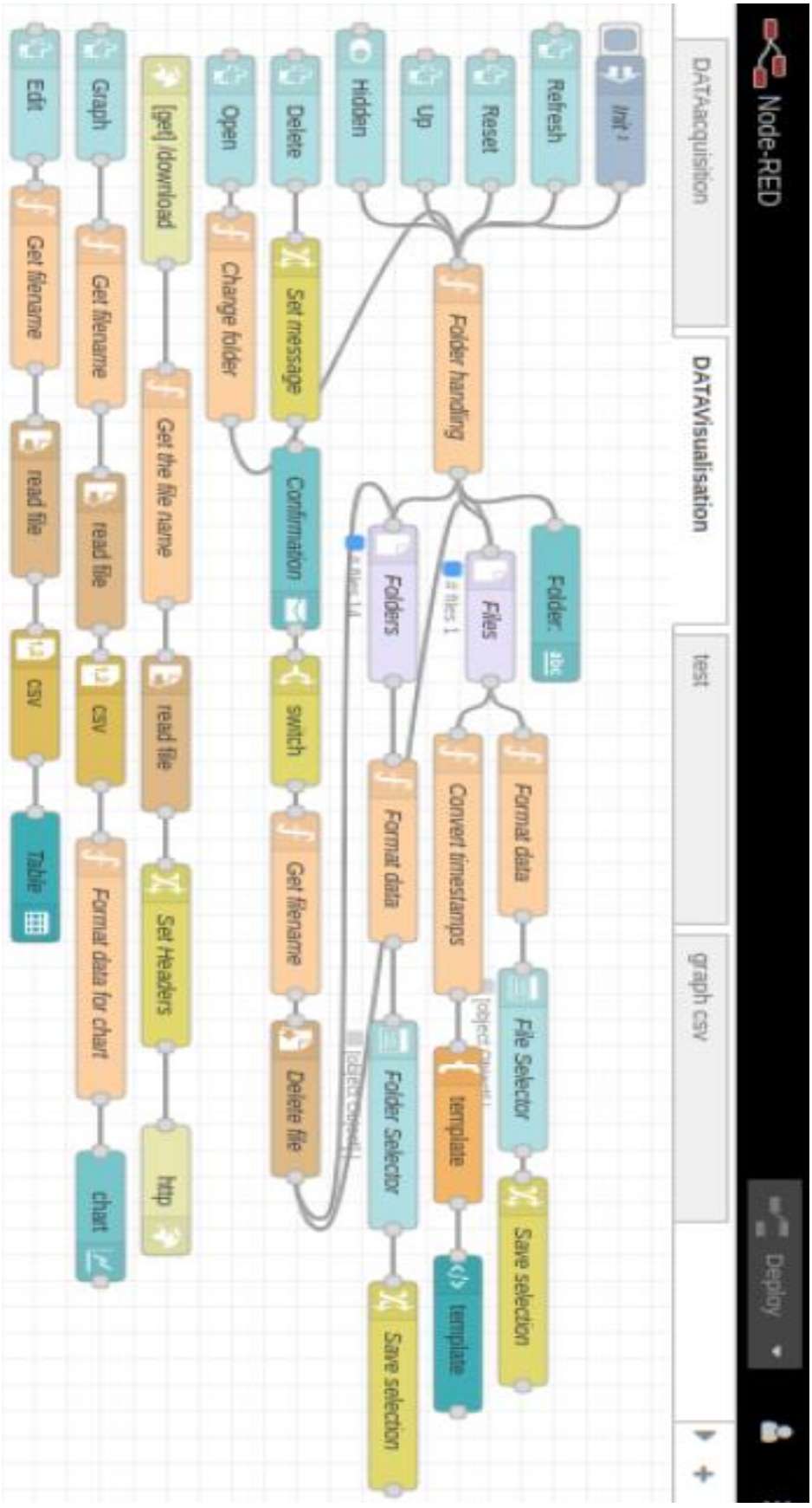


Figure III.37. Flow de visualisation des données

III.5.3.1. Différents nœuds utilisés

- ✓ Nœud Serial in et out : permettent l'ouverture de port série de RaspberryPi et communiquer avec le DataLogger via USB, le serial in pour la réception et out pour l'envoie.
- ✓ Exec node : permet l'exécution d'un programme en entrant juste la commande, on l'utilise pour exécuter le code LoRa en c en va l'expliquer dans la section suivante.
- ✓ Dashboard : fournit plusieurs nœuds permettent de créer une IHM, on utilise les boutons, le chart pour dessiner un graphe, le ui-table pour l'affichage des données sur l'IHM sous forme d'un tableau, les formulaires pour saisir la configuration suitée, les dropdown pour choisir une option, et les template permettent d'écrire un code HTML.
- ✓ Nœud de File et Folder : pour accéder à un dossier depuis l'interface et afficher les fichiers dedans, et effectuer des opérations affichage, suppression et téléchargement.
- ✓ Nœud change : permet de donner à un message entrant une nouvelle forme, json ou bien booléen, ou même le donner un nom d'une variable globale pour l'utiliser dans d'autres nœuds.
- ✓ Nœud switch : permet
- ✓ Nœud split : permet de deviser un message string en une séquence du message selon le diviseur choisi, peut-être une virgule ou par un ligne « \n », etc.
- ✓ Nœud json : permet la conversion d'un message en un objet json ou le contraire.
- ✓ Nœud CSV : permet de convertir un fichier texte en un fichier CSV.
- ✓ Nœud Write File et Read File : permettent l'écriture d'un fichier et sa lecture en entrant le chemin vers ce fichier.
- ✓ Nœud fonction : permet d'écrire un code en java script.
- ✓ Nœud de http in et out : permet de faire des requêtes http, dans ce cas pour télécharger un fichier sélectionné lors de l'ouverture de l'accès à l'interface par d'autres appareils
- ✓ Nœud remote access : permet l'accès à distance à l'interface par un appareil mobile.

III.5.2. Programmation de module LoRa

Pour le module LoRa on n'a pas un nœud sur node-red pour l'utiliser directement, et on a utilisé le nœud exec qui permet l'exécution d'un code en C ou python sur node-red, on a apte à utiliser le langage C pour écrire un programme de réception et de transmission.

On a utilisé une bibliothèque LoRa.h qu'on a trouvé sur github [52], on inclut ce fichier dans les deux codes, on va utiliser la même configuration que celle de module LoRa placé avec le DataLogger, comme suit :

- ✓ SF = 7 => 7 bits dans un symbol
- ✓ Fréquence = 434 MHz
- ✓ BW =125 KHz
- ✓ Le mode utilisé est explicite
- ✓ On active le contrôle d'erreur CRC, CRC=5

Le code est comme suit :

```
LoRa_ctl modem;
modem.spiCS = 0;//Raspberry SPI CE pin number
modem.tx.callback = tx_f;
modem.rx.callback = rx_f;
modem.eth.preambleLen=12;
modem.eth.bw = BW125;//Bandwidth 125KHz
modem.eth.sf = SF7;//Spreading Factor 7
modem.eth.ecr = CR5;//Error coding rate CR5
modem.eth.CRC = 1;//Turn on CRC checking
modem.eth.freq = 434000000;// 434MHz
modem.eth.resetGpioN = 4;//GPIO4 on lora RESET pin
modem.eth.dio0GpioN = 17;//GPIO17 on lora DIO0 pin to c
modem.eth.outPower = OP17;//Output power
modem.eth.powerOutPin = PA_BOOST;//Power Amplifire pin
modem.eth.AGC = 1;//Auto Gain Control
modem.eth.OCP = 240;// 45 to 240 mA. 0 to turn off protec
modem.eth.implicitHeader = 0;//Explicit header mode
modem.eth.syncWord = 0x39;
```

1. Code de réception :

On ajoute la fonction rx_f avant le main, qui retient un message reçu à la console dans le cas d'une réception d'un message, comme suit :

```
#include "LoRa.h"

void tx_f(txData *tx){
    printf("\n");
}
```

Et le code qui permet le démarrage de la réception est le suivant :

```
LoRa_begin(&modem);
LoRa_receive(&modem);
sleep(60);
```

Ce programme est redéclenché chaque deux secondes à l'aide de nœud trigger sur node-red.

2. C ode de transmission et réception :

On ajoute la fonction tx_f pour la transmission de message clé et rx_f comme suit :

```
#include "LoRa.h"

void tx_f(txData *tx){
    printf("\n");
}
void * rx_f(void *p){
    rxData *rx = (rxData *)p;
    fprintf(stdout,"%s\n", rx->buf);//Data we'v received
    free(p);
    return NULL;
}
```

Le code suivant permet de transmettre d'abord le message clé qu'on met en entrée puis on démarre la réception pendant 20 secondes.

```
LoRa_begin(&modem);
modem.tx.data.buf = txbuf;
modem.tx.data.size = sprintf(modem.tx.data.buf,"%s",argv[1])
LoRa_send(&modem);
sleep(((int)modem.tx.data.Tpkt/100000)+1);
LoRa_begin(&modem);
LoRa_receive(&modem);
sleep(20);
LoRa_end(&modem);
```

Pour la compilation de ces codes on utilise la commande gcc sur le terminal comme suit :

- gcc -o receiver receiver.c LoRa.c -lpigpio -lrt -pthread -lm
- gcc -o transmitter transmitter.c LoRa.c -lpigpio -lrt -pthread -lm

Après la compilation on peut utiliser le fichier exécutable de chaque code sur node-red utilisant la commande « sudo ./nom-de-l'executable » dans le nœud exec.

III.5.3. Résultat de l'IHM

L'interface qu'on a construit avec node-red est illustré sur la figure suivante ainsi on joins une explication de ses différentes composantes :

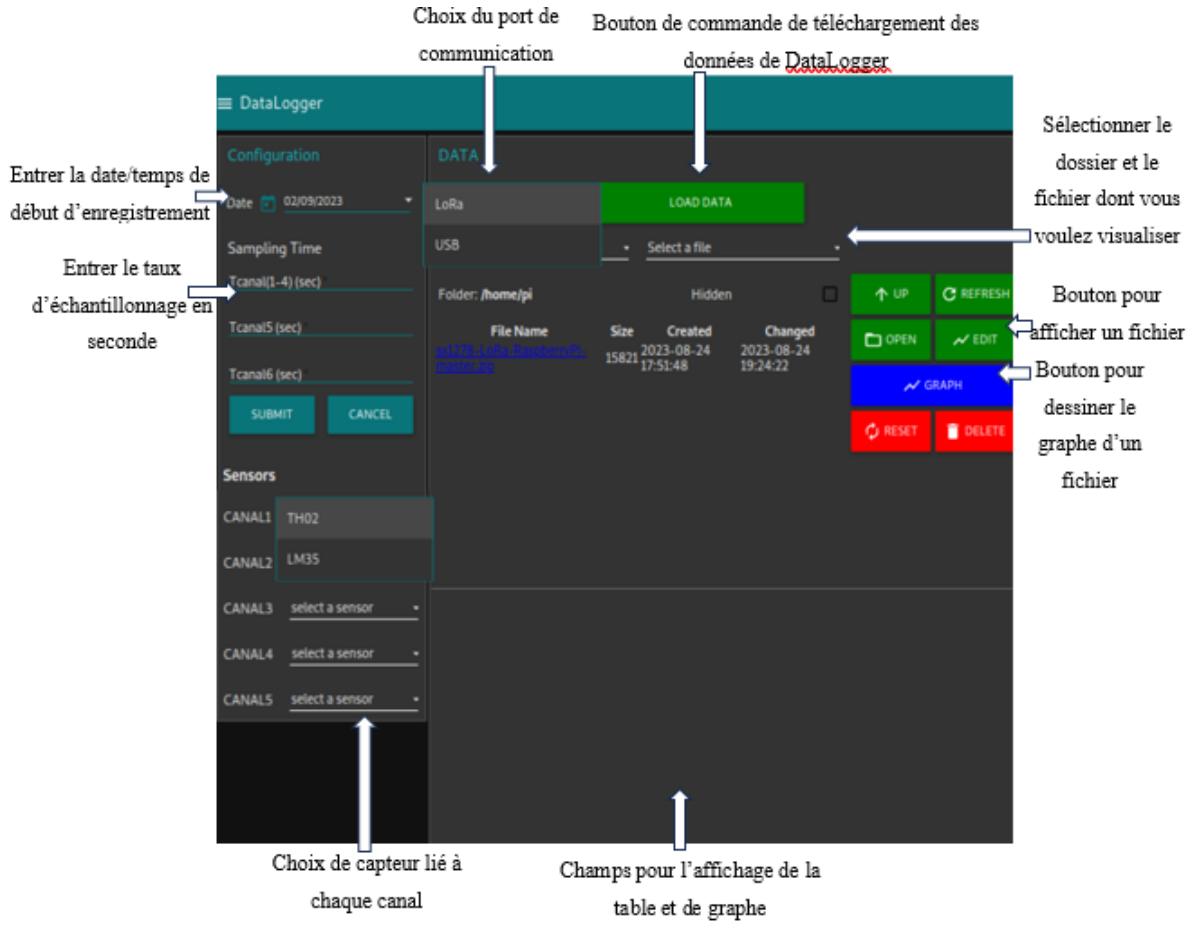


Figure III.38. Les composantes de l'IHM

III.6. Réalisation de PCB et du boitier du DataLogger

III.6.1. PCB

On a réalisé un circuit imprimé pour faciliter le câblage des composants avec la carte et éviter les problèmes causés par les fils. Pour cela on a choisi la méthode chimique dont on vous explique les étapes de travail ci-dessous :

1. **Dessin du circuit :** on a utilisé fritzing [53] qui est un logiciel de dessin des circuits électroniques, le schéma de câblage est illustré dans la figure III.38, on a relié les différents modules de conditionnement des signaux avec la carte, et des terminals headers pour relier les capteurs.

2. Dessin de PCB : après le câblage on peut passer au dessin de PCB dans la fenêtre de PCB, on a utilisé une couche avec deux faces une supérieure et une inférieure, le dessin est illustré dans la figure III.39.

3. Réalisation de PCB : on a utilisé la procédure chimique pour la réalisation de PCB, premièrement on a imprimé le schéma des deux faces sur un papier glaçant, ensuite on a fait le transfert de schéma imprimé sur la plaque a deux faces de cuivre utilisant le dissolvant, puis on passe à la gravure en mettant la plaque dans une solution alcaline pour retirer le cuivre indésirable, enfin on nettoie les pistes utilisant le dissolvant. Après on peut passer au perçage des trous pour soudes les composants, le résultat est illustré sur la figure III.40.

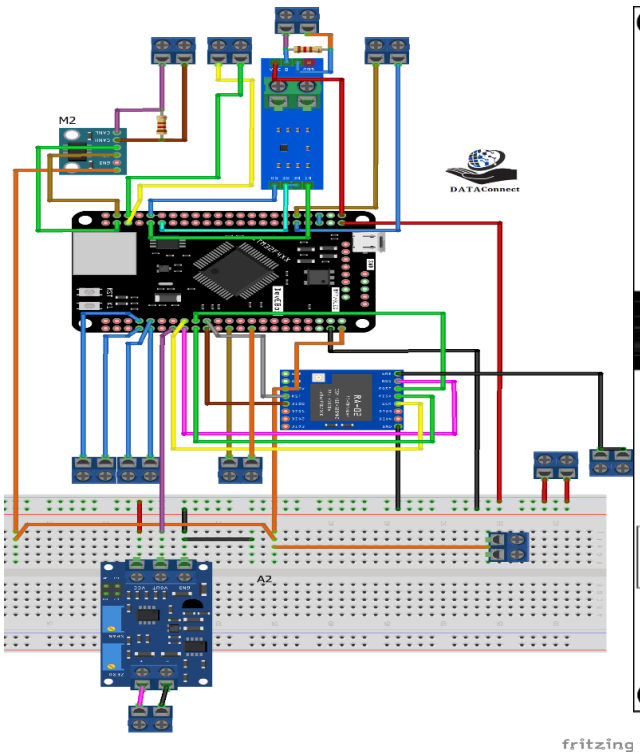


Figure III.39. Schéma de câblage

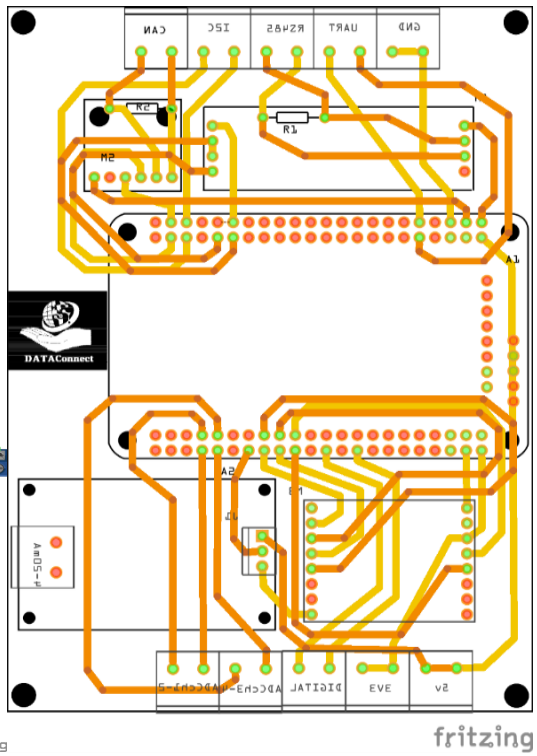


Figure III.40. Illustration de PCB

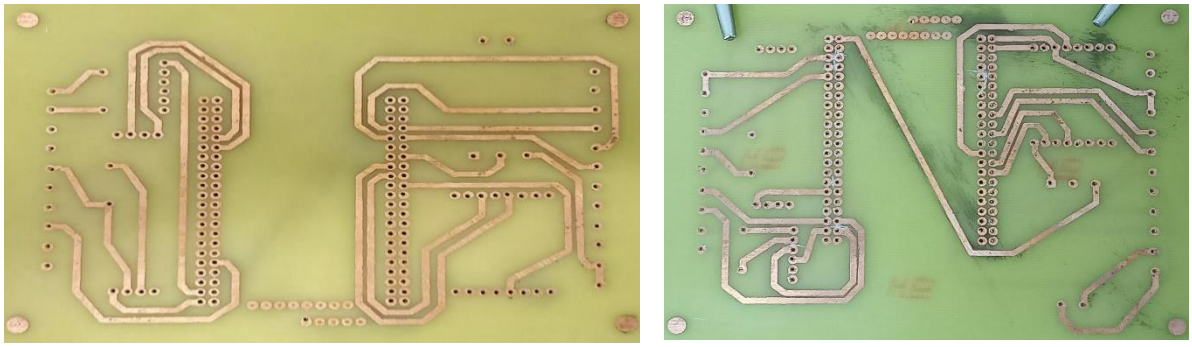


Figure III.41. Résultat de la conception de PCB face supérieur et inférieur

III.6.2. Conception et réalisation d'un boitier pour le DataLogger

On a utilisé le logiciel online ThinkerCad [54], qui permet la conception 3D et d'autres, on a pris les mesures nécessaires et on a fait la conception 3D de boitier, le résultat de conception est illustré dans la figure III.41, et pour l'impression on a généré un fichier STL, et l'envoyer à l'entreprise ALKADWORKS [5] qui a occupé de la réalisation. Les résultats sont illustrés sur la figure III.42 et la figure III.43.

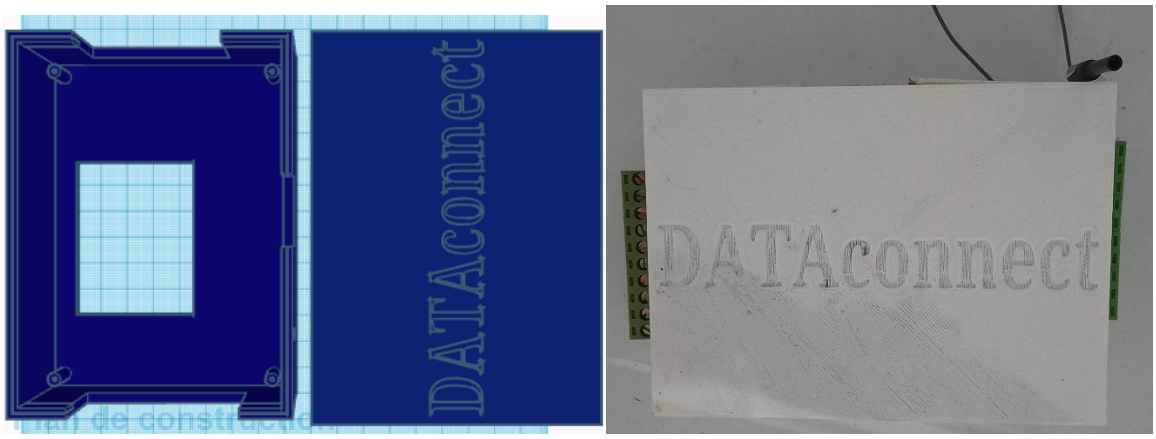


Figure III.42. Résultat de boitier

III.7. Test et résultat final

III.7.1. Capteurs utilisés pour le test

Depuis la liste des capteurs décrite dans le chapitre II, on a sélectionné les capteurs suivants pour le test (ceux qui sont disponibles) :

- ✓ LM35 branché au canal1 dont la formule de calcul de la température d'après la valeur convertie est la suivante :

$$Temp = \frac{ADC_{val} * V_{ref}}{(2^{12} - 1) * \mathrm{S}}$$

$$Temp = \frac{ADC_{val} * 3.3V}{4095 * 0.01V}$$

$$Temp = ADC_{val} * 0.08$$

$V_{ref} = 3.3V$: La tension de référence.

$\mathrm{S} = 10mV/°C$: La sensibilité de capteur.

- ✓ YL-69 branché au canal2, ce capteur permet de mesure l'humidité de sol utilisé dans l'agriculteur surtout, pour la formule de calcul on va juste le multiplier par le quantum pour avoir la valeur mesurée par ce capteur :

$$q = \frac{V_{ref}}{résolution\ d'ADC}$$

$$q = \frac{3.3V}{2^{12}} = 0.8e^{-4}$$

- ✓ Pour les autres canaux 3 et 4 on peut les liés au GND et au 3.3V pour le test.
- ✓ TH02 au canal6 port d'I2C, on a ajouté sa configuration nécessaire dans la bibliothèque I2Cexemple qu'on a créé (voir l'annexe pour plus de détail), la formule de calcul de ce capteur qui mesure la température et l'humidité d'après sa datasheet est :

Formule de calcul d'humidité : Formule de calcul de température :

$$\text{humidity} = (\text{HM_value}/16) - 24) * 0.05 \qquad temp = (TM_value/32) - 50$$

- ✓ YF_S201 capteur de débit d'eau 0.3-6L/min, 1/4″ 1.75MPa, connecté au canal7 qui permet de compter le nombre d'impulsion par seconde (fréquence en Hz). On a expliqué déjà le principe de fonctionnement de ce capteur qui génère un signal de sortie sous forme d'impulsions, en comptant le nombre d'impulsions générées, nous pouvons calculer le débit d'eau qui le traverse, on propose de calculer le débit d'eau en litres par heure et ceci utilisant la formule suivante :

$$Débit\left(\frac{L}{h}\right) = fréquence * \frac{60}{7.5} = fréqunece * 8$$

- ✓ Pour digital input (canal8) on a utilisé un buzzer, lorsqu'il est activé le DataLogger va enregistrer cet état et son temps.
- ✓ Pour les autres canaux séries (UART, RS485 et CAN) on n'a pas trouvé des capteurs disponibles dans les magasins d'électronique d'Alger, on a utilisé une autre carte STM32Nucleo64 pour le test (vous pouvez consulter l'annexe pour voir le code utilisé).

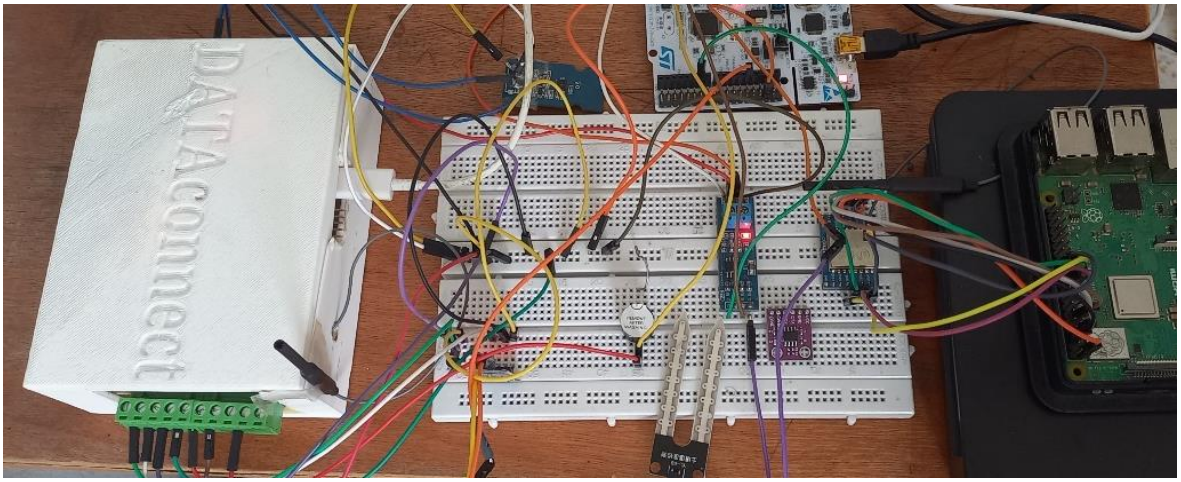


Figure III.43. Illustration de câblage

Remarque : on peut estimer le cout de ce premier prototype entre 15000DA et 20000DA sans compter la carte RaspberryPi parce que l'utilisateur peut la remplacer par un PC avec un module LoRa-USB. Ce module LoRa-USB on peut le construire à partir d'une autre carte STM32 avec un module LoRa puis les données reçues à partir LoRa seront transmettre au PC à travers le port USB.

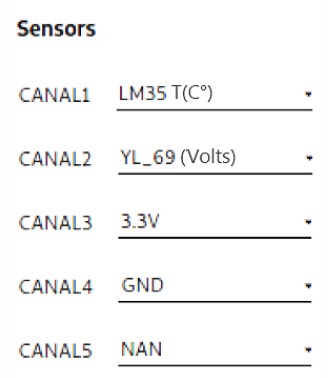
III.7.2. Visualisation des résultats sur l'IHM

Dans ce qui suit on affiche les résultats obtenus par chaque canal tel que la table des données et le graphe correspondant sur l'IHM.

En cliquant sur le bouton load sur l'IHM on obtient les données enregistrées sur le DataLogger, et les données seront enregistrée dans des fichiers dont chaque type d'entrée dans un fichier seul, on obtient fil1.csv, fil2.csv ainsi de suite selon le nombre de canaux.

❖ **Canaux analogiques :**

On sélectionne en premier le capteur lié à chaque canal comme suit :



La table suivante représente les données acquises par les canaux analogiques dont la première colonne représente les valeurs de température mesurée par le capteur de températures LM35, la deuxième colonne représente les valeurs mesurées par le capteur d'humidité de sol YL-69, dont le sol est sec, et pour la dernière colonne on a relié ce canal au GND donc les valeurs acquis sont des zéros, ces résultats montre le bon fonctionnement de notre DataLogger.

Timestamp	canal1	canal2	canal4
946684802	37.44	3.276	0
946684803	39.2	3.276	0
946684804	37.28	3.276	0
946684805	38.24	3.276	0
946684806	37.28	3.276	0
946684806	38.08	3.276	0
946684807	36.96	3.276	0
946684808	38.16	3.276	0
946684809	37.04	3.276	0
946684810	37.44	3.276	0
946684811	37.52	3.276	0
946684812	40.32	3.276	0
946684812	38.56	3.276	0
946684813	35.84	3.276	0
946684814	36.72	3.276	0
946684815	35.76	3.276	0

Figure III.44. Affichage des valeurs acquis par les canaux analogiques sur l'IHM

En cliquant sur le bouton Graph on obtient le graphe suivant qui représente l'évolution dans le temps (Timestamp) des valeurs représentées sur la table ci-dessus.

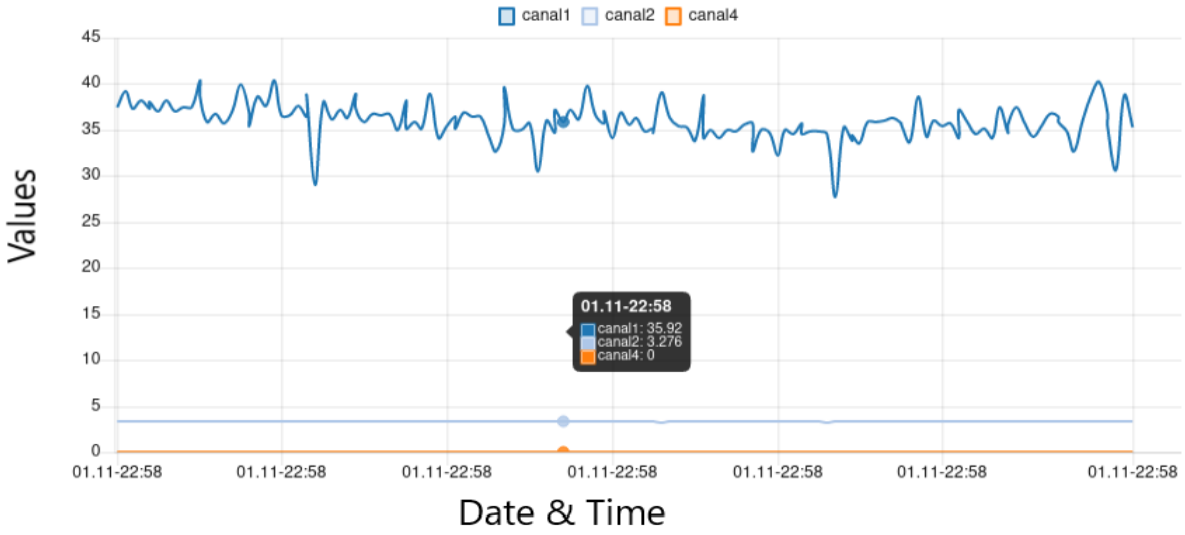


Figure III.45. Graphe d'évolution des valeurs acquises par les canaux analogiques

Interprétation :

On remarque que la température est entre 30 et 40 degrés, cette augmentation due à la mesure de température à travers le corps humaine, on a fait exprès de toucher le capteur à la main pour vérifier le fonctionnement de capteur, la température peut augmenter et diminuer. Pour les valeurs mesurées par le capteur d'humidité de sol sont constante car on n'a pas rajouté de l'eau au sol sinon cette valeur va diminuer.

- **Canaux numériques :**

En sélectionnant le fichier contenant les temps dont le buzzer est activé, et on affiche la table suivante sur l'IHM en cliquant sur le bouton Edit.

Date/Time	ETAT
2023-09-08T12:51:40.667Z	ON
2023-09-08T12:51:40.668Z	ON
2023-09-08T12:51:40.668Z	ON
2023-09-08T12:51:40.668Z	ON
2023-09-08T12:51:40.669Z	ON
2023-09-08T12:51:40.669Z	ON
2023-09-08T12:51:40.669Z	ON
2023-09-08T12:51:40.670Z	ON
2023-09-08T12:51:40.670Z	ON
2023-09-08T12:51:40.670Z	ON
2023-09-08T12:51:40.670Z	ON
2023-09-08T12:51:40.671Z	ON
2023-09-08T12:51:40.671Z	ON
2023-09-08T12:51:40.671Z	ON
2023-09-08T12:51:40.671Z	ON
2023-09-08T12:51:40.672Z	ON
2023-09-08T12:51:40.672Z	ON

Figure III.46. Affichage de sur des temps d'activation de buzzer acquis par la pine digitale sur l'IHM

En sélectionnant le fichier contenant les données acquises par l'entrée de compteur de fréquence, et en appuyant sur le bouton Edit on affiche la table suivante sur l'IHM dont on affiche le débit par heure de l'eau mesuré par le capteur de débit YF_S201.

Date&Time	Débit(L/h)
946684815	13
946684818	95
946684822	102
946684826	107
946684829	74
946684833	89
946684837	90
946684840	39
946684844	0
946684848	120
946684851	97
946684855	3
946684859	0

Figure III.47. Affichage des valeurs acquises par le canal7 (compteur d'impulsion)

En cliquant sur le bouton Graph vous pouvez voir l'évolution dans le temps de débit d'eau en fonction de Timestamp :

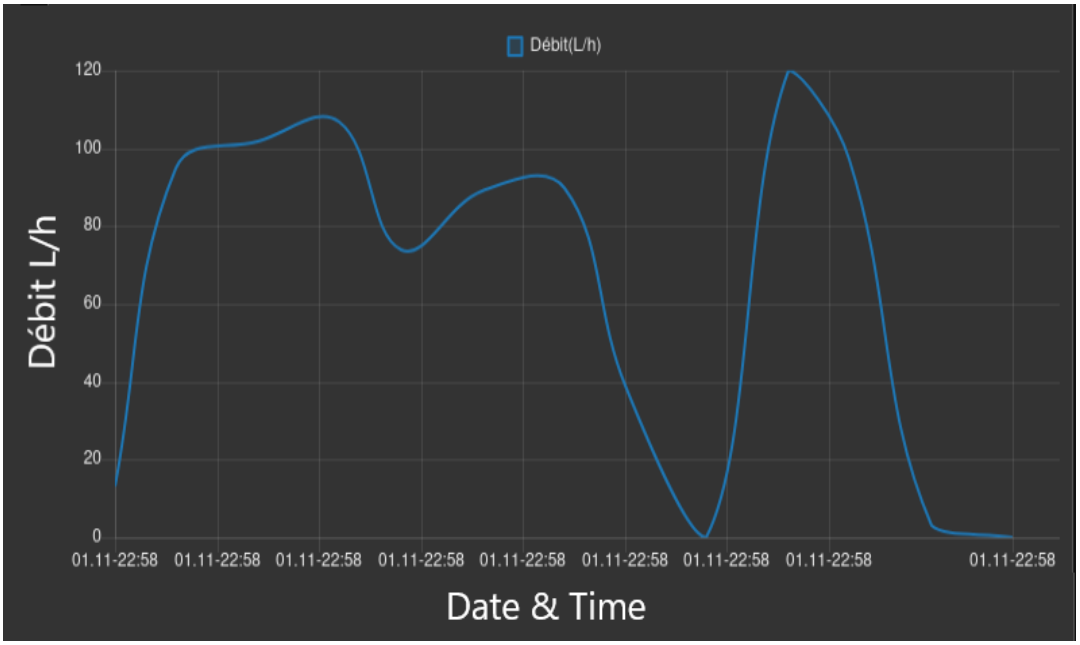


Figure III.48. Graphe d'évolution de débit d'eau dans le temps

Interprétation :

Le débit d'eau mesuré augmente et diminue puisque lors de test on ouvre le robinet de l'eau et on le fermer avec différente pression à chaque fois.

❖ **Canal d'I2C :**

En sélectionnant le fichier dont les valeurs acquis de l'entrée I2C sont enregistré puis avec le bouton Edit on affiche la table suivante sur l'IHM, qui contient les valeurs de température et d'humidité mesurées par le capteur TH02.

Timestamp	Temperature	Humidite
946684803	26	75.6
946684807	26	75.6
946684811	26	75.6
946684815	26	75.6
946684819	26	75.6
946684823	26	75.6
946684827	26	75.6
946684831	26	75.6
946684835	26	75.6
946684839	26	75.6
946684843	26	75.6
946684847	26	74.8
946684851	26	78

Figure III.49. Affichage des valeurs acquis de port I2C sur l'IHM

En cliquant sur le bouton Graph on obtient le graphe suivant qui représente l'évolution dans le temps (Timestamp) des valeurs représentées sur la table ci-dessus.

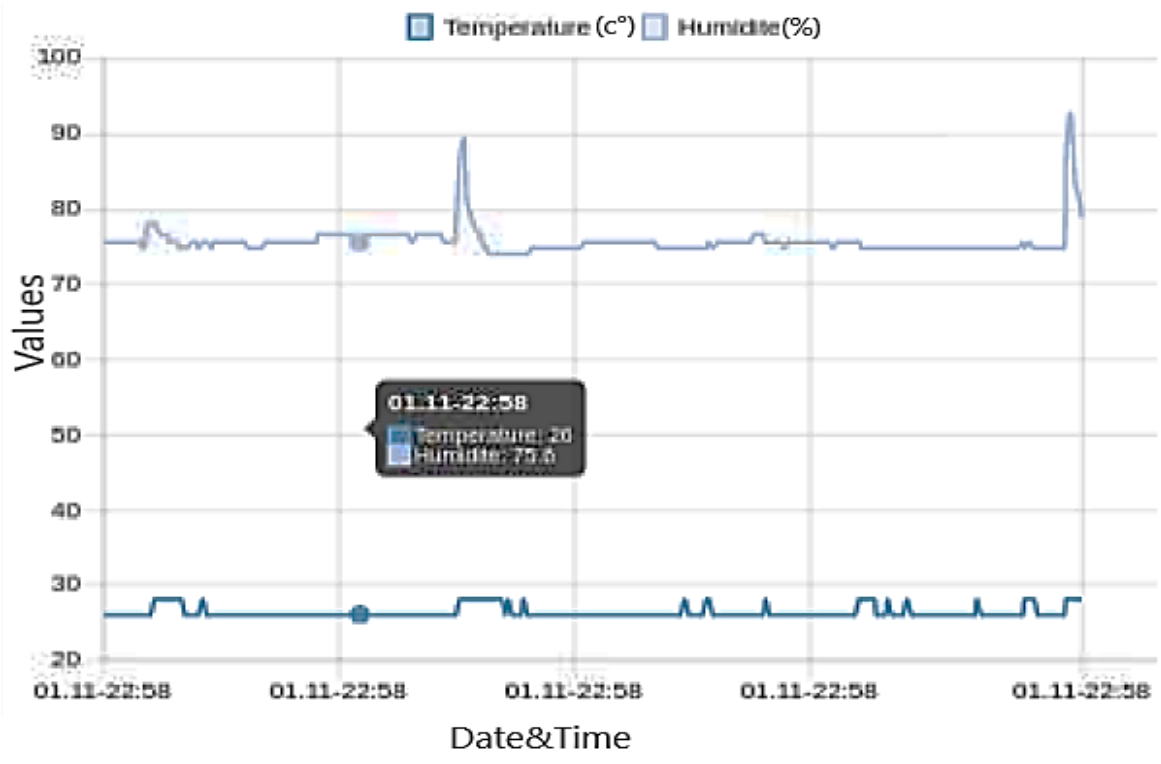


Figure III.50. Graphe d'évolution de température et d'humidité dans le temps

Interprétation :

La température mesurée par le capteur TH02 est de 26 C°, qui représente la température ambiante dans ce temps enregistré, et l'humidité mesuré par ce capteur est de 75%, elle augmente parfois à cause de courant d'air passant.

❖ **Canaux séries :**

En sélectionnant le fichier dont les valeurs acquis de l'entrée UART et RS485 sont enregistré puis avec le bouton Edit on affiche la table suivante sur l'IHM, qui contient les valeurs de température mesurées par le capteur interne de la carte STM32F446RE.

Date&Time	UART	RS485
2023-09-08T09:58:34.183Z	32.63	32.63
2023-09-08T09:58:34.183Z	33.28	33.28
2023-09-08T09:58:34.183Z	32.63	32.63
2023-09-08T09:58:34.183Z	33.28	33.28
2023-09-08T09:58:34.183Z	32.63	32.63
2023-09-08T09:58:34.184Z	33.28	33.28
2023-09-08T09:58:34.184Z	32.31	32.31
2023-09-08T09:58:34.184Z	32.63	32.63
2023-09-08T09:58:34.184Z	32.95	32.95
2023-09-08T09:58:34.184Z	32.63	32.63
2023-09-08T09:58:34.184Z	33.28	33.28
2023-09-08T09:58:34.185Z	32.95	32.95
2023-09-08T09:58:34.185Z	33.6	33.6
2023-09-08T09:58:34.185Z	32.95	32.95
2023-09-08T09:58:34.185Z	33.28	33.28
2023-09-08T09:58:34.185Z	32.95	32.95
2023-09-08T09:58:34.186Z	33.6	33.6

Figure III.51. Affichage des valeurs acquises de port série sur l'IHM

En cliquant sur le bouton Graph on obtient le graphe suivant qui représente l'évolution dans le temps (Timestamp) des valeurs représentées sur la table ci-dessus.

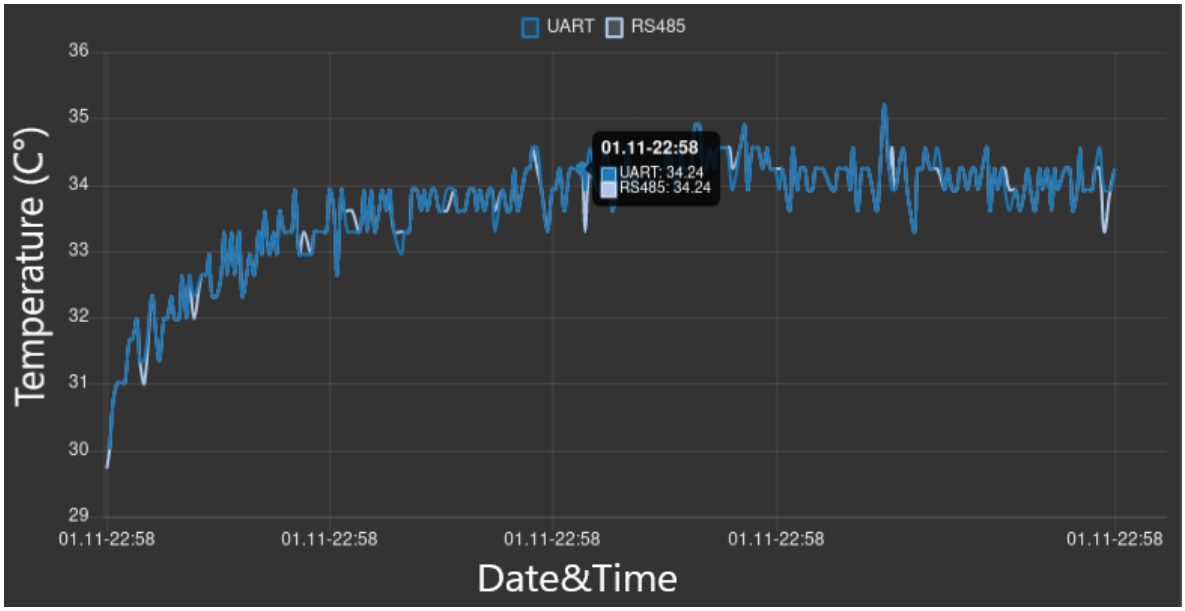


Figure III.52. Graphe d'évolution de la température dans le temps acquise par les ports séries

Interprétation :

La température mesurée par le capteur interne de MCU STM32F446RE augmente à cause de l'utilisation de MCU qui commence à s'échauffer à fur et à mesure de son travail.

- **Interprétation des résultats :**

On remarque que les valeurs de la température ambiante mesurée par le capteur LM35, celui intégré sur la carte STM32 et TH02 donnent des valeurs proches entre 30 et 40 (C°).

Les valeurs mesurées par le capteur d'humidité de sol sont égales à la valeur de la tension dont lequel est alimenté ce qui montre que le sol dont on l'a placé est sèche, dans le cas contraire la tension diminuera.

Le débit de l'eau mesuré par le capteur YF-S201 est élevé, ce qui montre une consommation élevée de l'eau.

- **Remarque :** les résultats obtenus peuvent être exploités dans plusieurs domaines surtout l'agronomie, la météo et le contrôle environnementale.

III.8. Conclusion

Dans ce chapitre nous avons décrits les différentes étapes de programmation et de réalisation de notre DataLogger ainsi son interface, beaucoup de détails ont étaient donnés sur chaque étape pour bien comprendre la procédure de conception et de réalisation de notre système. Pour confirmer le bon fonctionnement de notre datalogger, un ensemble de tests ont étant réalisées dans cette optique.

Les résultats atteints montrent et confirment le bon fonctionnement de notre système à travers différents capteurs utilisés et à travers même des graphes obtenus des données acquises et visualisées à travers notre IHM qui représente une autre bonne partie de ce travail.

Conclusion générale

Conclusion Générale

La réalisation de ce travail a permis d'explorer en profondeur les défis, les opportunités et les réalisations dans le domaine en constante évolution de l'Internet des Objets (IoT).

Ce travail avec la recherche approfondie effectuée, s'est concentrée sur la réalisation d'un DataLogger avec plusieurs entrées ainsi la capacité de transférer des données filaires et sans fil basées sur la technologie LoRa, destinée à collecter, stocker et transmettre efficacement des données provenant de capteurs y'est connectés à travers l'une de ses entrées. À travers ce projet, plusieurs objectifs ont été atteints, mettant en lumière l'utilisation de la nouvelle technologie LoRa dans ce domaine.

La conception et la réalisation du DataLogger ont été réalisées avec succès, combinant une approche matérielle et une approche logicielle par lesquelles, nous avons répondu au cahier des charges que nous avons décrit dans le chapitre II. Ce DataLogger a été conçu pour être efficace en termes de consommation d'énergie et d'espace de mémoire de stockage, fiable dans la transmission de données, adapté à une utilisation en conditions réelles à travers une interface interactive qui permet à l'utilisateur de configurer le DataLogger et de visualiser les données à n'importe quel moment et à distance à partir d'un ordinateur ou d'un appareil portable. Nous avons conçu et adopté un mécanisme de transfert des données automatique lors des opérations de son fonctionnement, de sorte à éviter une éventuelle saturation de l'espace de stockage, d'où il pourra fonctionner sans aucun problème de saturation mémoire.

Les contributions de ce projet à travers sa conception ont des implications potentielles dans divers domaines d'application, notamment la surveillance environnementale, l'agriculture intelligente, la gestion de l'énergie et bien d'autres. Le DataLogger connecté via LoRa offre une solution abordable et adaptable pour des applications de l'IoT à grande échelle.

Cependant, il est important de noter que le domaine de l'IoT continue à évoluer rapidement, avec de nouvelles technologies et des normes émergentes. Pour rester à la pointe de ce domaine, il est nécessaire de surveiller de près les développements futurs et d'adapter les solutions en conséquence. Des améliorations continues, telles que l'intégration de l'intelligence artificielle pour l'analyse des données, pourraient renforcer davantage l'efficacité et l'utilité des DataLoggers connectés via LoRa.

L'infrastructure de communication LoRa a été développée pour permettre une connectivité stable et à longue portée, offrant une solution adaptée aux environnements où l'accès à Internet est limité ou coûteux. Les résultats des tests expérimentaux ont démontré les performances du DataLogger dans des scénarios de terrain, confirmant sa capacité à collecter et à transmettre des données de manière fiable.

Références Bibliographiques

Liste des références bibliographiques

[1] James P.Karban and A.F.C.Sherrington. Data Acquisition and Processing in Biology and Medicine.1997,p.3

[2] John Park. Practical Data Acquisition for Instrumentation and Control Systems.2003,p. 1

[3] Web [consulté : 15/03/2023] :Digital Data Loggers - Lascar Electronics

[4] H.S.Kalsi. Electronic Instrumentation.Third Edition. Copyright © 2010, 2004, 1995 by Tata McGraw Hill Education Private Limited

[5] Jacob Fraden. Handbook of Modern Sensor. 5ème edition. © Springer International Publishing Switzerland 2004, 2010, 2016

[6] Web [consulté : 16/03/2023] : https ://www.campbellsci.com/data-loggers

[7] Web [consulté : 16/03/2023] : https ://www.omega.fr/prodinfo/enregistreurs-de-donnees.html

[8] Web [consulté : 16/03/2023] : Qu'est-ce qu'un Datalogger ou Enregistreur de Données ? (ip-systemes.com)

[9] Web[consulté :20/03/2023]:https://www.novarent.fr/datalogger-quest-ce-que-cest-types-applications-et-comment-ca-marche/

[10] Web [consulté : 25/03/2023] : https ://www.dataq.com/data-logger/

[11] Web[consulté :28/03/2023] :https://www.electronicdesign.com/technologies/iot/article/21801342/improve-datalogging-technique

[12] M.B.Waghmare,Dr.P.N.Chatur.Temperature and Humidity Analysis using Data Loggerof Data Acquisition System : An Approach.International Journal of Emerging Technology and Advanced Engineering.Website : www.ijetae.com (ISSN 2250-2459, Volume 2, Issue1, January 2012)

[13] Web [consulté : 5/04/2023] : https ://dewesoft.com/blog/data-logger-vs-daq-system

[14] Web[consulté :25/03/2023]:https://www.novarent.fr/datalogger-quest-ce-que-cest-types-applications-et-comment-ca-marche/**.**

[15] Pradeeka Seneviratne.Beginning LoRa Radio Networks with Arduino. Copyright© 2019. ISBN-13 (electronic): 978-1-4842-4357-2.https ://doi.org/10.1007/978-1-4842-4357-2.

[16] Marcelo G.Griese et João H.Kleinschmidt. Performance Analysis of a System for Vehicle Identification Using LoRa and RFID. International Conference on Green, Pervasive, and Cloud Computing.2019.pp 115–127.

[17] Web [consulté : 20/04/2023] : https ://www.semtech.com

[18] Document technique : LoRaWANT. M. Specifications,V1.0 .LoRa Alliance

[19] Ladislav.Polak1·Jiri.Milos. Performance analysis of LoRa in the 2.4 GHz ISM band: coexistence issues with Wi-Fi.© Springer Science+Business Media, LLC, part ofSpringer Nature 2020

[20] Sylvain MONTAGNY. LoRa-LoRaWAN et l'Internet des Objets.url :www.univ-smb.fr/lorawan.

[21] Datasheet of STM32F4: st.com/resource/en/reference_manual/dm00031020-stm32f405-415-stm32f407-417-stm32f427-437-and-stm32f429-439-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

[22] Web [consulté : 10/05/2023] : Guide: Connecting your debugger | STM32-base project
[23] Web [consulté : 15/05/2023] : SD Standard Overview | SD Association (sdcard.org)
[24] Web [consulté : 15/05/2023]: STM32 USB MSC (Mass Storage Class) » ControllersTech

[25] Document technique: Physical Layer Simplified Specification version 9.0 by the SD Card Association and the SD Group
[26] Web [consulté : 20/05/2023] : Image of the LM35 Module [6] | Download Scientific Diagram (researchgate.net)

[27] Web [consulté : 20/05/2023] : capteur de pression 1/2 IP68 Gaz mâle signal 4/20 mA - Paratronic
[28] Web [consulté : 20/05/2023] : Sensor MQ135 | orgontec

[29] Web [consulté : 21/05/2023] : ACS712 Current Sensor • Wolles Elektronikkiste (wolles-elektronikkiste.de)

[30] Web [consulté : 21/05/2023] : Capteur d'humidité du sol | YoupiLab Components

[31] Web [consulté : 21/05/2023] : Capteur de température et d'humidité Grove 101020074 Seeed Studio - Capteurs de t° et humidité | GO TRONIC
[32] Web [consulté : 21/05/2023] : Sonde température d'ambiance SD - Modbus RS485 (guilcor.fr)

[33] Web [consulté : 21/05/2023] : CAPTEUR DE TEMPÉRATURE AVEC TIGE ET TÊTE EN PLASTIQUE AVEC PROTOCOLE CAN DE SORTIE KDS 124 (guilcor.fr)

[34] Web [consulté : 21/05/2023] : 1/8 inch Water Flow Sensor - YF-S401 buy online at Low Price in India - ElectronicsComp.com
[35] Web [consulté : 25/05/2023] : Pt100 Sensor Explained | Working Principles - RealPars

[36] Web [consulté : 25/05/2023] : Power Supply Module 4 ~20mA to 0~3.3V/5V/10V Current to Voltage Converter Signal Conversion Module DC 12V 24V Voltage Transmitter (droking.com)

[37] Web [consulté : 25/05/2023] : Des boucles de courant 4...20 mA - Les principes de base (prelectronics.com)

[38] Web [consulté : 26/05/2023] : 电压转电流模块电路设计原理解析_电压转电流电路_>行者<的博客-CSDN博客

[39] Web [consulté : 28/05/2023] : UART Protocols and Applications in Embedded Systems
[40] MFE. Nahla BELLOULA.Design of a Modbus RTU to Lora gateway.ENST.30 July 2021.
[41] Web [consulté : 29/05/2023] : Arduino DMX Modules – Theatre Lighting Workshops

[42] Web [consulté : 29/05/2023] : Bus CAN : Spécifications du bus CAN | Techniques de l'Ingénieur (arn.dz)
[43] Web [consulté : 29/05/2023] : MCP2551 Transceiver: Circuits, Pinout, and Datasheet (utmel.com)

[44] Web [consulté : 30/05/2023] : Chap6-I2C.pdf (espacetechnologue.com)

[45] Web [consulté : 30/05/2023] :USB Wiring Diagram, Connection, PinOut, Terminals - ETechnoG

[46] Web [consulté : 30/05/2023] : STM32 f4 or f1 lqfp100 - EasyEDA open source hardware lab (oshwlab.com)

[47] Web [consulté : 01/06/2023] : Ra-01/Ra-02 LoRa Module User Manual | 安信可科技 (ai-thinker.com)

[48] Web [consulté : 01/06/2023] :LoRa Schema mit SX1278 - Mikrocontroller.net

[49] Web [consulté : 02/06/2023] :MR-J3BAT - Dantona Industries - Battery, 3.6 V, AA (newark.com)

[50] Web [consulté : 03/06/2023] : raspberry-pi-3-b-plus-product-brief.pdf (raspberrypi.com)

[51] Web [consulté : 03/06/2023] :STM32F407VGT6 - STM32F4XX M | STM32-base project

[52]lien de la bibliothèque LoRa : sx1278-LoRa-RaspberryPi/ at master · YandievRuslan/sx1278-LoRa-RaspberryPi · GitHub

[53] lien de logiciel de dessin de PCB : https://fritzing.org/

[54] lien de logiciel de conception 3D : https://www.tinkercad.com/

[55] site de l'entreprise ALCADWORKS : AlcadWorks | AlcadWorks

Annexes

Annexe I

A.I.1. Nombre de canaux des DataLoggers

Les dataloggers sont disponibles en simples ou multi-canaux (jusqu'à 32 canaux), on trouve ceux qui :

- Offrent moins d'options de connexion de capteurs :
  - Offre une option économique pour de installations qui ne nécessitent que quelques capteurs à connecter.
  - La quantité et le type de données pouvant être collectées à partir d'un seul site sont limités.
  - Peut être limité à de simples mesures analogiques.
  - Ils sont généralement plus petits et plus légers que les enregistreurs de données avec plusd'options de connectivité.
- Offrant plus d'options de connectivité :
  - Plus rentable lorsque de nombreux capteurs sont nécessaires.
  - Il est souvent modulaire et permet d'étendre le système d'acquisition de données grâce à des bornes ou des connexions supplémentaires pour mesurer plus de capteurs, répondant aux besoins actuels des applications et permettant une expansion future.

A.I.2. Calcul de la période d'enregistrement d'un DataLogger

La période d'enregistrement peut être calculée à l'aide de la formule suivante :

Période d'enregistrement = (Capacité de stockage/ Taille du fichier de données) / Taux d'échantillonnage.

Par exemple, supposons que vous utilisez un datalogger avec une capacité de stockage de 1 Go, un taux d'échantillonnage de 100 Hz et que la taille moyenne d'un enregistrement de données est de 1 Ko. La période d'enregistrement serait alors calculée comme suit :

Période d'enregistrement = (1 Go / 1 Ko) / 100 Hz = 10 000 secondes

Cela signifie que le datalogger peut enregistrer des données pendant environ 10 000 secondes (soit environ 2,8 heures) avant de remplir sa capacité de stockage.

A.I.3. Connexion des modules d'extension

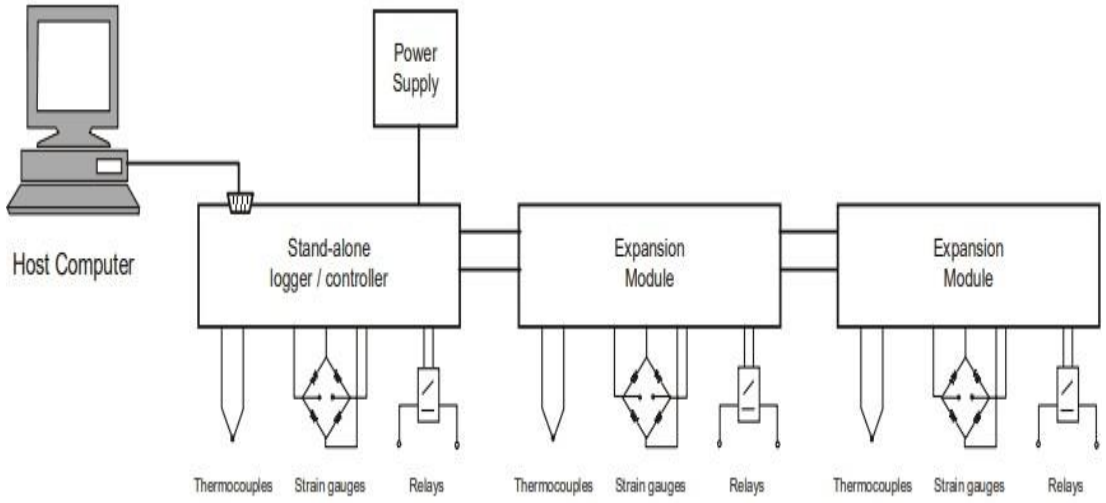


Figure A.I.3.1. Connexion des modules d'extension

A.I.4. Connecteur borne à vis



Figure A.I.4.1. Photo des connecteurs borne à vis

Annexe II

A.II.1 Principe de la modulation LoRa

LoRa utilise le CSS (Chirp Spread Spectrum) modulation qui utilise une méthode d'étalement de fréquence comme technique de modulation. Les impulsions dites chirp sont envoyées sous forme de symboles, qui augmentent ou diminuent la fréquence LoRa en continu au fil du temps. La transmission des données est alors réalisée par la séquence séquentielle de ces impulsions de chirp.

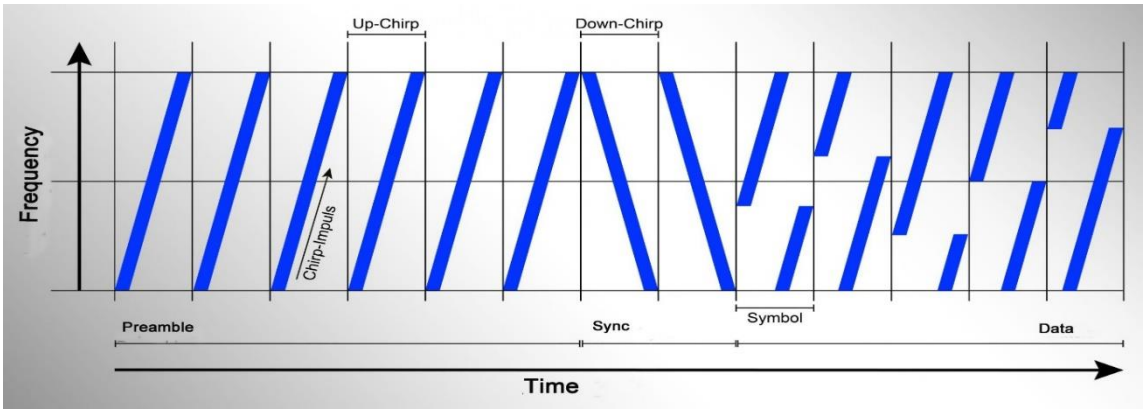


Figure A.II.1.1. Modulation LoRa

A.II.1.1. Propriétés spéciales

Puisque LoRa fonctionne dans les bandes de fréquences ISM (433 MHz, 868 MHz et 915 MHz), la puissance d'émission rayonnée est limitée. Afin d'avoir une plus grande portée radio que les types de modulation conventionnels tels que Pour atteindre FSK (Modulation par déplacement de fréquence), la sensibilité du récepteur a été considérablement améliorée avec LoRa.

Le récepteur LoRa peut toujours recevoir et décoder avec succès un signal LoRa utile jusqu'à 20 dB en dessous du niveau de bruit, ce qui se traduit par une sensibilité du récepteur d'un maximum de -149 dBm. Par rapport à la sensibilité FSK maximale d'env.. –125 dBm à -130 dBm, LoRa offre une amélioration significative. Avec le récepteur FSK, le signal ne peut être décodé avec succès que si le signal utile est d'env.

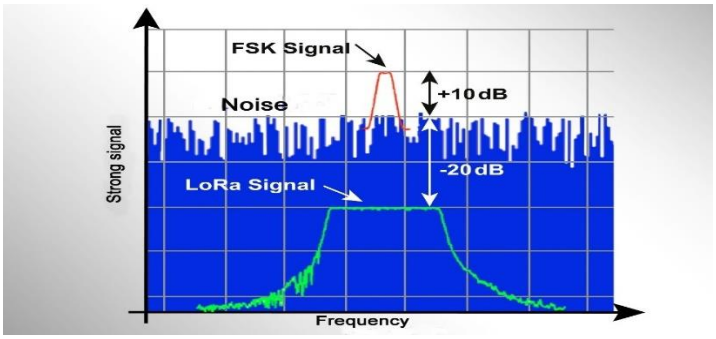


Figure A.II.1.2. Sensibilité de signal LoRa aux bruits

Grâce à la propriété que LoRa peut encore recevoir avec succès un signal utile jusqu'à 20 dB en dessous du niveau de bruit, la robustesse aux interférences radio est nettement meilleure que celle du FSK. Les systèmes FSK ne fonctionnent correctement que si le signal d'interférence est au moins 10 dB plus faible que le signal utile. Dans le meilleur des cas, Les systèmes LoRa peuvent toujours recevoir le signal utile si le signal d'interférence est 20 dB plus fort que le signal utile.

A.II.1.3. Limites

À partir du graphique ci-dessus, vous pouvez voir que LoRa peut recevoir environ 30 dB signaux plus faibles qu'avec FSK. Toutefois, il y a deux restrictions qui relativisent quelque peu cette grosse différence.

- Première, la modulation LoRa est à large bande par rapport à la modulation FSK, ce qui signifie que le niveau de bruit du récepteur LoRa est généralement supérieur à celui du récepteur FSK. Plus précisément, doubler la bande passante augmente le niveau de bruit de 3 dB.
- Deuxièmement, LoRa ne peut recevoir un signal utile que jusqu'à 20 dB en dessous du niveau de bruit à des débits de données très lents de ≤ 0.5 kbit / s. Dès que le débit de données est augmenté, soit le rapport signal / bruit négatif augmente encore vers zéro, soit la bande passante doit être augmentée davantage, qui à son tour augmente le niveau de bruit.

A.II.2. Schéma bloc de l'alimentation de la carte STM32F407VGT6

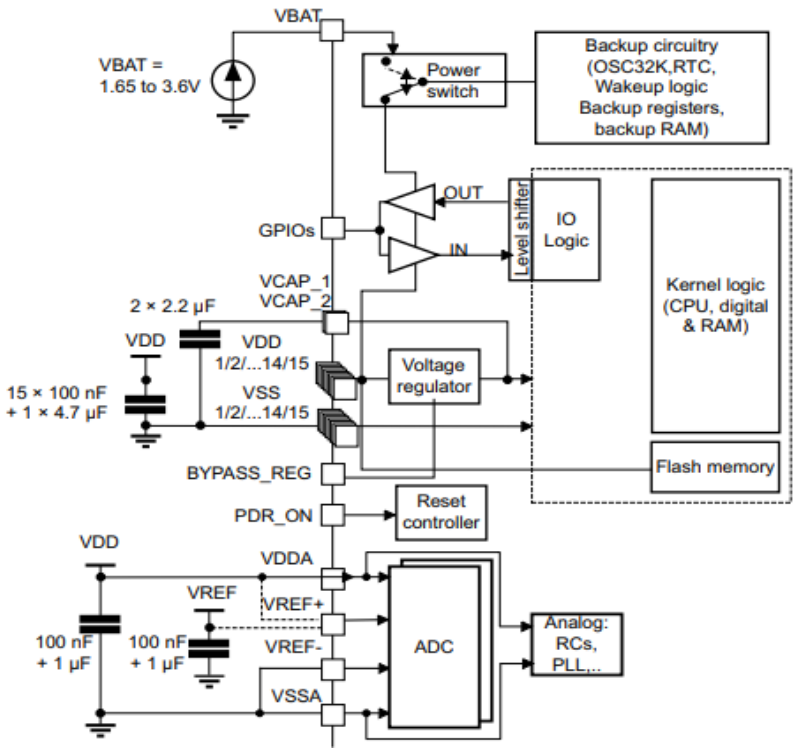


Figure A.II.2.1. Schéma bloc de l'alimentation de la carte STM32F407VGT6

Annexe III

A.III.1. Configuration de DMA

On a utilisé le mode DMA avec quelques modules tel que l'ADC, UART et SDIO, leurs configurations sur cubeMX est illustrées dans les figures ci_dessous pour chaque module.

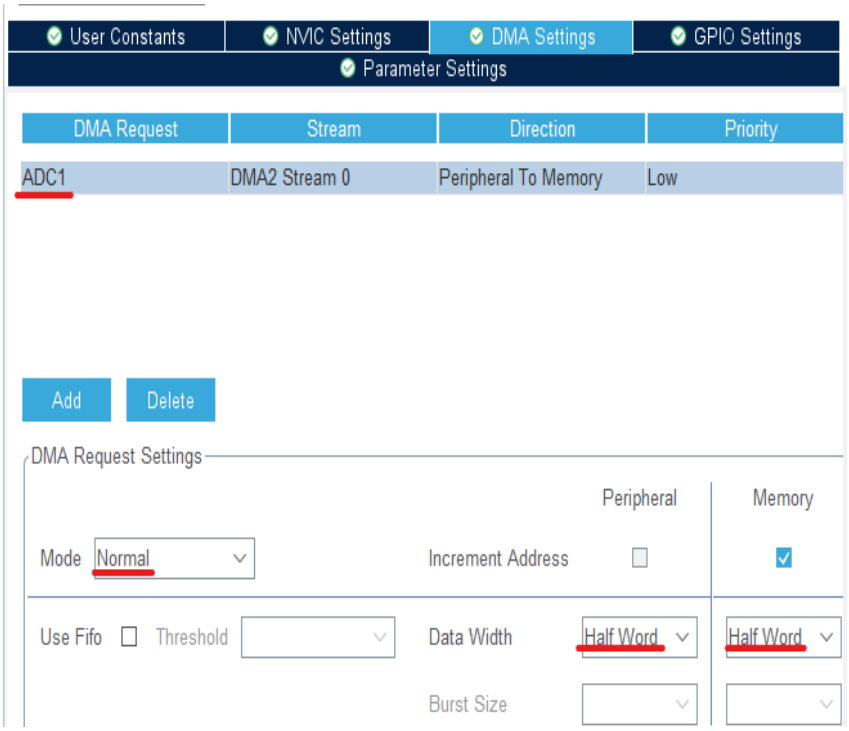


Figure A.III.1. Paramètre de DMA de l'ADC1 sur cubeMX

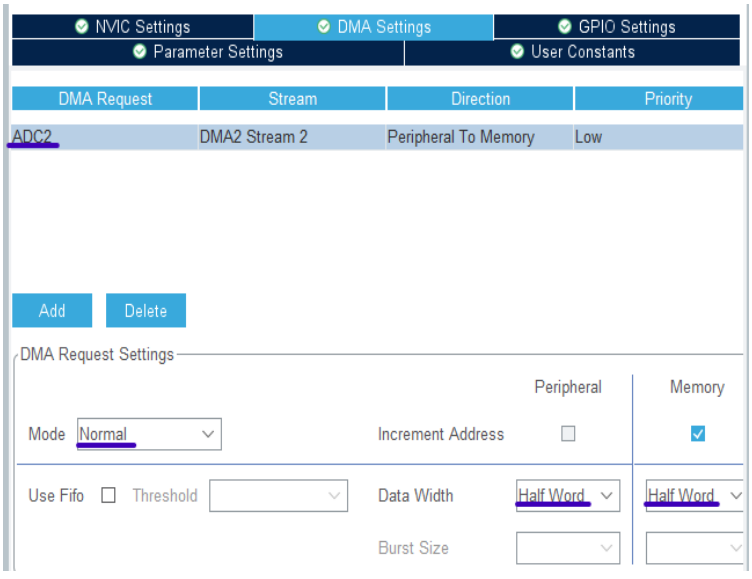


Figure A.III.2. Paramètre de DMA de l'ADC2 sur cubeMX

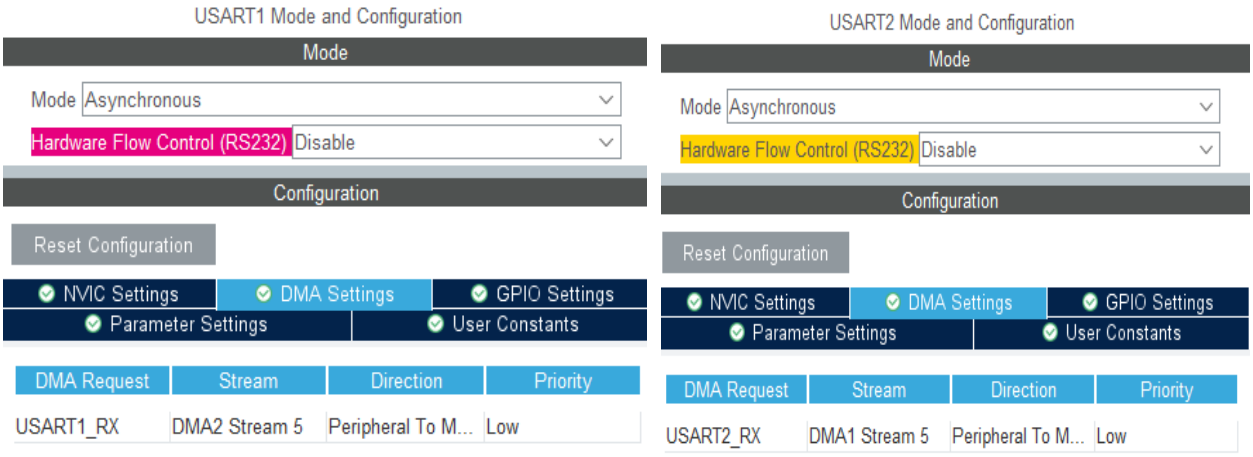


Figure A.III. 3. Configuration DMA pour UART1 et UART2

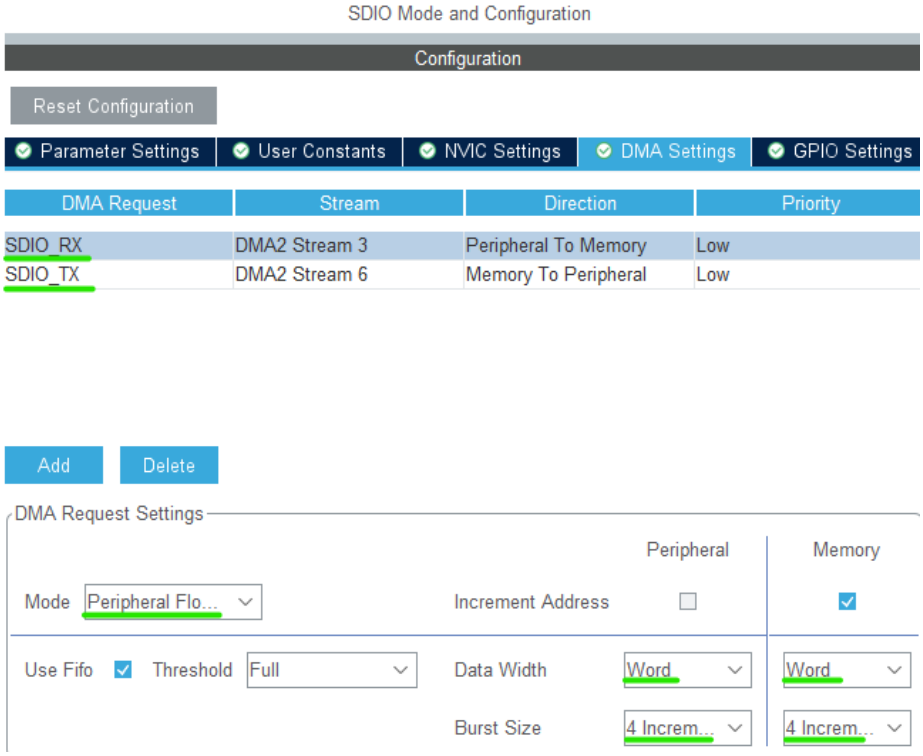


Figure A.III.4. Configuration DMA pour SDIO Tx et Rx sur cubeMX

A.III.2. La fonction de conversion de date/temps en forme UNIX

Pour économiser l'espace de stockage on a mis la date/temps obtenu de RTC en forme UNIX et ceci utilisant la fonction suivante :

```
/////Converter la date et le temps en forme Timestamp////
void get_time(void){
RTC_TimeTypeDef ti;
RTC_DateTypeDef da;
(void) HAL_RTC_GetTime(&hrtc, &ti, RTC_FORMAT_BIN);
(void) HAL_RTC_GetDate(&hrtc, &da, RTC_FORMAT_BIN);
struct tm tim = {0};
tim.tm_hour = ti.Hours;
tim.tm_min = ti.Minutes;
tim.tm_sec = ti.Seconds;
tim.tm_year = da.Year+100;
tim.tm_mon = da.Month -1;
tim.tm_mday = da.Date;
tim.tm_isdst = 0;
if (ti.DayLightSaving == RTC_DAYLIGHTSAVING_SUB1H)
{
tim.tm_isdst = -1;
}
else if (ti.DayLightSaving == RTC_DAYLIGHTSAVING_ADD1H)
{
tim.tm_isdst = 1;
}
dt.UnixDtTm=mktime(&tim);
//return mktime(&tim);
}
```

Figure A.III.2.1 Le code de la fonction get_time()

A.III.3. Configuration du date/temps

Au début d'utilisation de DataLogger l'utilisateur doit sélectionner la date et le temps sur l'IHM et seront envoyer au DataLogger à partir de l'USB, le code ci_dessus permet de traiter le message reçu et configurer RTC utilisant les fonctions HAL_RTC_SeDate et SetTime.

```
if(strncmp(buffer,Dt,2) == 0){
     memcpy(setDT, buffer + 2, 20);
     token = strtok(setDT, delim);
     sDate.Year = atoi(token);
     for (int k = 0; k < 5; k++) {
         token = strtok(NULL, delim);
             if (k == 0) sDate.Month = atoi(token);
             if (k == 1) sDate.Date = atoi(token);
             if (k == 2) sTime.Hours = atoi(token);
             if (k == 3) sTime.Minutes = atoi(token);
             if (k == 4) sTime.Seconds = atoi(token);
         }
     if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN) != HAL_OK ||
             HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) != HAL_OK) {
            //  Handle error
         }
     memset(buffer,'\0',64);
     memset(setDT,'\0',20);
     i=0;
 }
```

Figure A.III.3.1. Code de configuration de la date et le temps

A.III.4. Configuration des taux d'échantillonnage :

Les taux d'échantillonnage aussi sont sélectionnés par l'utilisateur sur l'IHM, et les données seront reçus par USB, et on change ces taux en envoyant la valeur au registre de preload de TIMER considéré, et ceci pour Tad1 et Tad2, pour Tserie va être comparé directement avec le flagSerie.

```
Configuration des taux d'échantillonnages
  if(strncmp(buffer,config,2) == 0){
       mode=1;
       memcpy(val, buffer + 2,6);
       token1 = strtok(val, delim1);
       Tad1 = atoi(token1);
       for (int K = 0; K < 2; K++) {
           token1 = strtok(NULL, delim1);
           if(K == 0) Tad2 = atoi(token1);
           if(K == 1) Tserie = atoi(token1);
       }
      TIM2->ARR =  Tad1*2000;
      TIM5->ARR =  Tad2*2000;
       memset(buffer,'\0',64);
       memset(val,'\0',8);
       HAL_Delay(200);
       mode=2;
   }
```

Figure A.III.4.1. Code de traitement de la réception des taux d'échantillonnage

A.III.5. Fonctions de déclenchement de Timer et de DMA

Les fonctions suivantes font l'objet de déclenchement des TIMER pour commencer le comptage, sont mis avant la boucle infinie, et celle de DMA pour l'echange des adresses de mise des valeurs acquises par le module correspondant et de début de conversion pour ADC et de réception pour l'UART.

```
//////Scan ADC channels///////
HAL_ADC_Start_DMA(&hadc1,ADC1val,4);
HAL_TIM_Base_Start(&htim2);

//////Scan ADC2(4-20mA) channel///////
HAL_ADC_Start_DMA(&hadc2,ADC2val,1);
HAL_TIM_Base_Start(&htim3);

///////start the timer4 FOR I2C scan and frquency /////
HAL_TIM_Base_Start_IT(&htim6);

/////START COUNTER OF FREQ//////
__HAL_TIM_SetCounter(&htim1,0);
 HAL_TIM_Base_Start(&htim1);

 HAL_UART_Receive_DMA(&huart3,rx_dataUART, 5);
 HAL_UART_Receive_DMA(&huart2,rx_dataRS, 5);
```

Figure A.III.5.1. Code d''initialisation des TIMER et DMA

A.III.6. Fonctions callbacks

Dans le champ de code user4 on ajoute les fonctions callbacks :

- HAL_ADC_ConvCpltCallback : active le flagADC1 et flagADC2 lors de complétiotion de conversion de l'un d'entre eux.
- HAL_TIM_PeriodElapsedCallback : on met le programme qu'on veut traiter par l'interruption générée par le TIMER6.
- HAL_UART_RxCpltCallback : active le flaguart et le flagRS lorsque la reception d'un message par l'UART1 ou 2 sera complet.
- HAL_GPIO_EXTI_Callback : traite l'interruption générée par le GPIO pin.

```
////Fonction d'échantillonnage de l'ADC1 et l'ADC2///////////
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *AdcHandle) {
      /* This is called after the conversion is completed */
    if(AdcHandle->Instance==ADC1){
          flagADC1=1;
    }
    if(AdcHandle->Instance==ADC2){
         flagADC2=1;
    }
}
////////detect the state by gpio/////
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin==Digital_Pin){
        stat=1;
}}
//Fonction d'échantillonnage des entrées series//////
//et de redéclenchement de compteur de frequence/////
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if(htim==&htim6){
            flagSerie++;
            frequency=TIM1->CNT;
            __HAL_TIM_SetCounter(&htim1, 0);

}}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    if(huart->Instance == USART1){
        HAL_UART_Receive_DMA(&huart3,rx_dataUART,5);
        flaguart=1;
        }
    if(huart->Instance == USART3){
        HAL_UART_Receive_DMA(&huart2,rx_dataRS,5);
        flagRS=1;
``
```

Figure A.III.6.1. Fonctions Callbacks

A.III.7. Le fichier I2Cexemple

Les composants utilisant l'I2C nécessitant une configuration de certain registre sur le composant pour que le MCU puisse communiquer avec lui, on trouve cette configuration dans la datasheet de

composant, pour ceci on a créé un fichier I2Cexemple.c avec son header pour traiter certains capteurs, les fonctions qu'on a créées sont les suivantes :

```
HAL_StatusTypeDef ADS7828_readADC(uint8_t address, uint8_t pinCfg, uint16_t *adcdata);
HAL_StatusTypeDef MAX30205_readTemp(uint8_t dev_address, uint8_t config, float *temp);
HAL_StatusTypeDef TH02_TM(uint8_t address, uint8_t Cfg[5], uint16_t *DATA,uint16_t *DATAH);
void adc_toVolt(float vref, uint8_t res, uint16_t adcval, float *voltage);
void get_hm(uint16_t HM,float *Hvalue);
void get_temp(uint16_t TM,float *TMvalue);
```

A.III.8. Configuration et programmation de la carte STM32Nucleo64

Pour les entrées RS485 et UART on n'a pas trouvé des capteurs donc on a utilisé une autre carte STM32F446RE pour les tester, même pour tester le canal de fréquence on va utiliser une simple bliking led avec un délai.

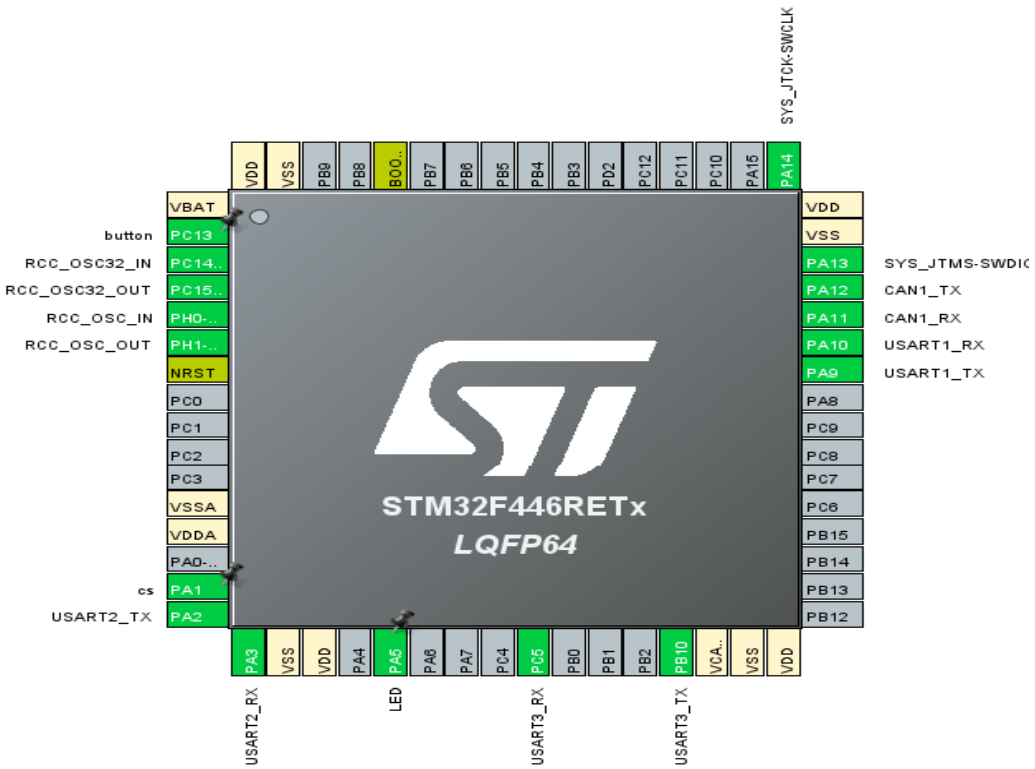


Figure A.III.8.1. Configuration de MCU F446RE sur cubeMX

❖ **Programme :**

On ajoute ces variables dans le champ des variables.

```
  uint32_t ADCval;
  char cle[5]="read";
  char tx_dataUART[7];
  char tx_dataRS[7];
  float temp,val_temp;
  uint8_t rx_dataUART[5];
  uint8_t rx_dataRS[5];

  int flag=0;
  int flagRS=0;
  int flagUART=0;

  ////var simulation de frequence et de pin digital
  int delai=1;
```

```
val_temp=ADCval*3.3/4096;
temp = ((val_temp-0.76)/0.0025)+25;
if(strcmp((char *)rx_dataUART,cle)==0){
    memset(rx_dataUART,'\0',5);
    sprintf(tx_dataUART,"%.2f",temp);
    send_dataUART(tx_dataUART);
    printf("sending :%s", tx_dataUART);
    HAL_Delay(500);
    memset(tx_dataUART,'\0',7);
}
if(strcmp((char *)rx_dataRS,cle)==0){
        memset(rx_dataRS,'\0',5);
        sprintf(tx_dataRS,"%.2f",temp);
        send_dataRS(tx_dataRS);
        printf("sending :%s", tx_dataRS);
        HAL_Delay(500);
        memset(tx_dataRS,'\0',7);
    }
if(HAL_GPIO_ReadPin(button_GPIO_Port,button_Pin)== GPIO_PIN_RESET){
    HAL_GPIO_WritePin(stat_GPIO_Port, stat_Pin, GPIO_PIN_SET);
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
    HAL_Delay(delai*100);
    delai++;
}
else{
    HAL_GPIO_WritePin(stat_GPIO_Port, stat_Pin, GPIO_PIN_RESET);
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
    HAL_Delay(delai*100);
}

  HAL_Delay(1000);
```

La carte STM32F446RE a un capteur de température intégré, lié avec l'ADC1 on peut l'activer et l'utiliser pour mesurer la température, de la même façon qu'on a configuré les modules d'ADC, UART, sur la carte STM32 utilisée pour le DataLogger sur cubeMX.

Lorsque la carte reçois le message clé « read » de DataLogger, sur l'un de ses ports va l'envoyer la valeur de température vers le port correspond, et ceci utilisant la fonction « send_dataUART » pour l'UART1 et « send_dataRS » pour (UART3) RS485 qui nécessite

l'activation de pin « cs » lors de transmission et la désactivation lors de réception. Elles sont placées avant la fonction main.c.

```
void send_dataRS(char *data){
    HAL_GPIO_WritePin(cs_GPIO_Port, cs_Pin, GPIO_PIN_SET);
    HAL_UART_Transmit(&huart3,(uint8_t *) data, 5, 1000);
    HAL_GPIO_WritePin(cs_GPIO_Port, cs_Pin, GPIO_PIN_RESET);
}
void send_dataUART(char *data){
    HAL_UART_Transmit(&huart1,(uint8_t *) data, 5, 1000);
}
```

Remarque : on a activé l'UART2 qui lié avec le port st_link, pour communique avec la console utilisant printf pour voir les résultats, et ceci en ajoutant le code suivant avant le main :

```
int _write(int file, char *ptr, int len)
{
  HAL_UART_Transmit(&huart2, (uint8_t *)ptr, len, HAL_MAX_DELAY);
  return len;
}
```

- **Test :**

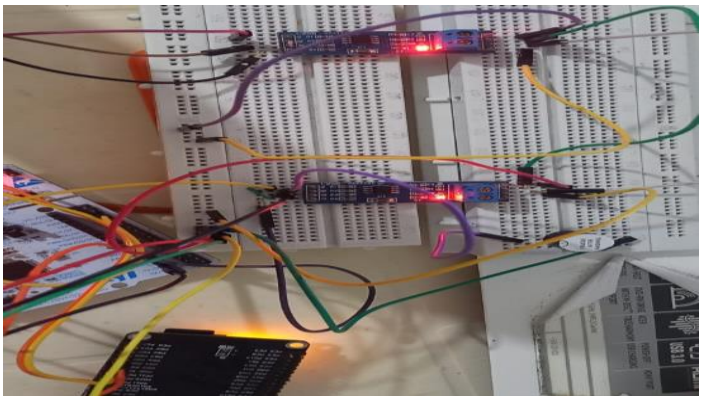


Figure A.III.8.2. Illustration de câblage

Résultat sur la console :

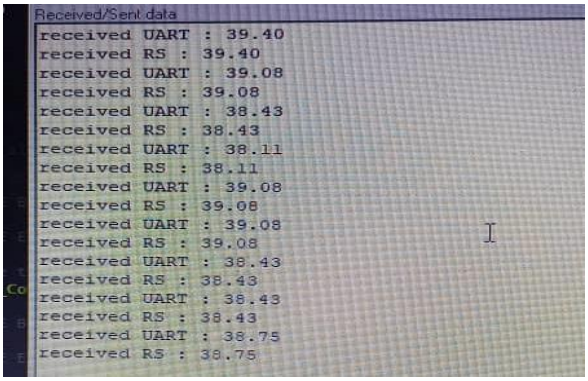


Figure A.III.8.3. Résultat de réception des données par DataLogger

Résumé

Français : Notre projet de fin d'études porte sur la conception et la réalisation d'un DataLogger connecté utilisant la technologie LoRa (Long Range) pour la communication sans fil et USB pour la communication filaire. Ce dispositif permettra la collecte de données à travers ses différentes canaux, telles que la température, l'humidité, et d'autres paramètres environnementaux, et leur transmission vers une base de données centrale. Cette solution offre une surveillance et une gestion efficaces des données pour diverses applications industrielles et environnementales.

Mots clés : DataLogger, LoRa, USB, Données.

English: Our final year project focuses on the design and implementation of a connected datalogger using LoRa (Long Range) technology for wireless communication and USB for wired communication. This device will enable data collection through various channels, such as temperature, humidity, and other environmental parameters, and transmit them to a central database. This solution provides efficient data monitoring and management for various industrial and environmental applications.

Key words: DataLogger, LoRa, USB, Data.

اللغة االعربية : يتعلق مشروعنا بتصميم وتحقيق DataLogger متصل باستخدام تقنية LoRa (بعيد المدى) للاتصالات اللاسلكية و USB للاتصالات السلكية. سيسمح هذا الجهاز بجمع البيانات من خلال قنواته المختلفة، مثل درجة الحرارة والرطوبة والمعايير البيئية الأخرى، وإرسالها إلى قاعدة بيانات مركزية. يوفر هذا الحل رصد البيانات وإدارتها بشكل فعال لمختلف التطبيقات الصناعية والبيئية .

الكلمات الرئيسية: DataLogger، LoRa، USB، البيانات.