



RAPPORT FINAL DE PROJET

SAE LoRaWAN - Semestre 3

Créé par :

Hugo Meleiro
Marius Deias

Responsables : M. Diallo & M. Laurent - GEII



1 Introduction

Au cours du semestre 3 de notre cursus en BUT Génie Électrique et Informatique Industrielle (GEII), nous nous lançons dans un projet axé sur la transformation d'un tricycle électrique en un véhicule "communicant" avec le monde extérieur en embarquant une carte mBED.

Le Projet SAE LoRaWAN est un projet innovant qui offre de nouvelles perspectives à la mobilité urbaine en libre-service. Il consiste à établir une réflexion pour équiper et gérer une flotte de tricycles électriques à Nice. Ce véhicule intelligent sera connecté au réseau LoRaWAN et pourra envoyer des informations sur la qualité de l'air. Il sera également géolocalisé pour permettre sa réservation. Ce projet s'inscrit dans le cadre du développement de la ville de Nice pour en faire une ville intelligente, une « smart city ». Il s'agit donc d'un projet écologique.

L'accent est particulièrement mis sur le développement des fonctions nécessaires à la communication entre tricycle et à une plateforme "Smart Cities IoT". Au cœur de cette initiative se trouve la conception d'une antenne LoRa à 868 MHz, intégrée au système pour assurer une connectivité fiable et une portée étendue. Lors de ce rapport, nous évoquerons les étapes de la conception et l'intégration de cette antenne grâce au logiciel Advanced Design System (ADS) concu par Keysight et RS DesignSpark PCB, puis la programmation nécessaire pour suivre le cahier des charges imposé.

Table des matières

1	Introduction	i
2	Présentation du projet	1
2.1	Objectifs	1
2.2	Spécifications techniques	1
2.3	Plan de développement	2
3	Électronique	3
3.1	Conception de la carte	3
3.2	Design de l'antenne	8
3.2.1	Définition du substrat	8
3.2.2	Design de l'antenne	10
3.2.3	Paramètres S	11
3.2.4	Far Field	12
3.2.5	Circuit d'adaptation avant rétrosimulation	13
3.2.6	Rétrosimulation	14
3.2.7	Adaptation de l'antenne	16
3.3	Réalisation du PCB	18
3.4	Soudure	20
4	LoRaWAN	22
4.1	Le protocole	22
4.1.1	Architecture	22
4.1.2	Fonctionnement	23
4.1.3	Caractéristiques	24
4.2	Les capteurs	25
4.2.1	Température et humidité	25
4.2.2	Luminosité	26
4.2.3	Gaz	26
4.2.4	Localisation GPS	27
4.2.5	Identification RFID	28
5	Programmation	29
5.1	Microcontrôleur	29
5.1.1	mBED	29

5.1.2	Configuration	30
5.1.3	Programme	32
5.2	NodeRED	37
5.2.1	Récupération des données	39
5.2.2	Affichage des données	40
5.2.3	Sauvegarde des données	42
5.2.4	Vérification du badge	42
5.2.5	Service HTTP	46
5.3	InfluxDB	46
5.4	Application mobile	48
6	Conclusion	51
7	Sources & Bibliographie	52

2 Présentation du projet



FIGURE 1 – *La Kiffance - Naps*

2.1 Objectifs

- Rendre le tricycle électrique "communicant" avec le monde extérieur et capable de se géolocaliser.
- Collecter des données environnementales (température, humidité, luminosité, pollution, GPS) ainsi que l'autorisation de démarrage du véhicule.
- Stocker les données dans une base de données.
- Créer des tableaux de bords et des applications pour visualiser et analyser les données.

2.2 Spécifications techniques

Carte mBED

- Disco L072
- Protocole LoRaWAN

Capteurs

- Température/humidité : Grove Temperature/Humidity Sensor
- GPS : Grove GPS
- Luminosité : Grove Sunlight Sensor
- Pollution : Grove Multichannel Gas Sensor
- Lecteur de badge RFID/NFC : MFRC522

Actionneurs

- 3 voyants LED (rouge, orange, vert)
- Contacteur de démarrage (MOSFET)

2.3 Plan de développement

Phase 1 (2 semaines)

- Prise en main du serveur d'activation et du serveur de réseau TheThingsNetwork
- Ouverture d'un compte et création d'un environnement pour la connexion d'un device
- Connection de la carte mBED à la plateforme TheThings Network
- Design du schématique sur DesignSpark PCB de la carte Mezzanine
- Design de l'antenne 868MHz sur ADS

Phase 2 (4 semaines)

- Mise en place de différents scénarios de récupération et d'analyse des données en provenance de la carte mBED
- Récupération des données en provenance de la carte (t°, humidité, GPS, badges...)
- Création de tableaux de bords à partir des données
- Authentification et validation à l'aide de la carte NFC, envoie d'une autorisation de démarrage au véhicule
- Design et impression du PCB de la carte Mezzanine

Phase 3 (4 semaines)

- Mise en place de règles, stockage des données dans une Base de Données InfluxDB
- Exploitation de la Localisation GPS pour la géolocalisation sur une carte
- Mise en place API HTTP pour récupérer les données au format JSON à partir d'un smartphone
- Soudures et tests de la carte Mezzanine

Conclusion

Ce cahier des charges définit les objectifs, les fonctionnalités, les spécifications techniques et le plan de développement du projet de SAE LoRaWAN sur tricycle électrique. Il constitue le document de référence pour le projet.

3 Électronique

3.1 Conception de la carte

Comme pour les projets des semestres précédents, nous avons été amenés à concevoir une carte électronique, la carte mezzanine, sur le logiciel RS DesignSpark PCB. En revanche, la conception diffère des cartes des projets précédents, car premièrement la carte comporte des composants CMS, et une partie de la carte est à concevoir sur un nouveau logiciel, Advanced Design System (ADS) de l'entreprise Keysight, pour ce qui concerne la partie antenne de la carte.

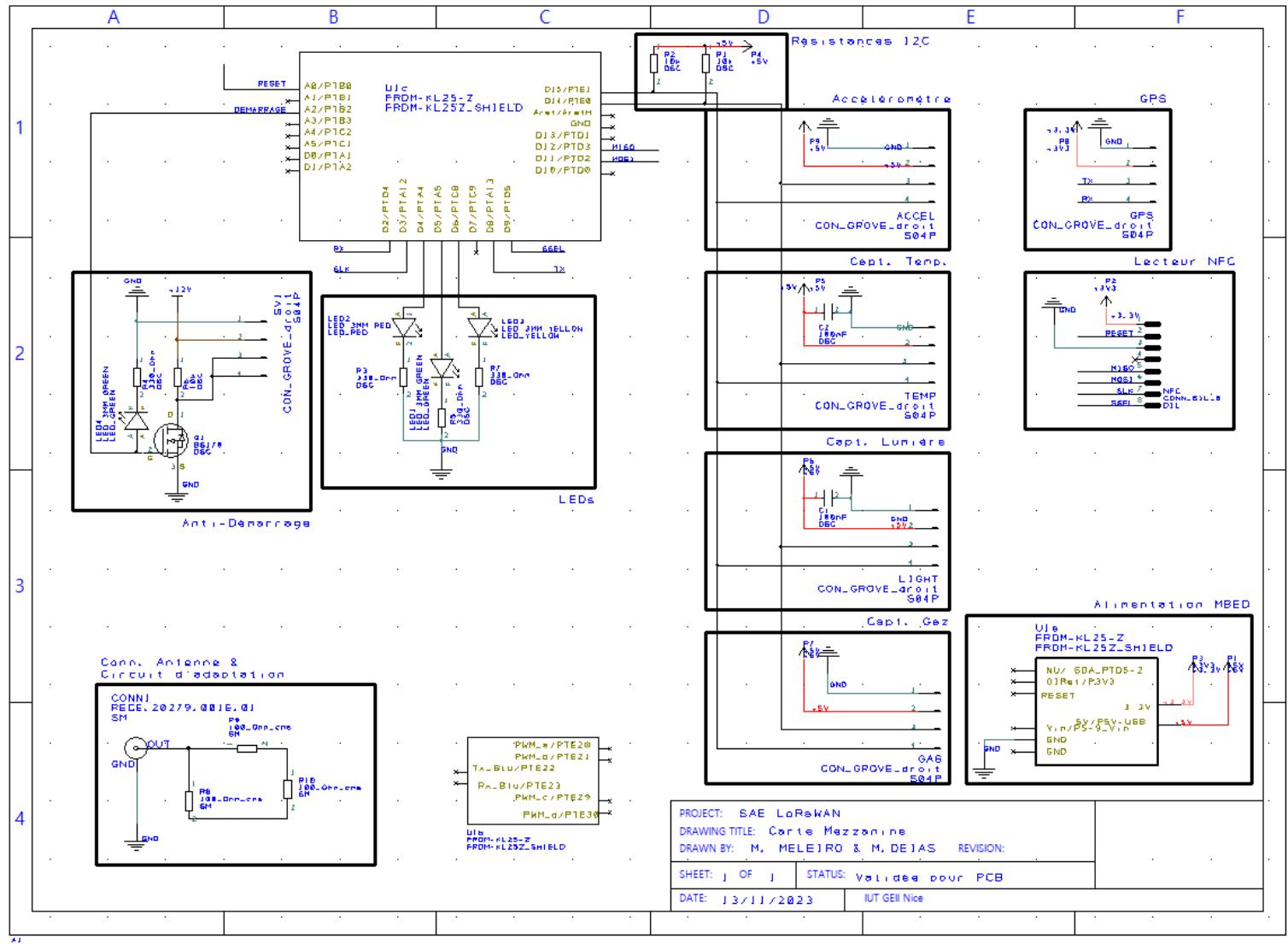


FIGURE 2 – Schematic

Le schematic est divisé en plusieurs parties.

Les parties capteurs, qui comportent les connecteurs Grove permettant de connecter chaque capteur correspondant à chaque mesure que la carte est capable de réaliser.

Les capteurs de Température, de Lumière (UV), de Gaz, et l'accéléromètre sont connectés en liaison I2C au microcontrôleur. Nous avons donc utilisé deux résistances I2C de $10k\Omega$, et connecté ces 4 capteurs en séries à ces mêmes résistances. Nous avons tenu à ce que ces 4 connecteurs Groves soient alignés sur le PCB afin que la carte soit cohérente au schematic, et donc visuellement plus compréhensible.

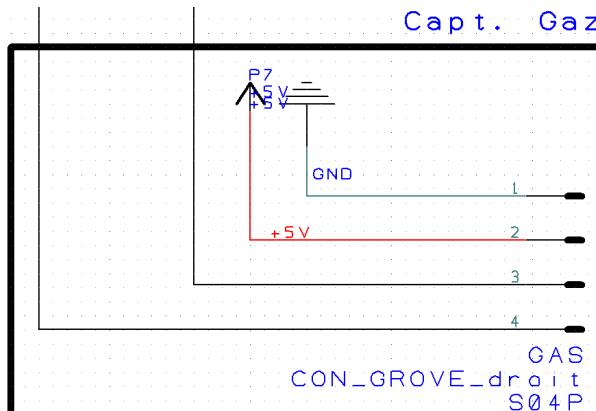


FIGURE 3 – Partie Gaz

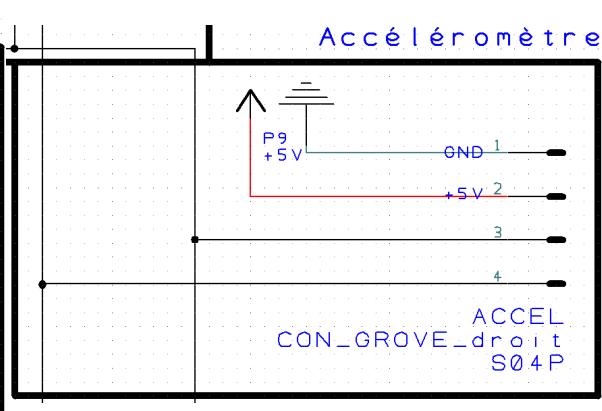


FIGURE 4 – Partie Accéléromètre

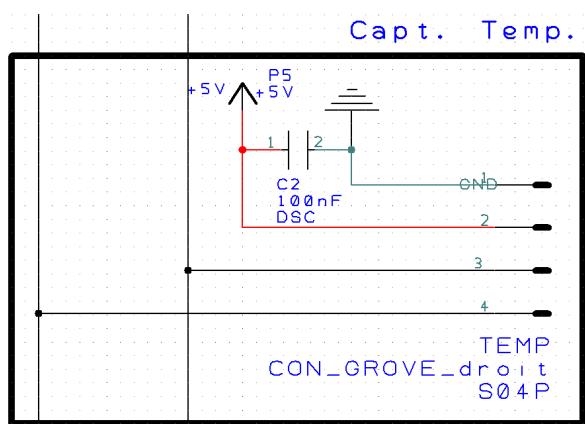


FIGURE 5 – Partie Température/Humidité

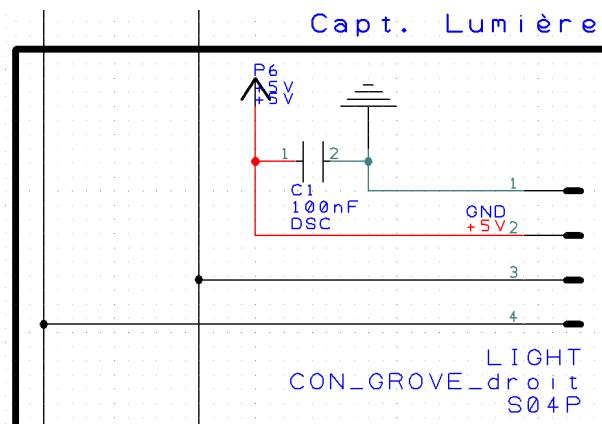


FIGURE 6 – Partie Lumière

Le capteur GPS, quant à lui, fonctionne en liaison UART. Nous avons donc connecté les pin 3 et 4 du connecteur Grove, respectivement à des pins TX et RX du microcontrôleur.

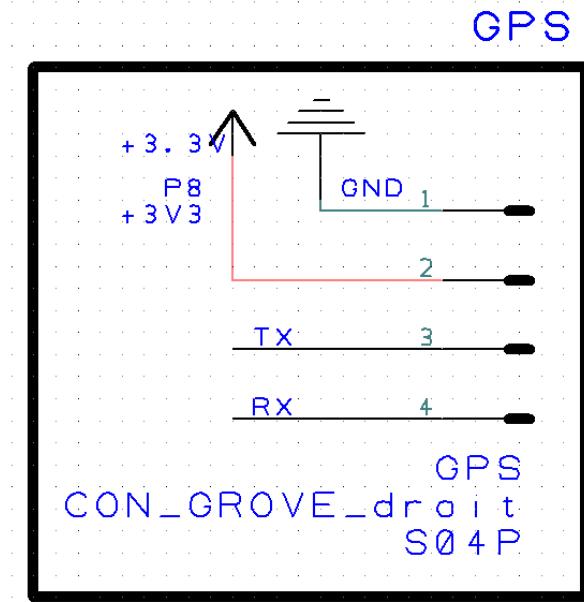


FIGURE 7 – Partie GPS

Enfin, le lecteur NFC est connecté en liaison SPI au microcontrôleur. Il a donc fallu lier les pins 5, 6, 7, et 8 respectivement au MISO, MOSI, SLK et SSEL. Cette fois, le connecteur utilisé est un connecteur SIL8.

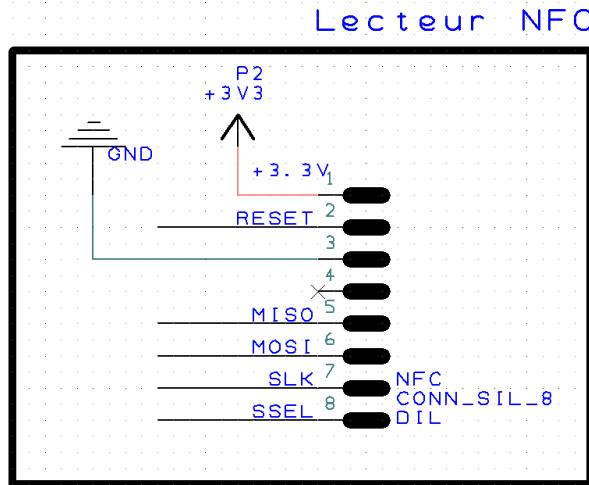


FIGURE 8 – Partie NFC

La partie alimentation, elle, comporte les sources de tensions 3V3 et 5V provenant du microcontrôleur.

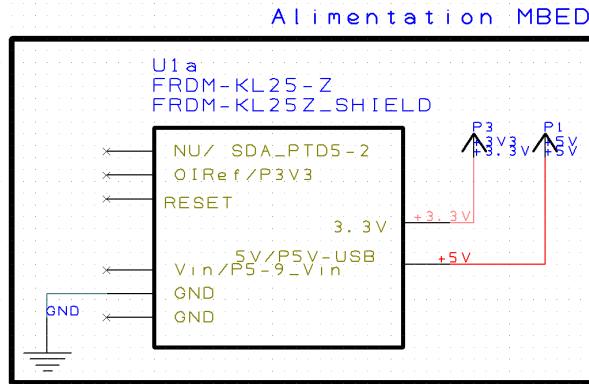


FIGURE 9 – Partie Alimentation

Aussi, la partie LEDs qui sert de IHM pour l'utilisateur. Trois LEDs sont présentes : une rouge, une orange et une verte.

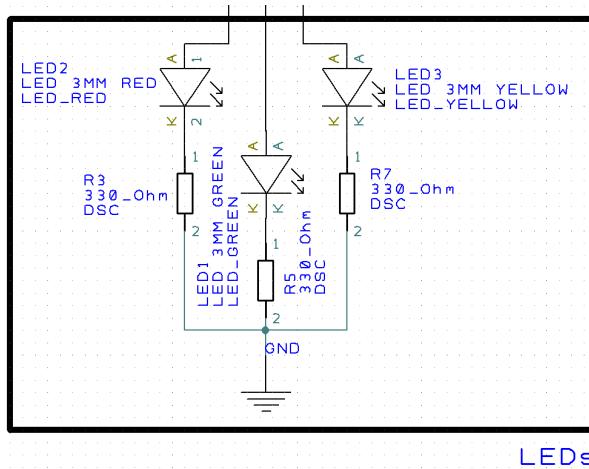


FIGURE 10 – Partie LEDs

La partie « Anti-démarrage » quant à elle permet au Kiffy de ne pas allumer le moteur tant que le badge n'est pas reconnu et autorisé à démarrer le Kiffy.

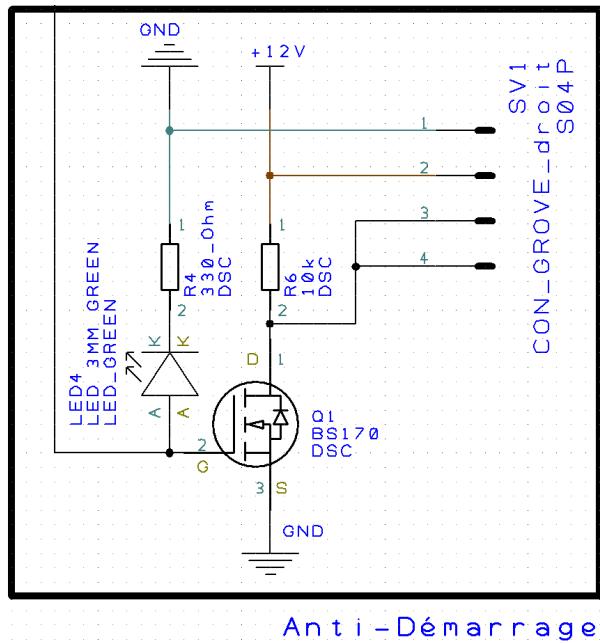


FIGURE 11 – Partie Anti-démarrage

Enfin, il y a la partie concernant les composants d'adaptation à l'antenne, constituée de composants CMS (voir Design de l'antenne).

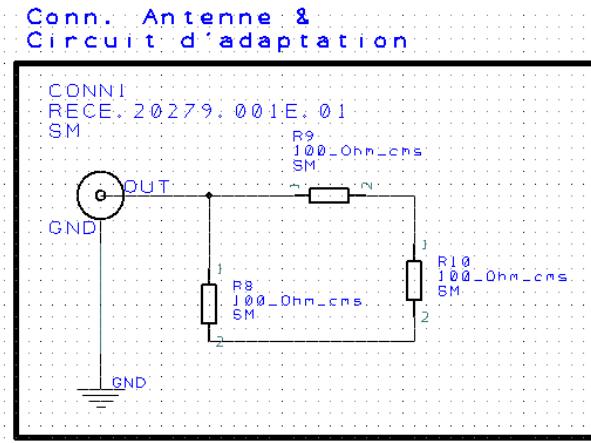


FIGURE 12 – Partie CMS

3.2 Design de l'antenne

3.2.1 Définition du substrat

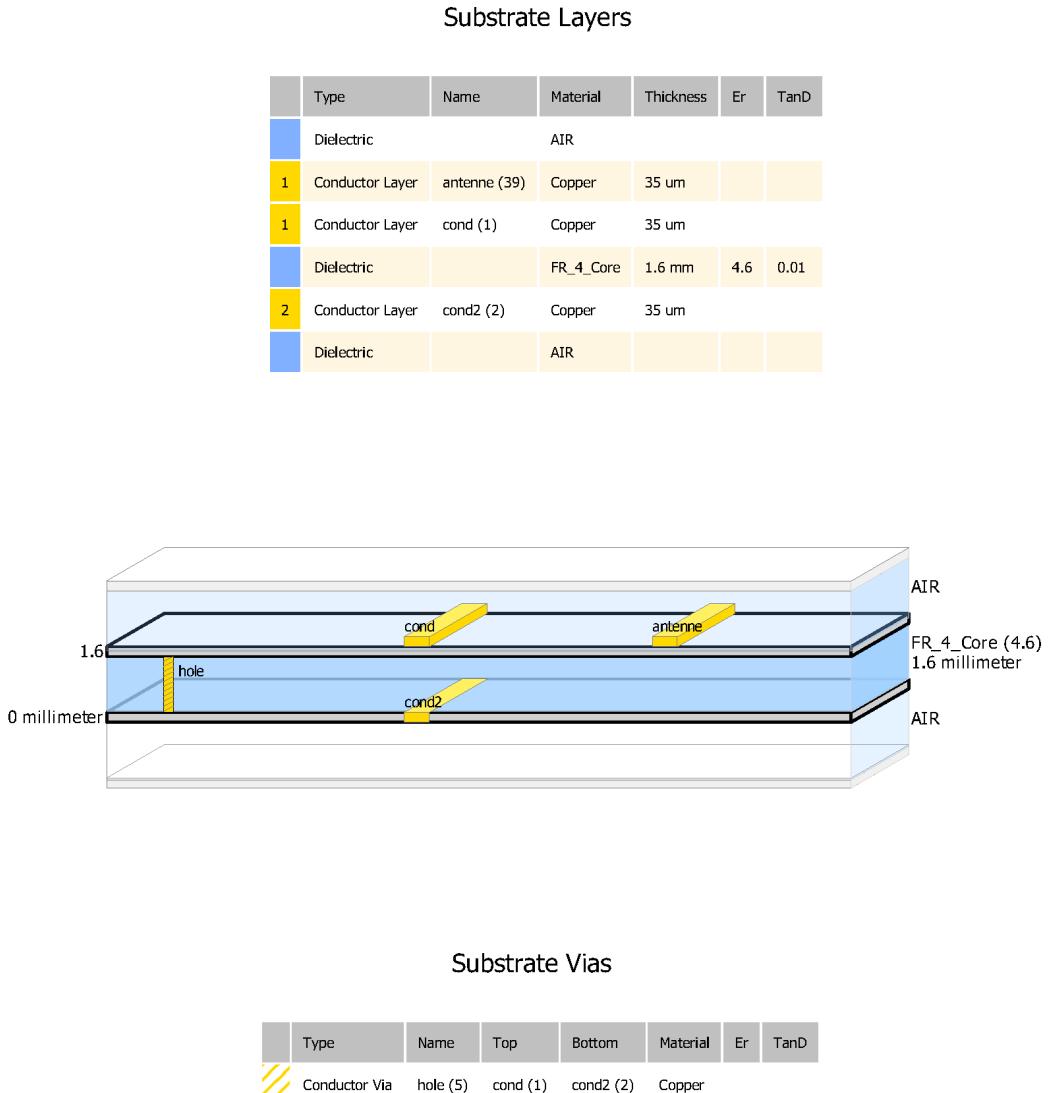


FIGURE 13 – Définition du substrat

La première capture d'écran montre les couches du substrat d'une antenne. Le substrat est la partie du circuit imprimé qui supporte les composants électroniques. Il est généralement constitué d'un matériau diélectrique, comme la céramique ou le polymère.

Sur la capture d'écran, on peut voir deux couches conductrices, une couche supérieure et une couche inférieure. La couche supérieure est l'antenne proprement dite. Elle est constituée d'un conducteur métallique, comme l'or ou l'argent. La couche inférieure est un conducteur de référence. Elle est également constituée d'un conducteur métallique, mais elle est reliée à la masse du circuit.

Entre les deux couches conductrices se trouve une couche de matériau diélectrique. Elle joue le rôle d'isolant entre les deux conducteurs.

Enfin, la couche inférieure est en contact avec une couche diélectrique. La couche diélectrique est un matériau qui ne conducte pas l'électricité. Elle a pour fonction d'empêcher les courants électriques de circuler entre les deux couches conductrices.

La deuxième capture d'écran montre les vias du substrat. Les vias sont des trous percés dans le substrat qui relient deux couches conductrices. Ils sont généralement utilisés pour relier des couches situées sur des côtés opposés du substrat.

Sur la capture d'écran, on peut voir un via qui relie la couche conductrice supérieure à la couche conductrice inférieure. Le via est constitué d'un trou percé dans le substrat, qui est ensuite rempli de métal.

Le métal utilisé pour remplir le via est généralement le même que le métal utilisé pour les couches conductrices. Cela permet d'assurer une bonne conductivité électrique.

3.2.2 Design de l'antenne

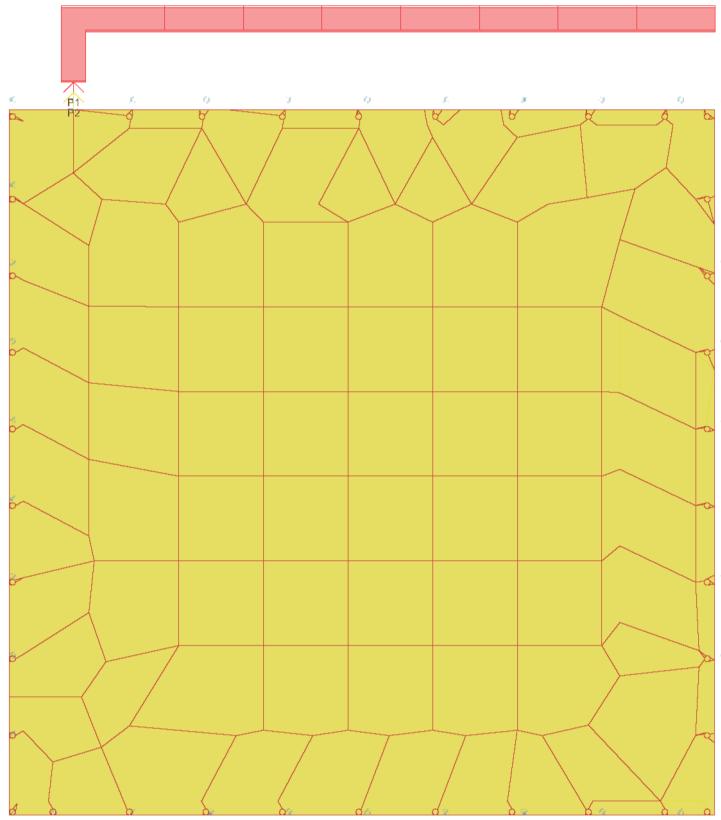


FIGURE 14 – *Design de l'antenne*

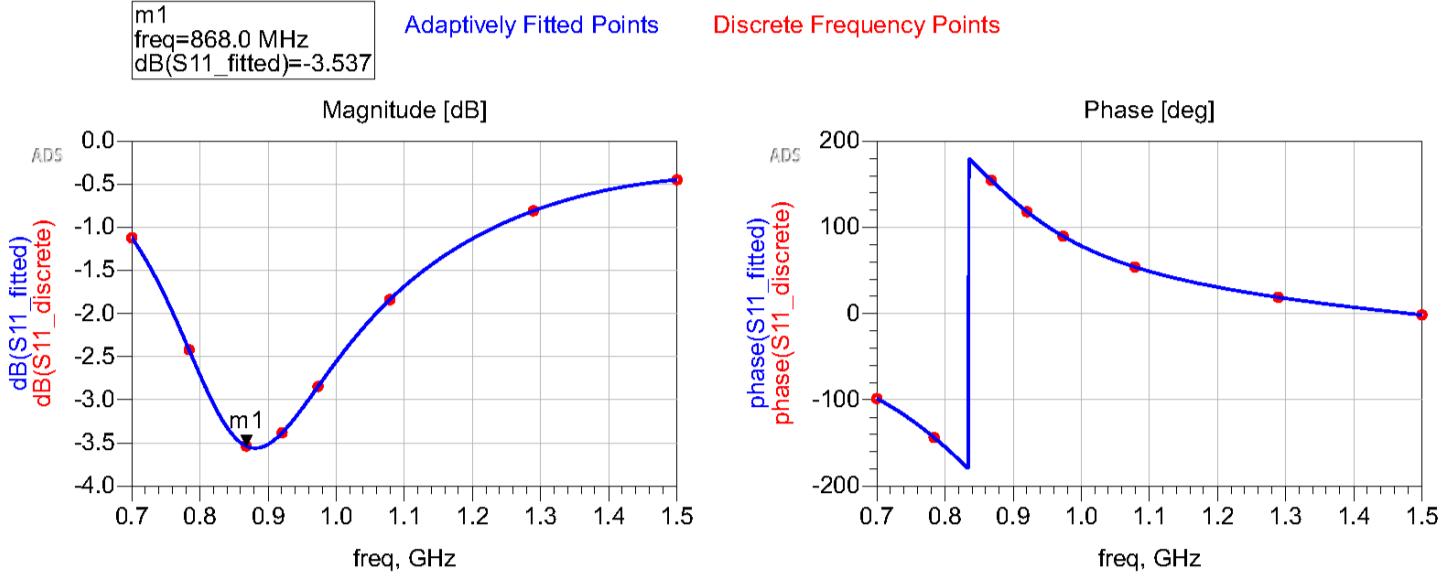
L'antenne est constituée de deux couches conductrices, une couche supérieure et une couche inférieure. La couche supérieure est l'antenne proprement dite, tandis que la couche inférieure est un conducteur de référence.

La couche supérieure est reliée à un connecteur, ce qui permet de connecter l'antenne à un autre circuit. La couche inférieure est reliée à la masse du circuit, ce qui permet d'assurer une bonne stabilité de l'antenne.

Les dimensions de l'antenne sont de 10 mm x 5 mm. La permittivité du matériau diélectrique est de 4.5.

3.2.3 Paramètres S

Discrete Frequencies vs. Fitted (AFS or Linear)



Dataset: antenneLORA_MomUW_a - Nov 14, 2023

FIGURE 15 – Paramètres S

Les paramètres S, ou coefficients de diffraction ou de répartition, sont des paramètres utilisés en hyperfréquences, en électricité ou en électronique pour décrire le comportement électrique de réseaux électriques linéaires en fonction des signaux d'entrée.

Sur l'axe horizontal, on retrouve la fréquence du signal RF. Sur l'axe vertical, on retrouve la magnitude et la phase du signal réfléchi par l'antenne. Les paramètres S sont représentés par deux courbes, une pour la magnitude et une pour la phase. La magnitude du signal réfléchi est représentée par la ligne bleue. La phase du signal réfléchi est représentée par la ligne rouge. L'antenne a une fréquence de résonance de 868 MHz. À cette fréquence, la magnitude du signal réfléchi est minimale, ce qui signifie que l'antenne est la plus efficace. La phase du signal réfléchi est également minimale à la fréquence de résonance. Cela signifie que l'antenne est la plus directive à cette fréquence.

L'antenne a une largeur de bande de 50 MHz. Cela signifie que l'antenne est efficace dans une plage de fréquences de 818 à 918 MHz.

3.2.4 Far Field

Dataset: emFar - Nov 14, 2023

Frequency	E_max	Theta_max	Phi_max	Directivity_max	Gain_max	RadiatedPower	InputPower	Efficiency	CutType	CutAngle
8.680E8	0.329	4.000	167.000	2.410	1.115	0.001	0.001	0.742	Phi	0.000

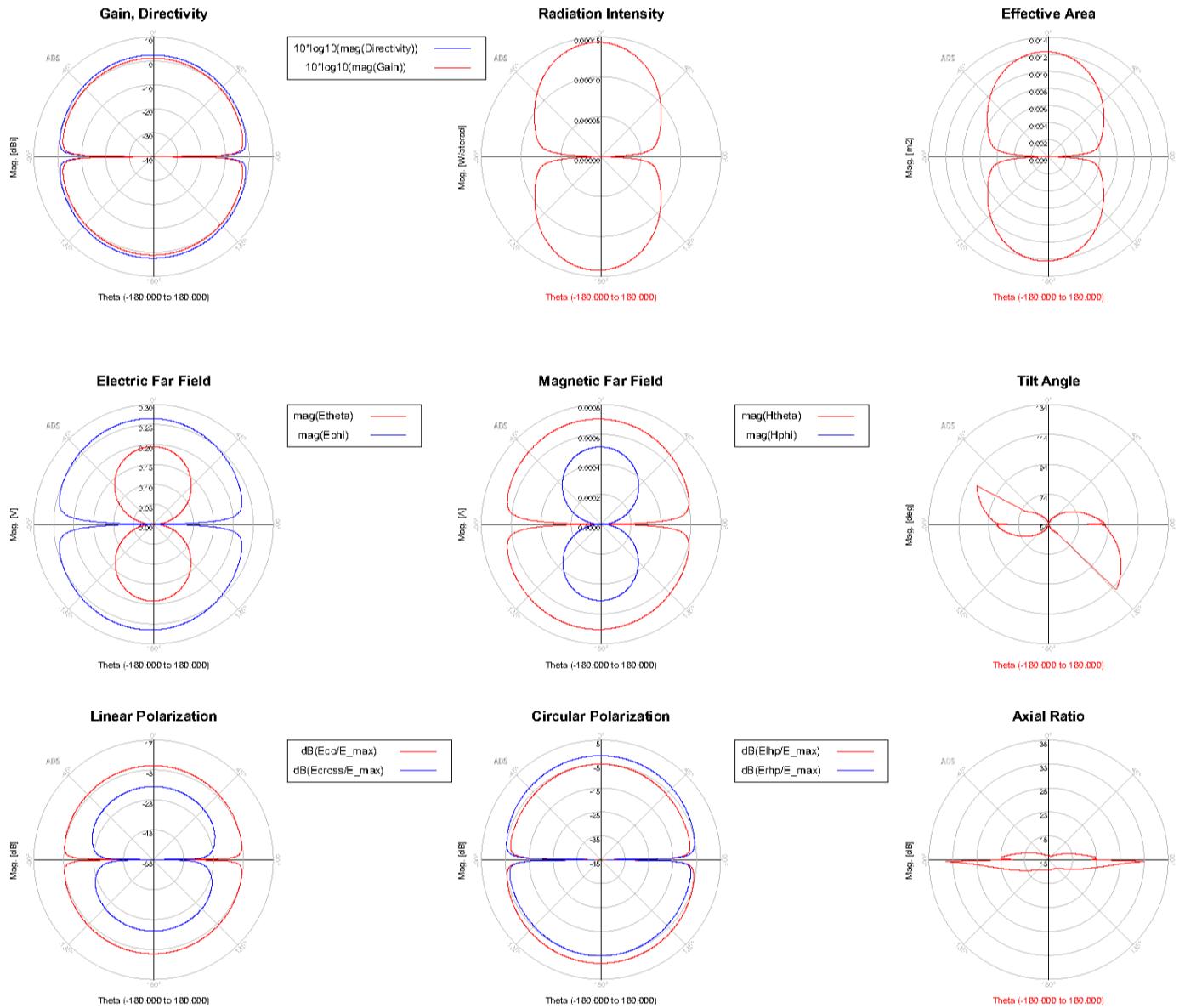


FIGURE 16 – Far Field

Le diagramme de rayonnement est une représentation de la distribution de la puissance rayonnée par une antenne dans l'espace. Sur l'axe horizontal, on retrouve la direction de rayonnement. Sur l'axe vertical, on retrouve le niveau de puissance rayonnée. Le diagramme de rayonnement est représenté par une série de courbes, une pour chaque direction de rayonnement. Les courbes sont tracées en noir. Les zones sombres indiquent que la puissance rayonnée est élevée. Les zones claires indiquent que la puissance rayonnée est faible.

L'antenne a un diagramme de rayonnement unidirectionnel. Cela signifie que la puissance est principalement rayonnée dans une seule direction. La direction de rayonnement la plus élevée est perpendiculaire à l'antenne. Cela signifie que l'antenne est la plus efficace pour la transmission de signaux dans cette direction.

L'antenne a un gain de 10 dBi. Cela signifie que l'antenne est capable de concentrer la puissance rayonnée dans une direction donnée par un facteur de 10.

3.2.5 Circuit d'adaptation avant rétrosimulation

Nous pouvons voir que sur l'abaque (issu de la simulation de fonctionnement de l'antenne non-adaptée), les réflexions d'impédance peuvent entraîner des pertes significatives de puissance et une inefficacité dans la transmission des signaux LoRa à 868 MHz.

Afin de remédier à cette situation, le circuit d'adaptation (une impédance série et un condensateur en parallèle) intervient en ajustant les paramètres électriques pour garantir une transition en douceur de l'antenne vers le système électronique. En optimisant l'impédance, le circuit d'adaptation permet d'exploiter pleinement le potentiel de l'antenne.

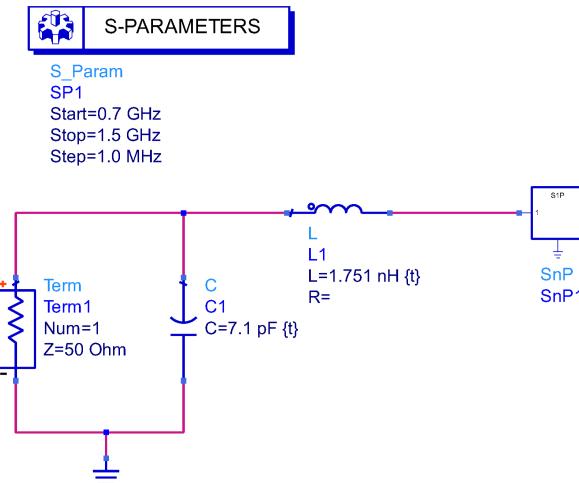


FIGURE 17 – Circuit d'adaptation avant rétrosimulation

3.2.6 Rétrosimulation

La rétrosimulation d'antenne est une technique de simulation par éléments finis (FEM) qui permet de calculer les performances d'une antenne à partir de son modèle géométrique et de ses propriétés physiques. Elle est utilisée pour concevoir et analyser des antennes pour une variété d'applications, notamment les communications sans fil, la navigation et la surveillance.

Dans ADS, la rétrosimulation d'antenne est effectuée à l'aide de l'outil Momentum. Momentum est un logiciel FEM puissant qui permet de simuler des structures 3D complexes comme on peut le voir sur les captures d'écran, issues de notre antenne, ci-dessous.

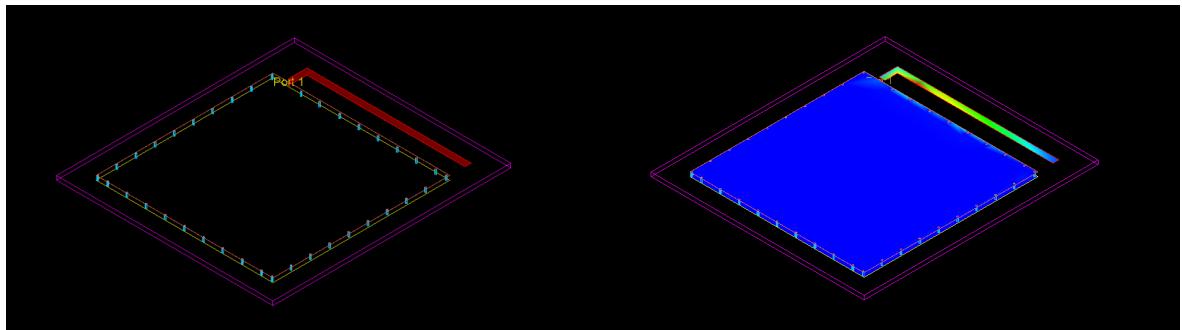


FIGURE 18 – (1)

FIGURE 19 – (2)

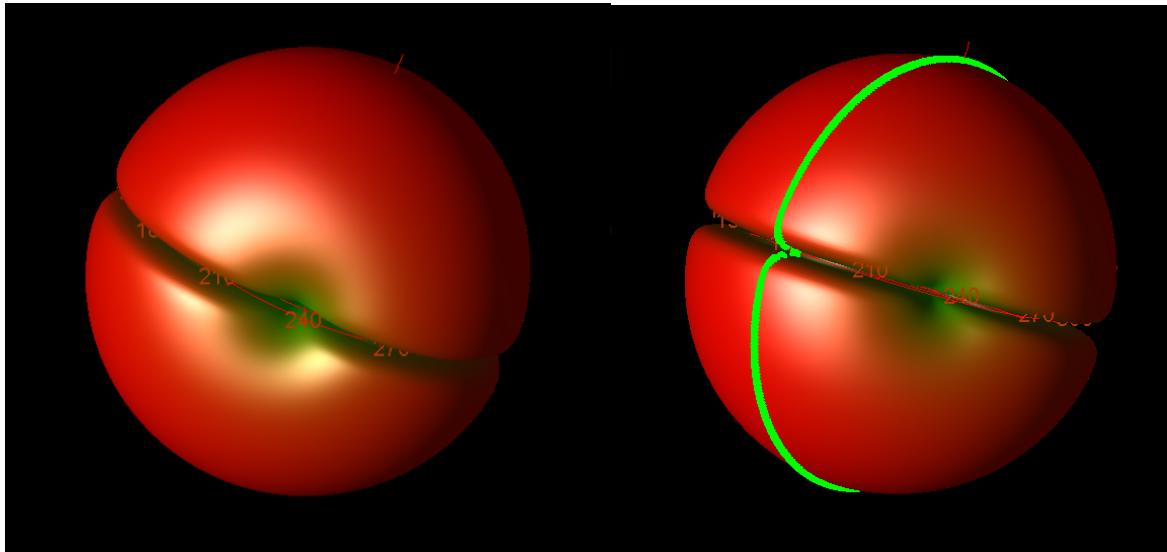


FIGURE 20 – (3)

FIGURE 21 – (4)

Le processus de rétrosimulation d'antenne dans ADS se déroule comme suit :

1. Création du modèle géométrique de l'antenne. Le modèle géométrique de l'antenne doit être créé avec précision pour que les résultats de la simulation soient fiables.
2. Définition des propriétés physiques de l'antenne. Les propriétés physiques de l'antenne, telles que la permittivité relative et la conductivité, doivent être définies pour que la simulation soit précise.
3. Exécution de la simulation. Une fois que le modèle géométrique et les propriétés physiques de l'antenne ont été définis, la simulation peut être exécutée.

Ici, on veut déplacer virtuellement le CON UFL jusqu'à l'endroit où l'on veut placer notre circuit d'adaptation. Dans notre cas , il faut d'abord un déplacement de -L1 (ligne de largeur 1.34mm), puis un déplacement de -L2 (ligne de largeur 2.44mm).

Nous avons donc calculé les valeurs de capacité et d'inductance à utiliser, ainsi que déterminé quels composants disposer en parallèle et en série.

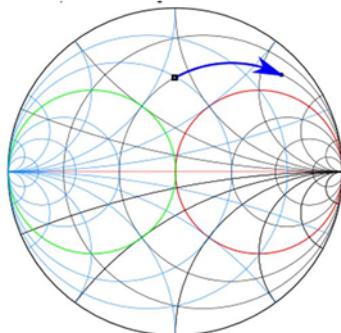


Fig.1 : Modification de l'impédance avec une inductance en série

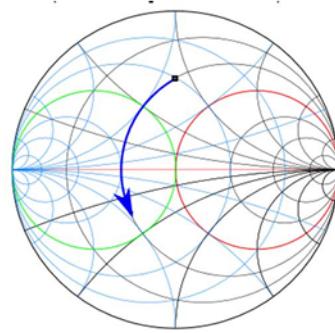


Fig.2 : Modification de l'impédance avec une capacité en série

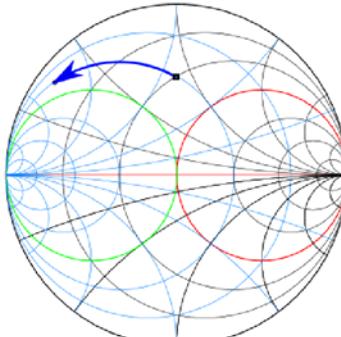


Fig.3 : Modification de l'impédance avec une inductance en parallèle

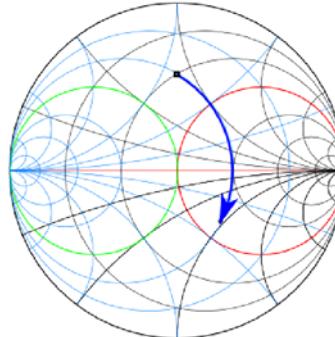


Fig.4 : Modification de l'impédance avec une capacité en parallèle

FIGURE 22 – Modifications sur l'abaque de Smith

3.2.7 Adaptation de l'antenne

Nous avons donc d'abord soudé une résistance de 0Ω afin de connecter la carte à l'antenne, et ainsi tester cette dernière pour connaître ses performances « naturelles ».

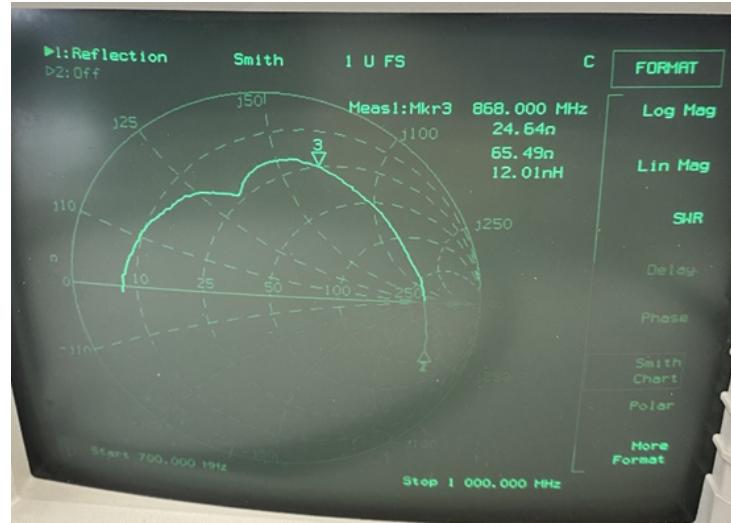


FIGURE 23 – Performances de l'antenne "Naturelle"

Compte tenu de l'abaque de Smith « naturel » de notre antenne, nous avons dû opter pour un circuit avec un condensateur(8.2pF) en parallèle, et une inductance(6.8nH) en série afin de centrer le point sur l'abaque.

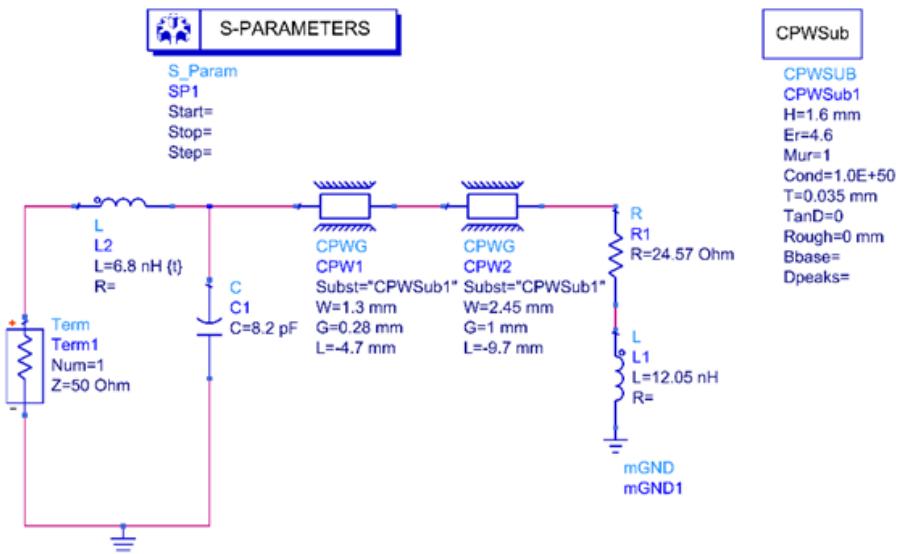


FIGURE 24 – Circuit d’adaptation réalisé sur ADS

Nous avons finalement obtenu un SNR (Signal to Noise Ratio) de -7.8 dB, ce qui est satisfaisant étant donné l’objectif idéal de -10dB.

La carte a donc été finalisée lors de l’avant-dernière séance, ce qui nous a permis de réaliser plusieurs tests, notamment avec les capteurs Température/Humidité, UV et GPS, qui se sont montrés concluants.

3.3 Réalisation du PCB

Nous avons par la suite pu valider la conception de notre schematic, pour le convertir en PCB, et pouvoir ainsi disposer les composants de manière à respecter le cahier de charges imposé.

Les consignes demandaient à ce qu'un minimum de vias soient utilisés, afin de ne pas polluer l'espace de la carte et la rendre plus « lisible ». Nous avons donc premièrement disposé les connecteurs Grove, alignés en bas de la carte, pour améliorer la cohérence et la facilité des branchements.

Le Connecteur SIL8 aussi à été placé dès le début, car très imposant et lié aux connecteurs Grove d'au-dessus, étant donné qu'il est utilisé pour brancher le capteur NFC. Le placement des vias tout autour de la carte afin de connecter les 2 plans de masse a été une des premières choses réalisées sur le PCB, car la distance entre chaque via se devait d'être précise et respectée.

La réalisation du PCB a été une étape assez complexe du projet, car il s'est avéré difficile de respecter toutes les contraintes, tout en ayant une carte fonctionnelle. L'étape la plus cruciale fut celle de la mise en place de l'antenne. En effet, les pistes devaient respecter des tailles très précises, correspondant à la rétrosimulation, afin de pouvoir avoir un signal convenable.

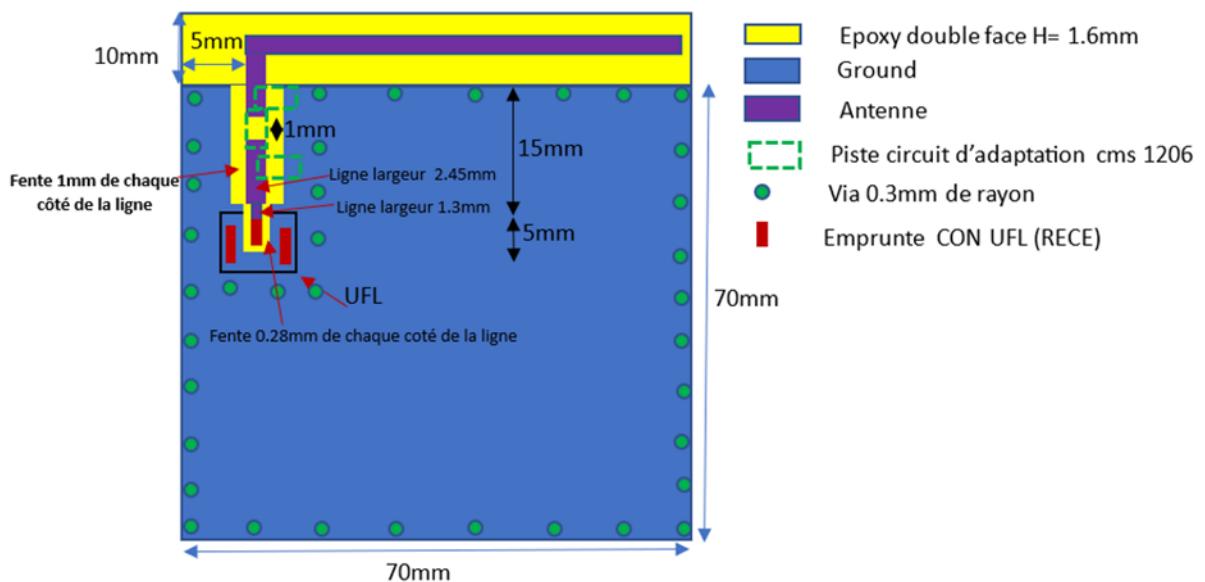


FIGURE 25 – Contraintes Géométriques de l'antenne

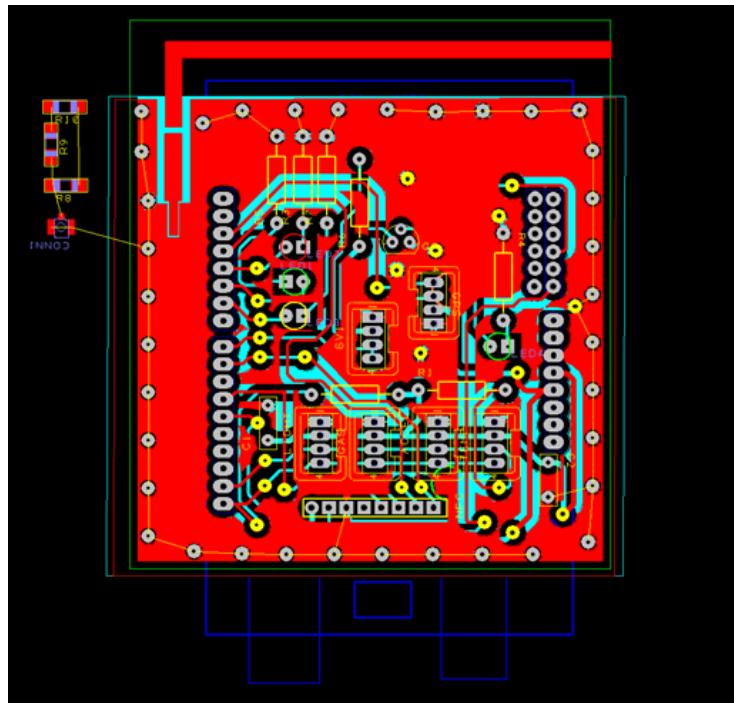


FIGURE 26 – *PCB avec les plans de masse*

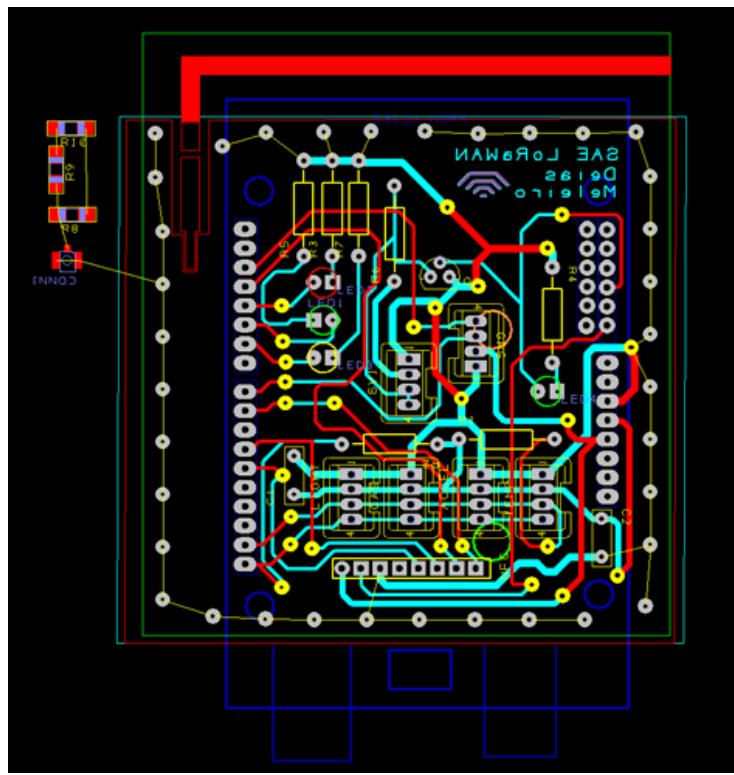


FIGURE 27 – *PCB sans les plans de masse*

3.4 Soudure

Le PCB de la carte a donc été réalisé sur DesingSpark et son millar (autrement dis « l'ombre » de son cuivre, donc les pistes, les pads et les plans de masses) est imprimé sur une feuille (avec un effet miroir). Ce millar va servir pour faire de l'ombre aux UV émis par l'insoleuse afin que sous l'encre du papier, le cuivre pré sensibilisé aux UV ne perde pas sa protection. A l'inverse, les zones sans encre, et donc sans cuivre sur le PCB, seront attaquées par les UV. Une fois les UV appliqués, il ne reste plus qu'à faire tremper le cuivre dans 2 solutions différentes pour réaliser la gravure. Notre avance nous a permis de commencer la soudure relativement tôt. La première étape fut donc de percer la carte afin de faire tous les trous nécessaires à la soudure des composants. Néanmoins, le bottom copper était « décalé » par rapport au top copper.

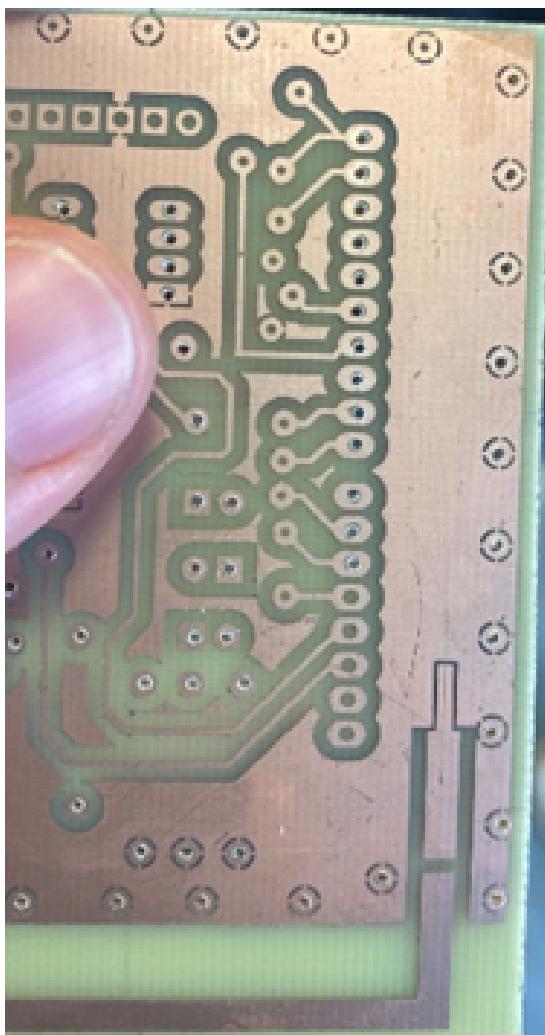


FIGURE 28 – *TOP copper*

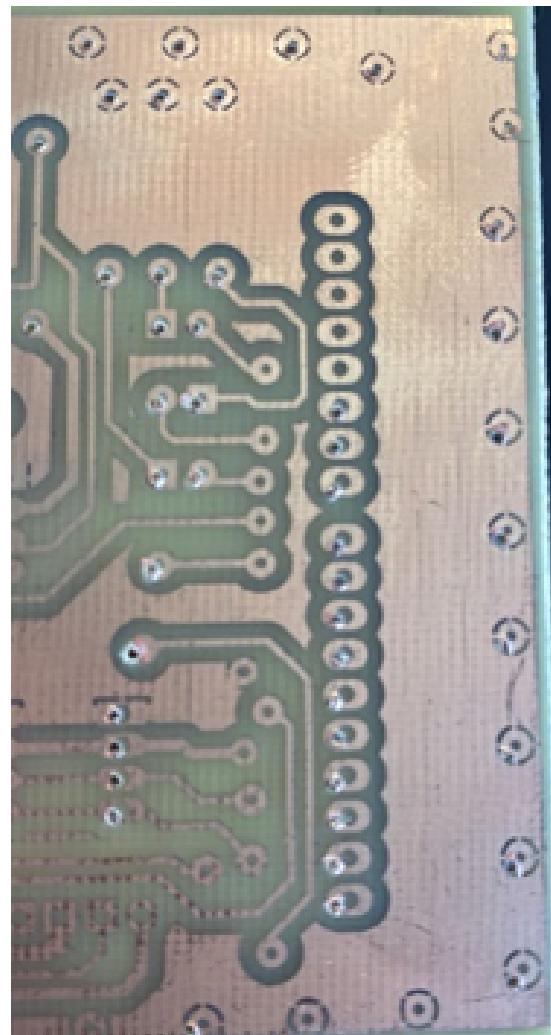


FIGURE 29 – *BOTTOM copper*

La soudure des composants CMS, afin d'adapter l'antenne de la carte, a nécessité une technique particulière. Dans ce cas précis, nous n'avions jamais réalisé de soudure manuelle de composants CMS. Nous avons donc dû apprendre sur le tas, ce qui a pris du temps et a nécessité de nombreux essais. La soudure manuelle de composants CMS est une tâche difficile, car les composants sont très petits et fragiles. Il faut une grande précision pour placer les composants correctement et pour réaliser une soudure solide.

En suite, les vias, qui relient les différentes couches du circuit imprimé, n'étaient pas compatibles avec une soudure au four. Il a donc fallu aussi les souder manuellement.

Les vias sont des trous percés dans le circuit imprimé, qui sont ensuite remplis de métal. Dans le cas présent, les vias ont été réalisés avec des « fils », car nous n'avions pas encore reçu la machine permettant de les faire sans soudure.

Si la soudure des composants CMS avait été réalisée au four, la chaleur aurait fait fondre le métal des vias, ce qui aurait rendu la carte inutilisable.

Au final, nous avons réussi à souder les composants CMS et les vias correctement, mais cela a été une tâche ardue. L'ajout de composants CMS était nécessaire à l'adaptation de l'antenne de la carte. En effet, l'objectif était d'obtenir un SNR (Signal to Noise Ratio) de -10dB. Pour cela, il fallait ajouter une inductance et un condensateur, en série parallèle ou parallèle série selon la valeur de l'impédance et d'admittance du circuit sur un abaque de Smith.

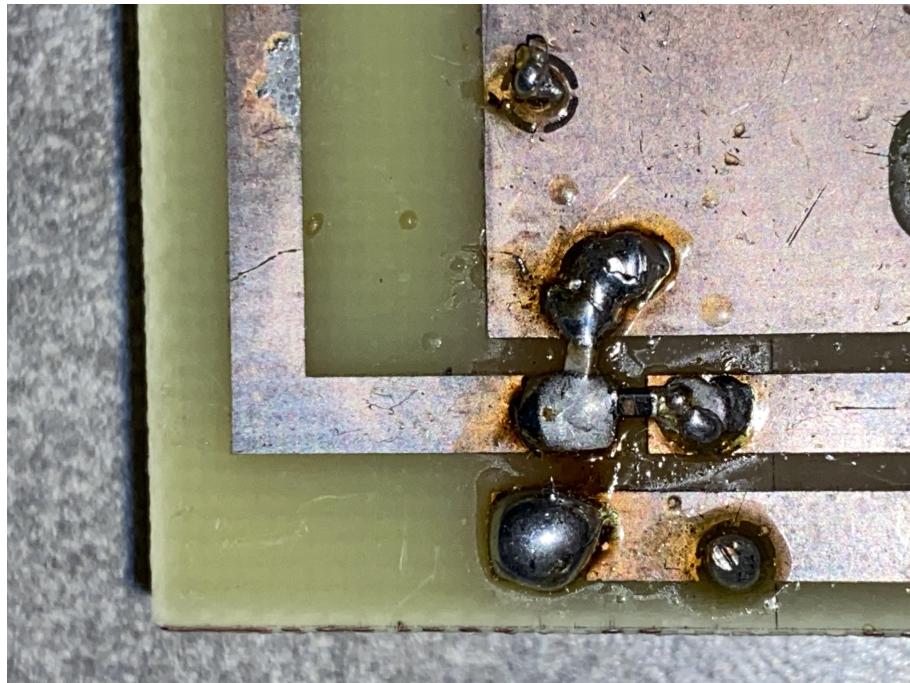


FIGURE 30 – Soudure des Composants CMS

4 LoRaWAN



FIGURE 31 – Logo LoRaWAN

4.1 Le protocole

LoRaWAN est un protocole de communication radio qui définit comment des équipements terminaux communiquent sans fil au travers de passerelles, constituant ainsi un réseau étendu à basse consommation (LPWAN). En Europe, il utilise la bande radio libre 868 MHz. C'est un média radio réglementé : seulement 1% du temps autorisé en uplink sur une durée de 1H, soit 36s de transmission uplink par heure seulement. De plus, la transmission a une limite de 140 messages par jour d'une charge utile de 12 octets chacun.

4.1.1 Architecture

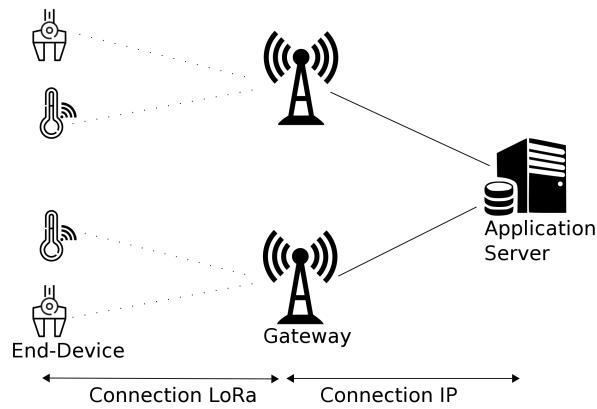


FIGURE 32 – Architecture LoRaWAN

Le réseau LoRaWAN est composé de trois types d'entités :

- Les nœuds terminaux (End-Devices) sont les équipements IoT qui communiquent avec le réseau. Ils sont alimentés par batterie et ont une faible consommation énergétique.
- Les passerelles (Gateways) sont des points d'accès au réseau. Elles reçoivent les messages des nœuds terminaux et les transmettent au serveur LoRaWAN.
- Le serveur LoRaWAN est le centre de contrôle du réseau. Il reçoit les messages des passerelles et les diffuse aux applications.

Dans notre cas, à l'I.U.T., le noeud terminal sera notre carte mBED B-L072Z-LRWAN1 ; la passerelle sera celle installée dans la salle des professeurs du département GEII de l'I.U.T. ; et enfin le serveur LoRaWAN sera celui hébergé par The Things Network.

4.1.2 Fonctionnement

Le protocole LoRaWAN utilise une technique de modulation par étalement de spectre appelée LoRa. Cette technique permet de transmettre des données sur de longues distances avec une faible consommation énergétique.

Les nœuds terminaux communiquent avec le réseau en utilisant des trames LoRa. Chaque trame LoRa contient les informations suivantes :

- L'adresse du nœud terminal
- Le type de message
- Les données du message

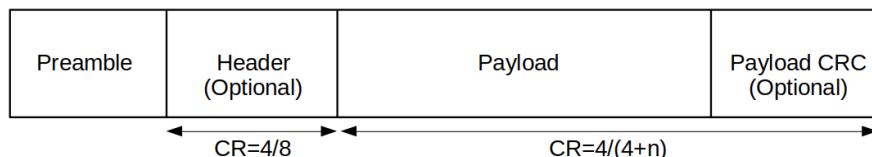


FIGURE 33 – Trame LoRa

Les passerelles reçoivent les trames LoRa des nœuds terminaux et les transmettent au serveur LoRaWAN. Le serveur LoRaWAN reçoit les trames des passerelles et les diffuse aux applications.

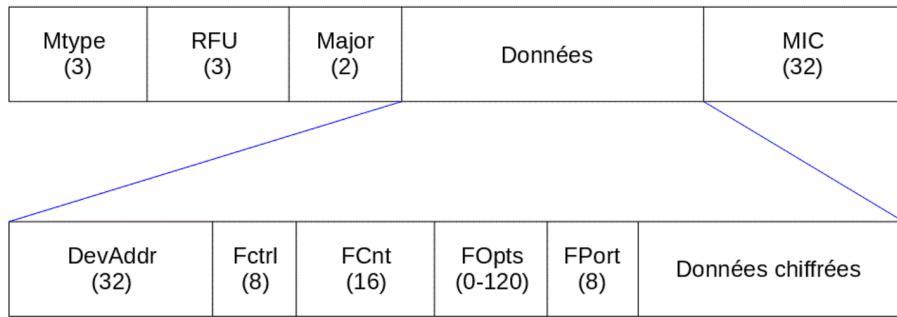


FIGURE 34 – *Trame LoRaWAN*

4.1.3 Caractéristiques

Les principales caractéristiques du protocole LoRaWAN sont les suivantes :

- Portée étendue : Le protocole LoRaWAN permet une portée allant jusqu'à 15 km en zone rurale et jusqu'à 5 km en zone urbaine.*
- Faible consommation énergétique : Les noeuds terminaux LoRaWAN sont alimentés par batterie et ont une durée de vie pouvant aller jusqu'à 10 ans.
- Bidirectionnalité : Le protocole LoRaWAN permet des communications bidirectionnelles entre les nœuds terminaux et le serveur LoRaWAN.
- Sécurité : Le protocole LoRaWAN intègre des mécanismes de sécurité pour protéger les données transitant sur le réseau.

*Il a été démontré qu'il est possible d'atteindre une portée de 1336 km

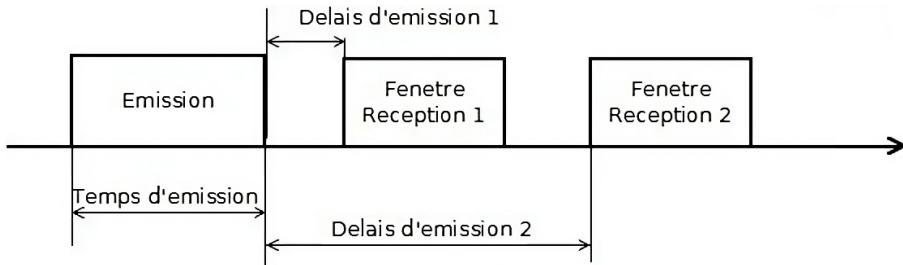


FIGURE 35 – *Uplink / Downlink*

4.2 Les capteurs

À travers le réseau LoRaWAN, nous distribuerons des données issues de 5 capteurs.

4.2.1 Température et humidité

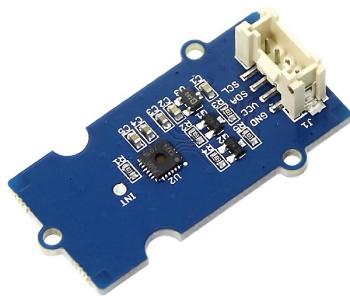


FIGURE 36 – Capteur de température et d'humidité

Il s'agit d'un capteur multifonctions qui nous donne en même temps des informations sur la température et l'humidité relative. Il utilise un capteur TH02 qui peut répondre aux besoins de mesure à des fins générales. Il fournit des lectures fiables lorsque l'humidité ambiante est comprise entre 0 et 80% d'humidité relative et que la température est comprise entre 0 et 70 °C, couvrant les besoins de la plupart des applications domestiques et quotidiennes qui ne contiennent pas de conditions extrêmes.

Selon la documentation, le capteur fonctionne sur une large plage de tension de fonctionnement (3,3 VDC à 5 VDC). Consomme très peu : à peu près 350 µA pendant la mesure. Détient une précision de l'ordre de : $\pm 4,5\%$ pour l'humidité relative et $\pm 0,5^{\circ}\text{C}$ pour la température. Et enfin transmet ses données sur un bus I2C.

4.2.2 Luminosité



FIGURE 37 – Capteur de luminosité

Il s'agit d'un capteur de lumière numérique multicanal, capable de détecter la lumière UV, la lumière visible et la lumière infrarouge. Cet appareil est basé sur SI1151, un capteur de SiLabs. Le Si1151 est un capteur de proximité infrarouge, d'indice UV et de lumière ambiante à faible consommation, basé sur la réflectance, avec interface numérique I2C et sortie d'interruption d'événement programmable. Cet appareil offre d'excellentes performances sous une large plage dynamique et une variété de sources lumineuses, y compris la lumière directe du soleil.

Selon la documentation, le capteur fonctionne sur une plage de tension de fonctionnement entre 3,3 VDC à 5 VDC et sur une plage de température de -45 à 85°C. Consomme à peu près 3.5 mA pendant la mesure. Mesure sur une large plage de longueur d'onde (entre 280 et 950 nm). Et enfin transmet ses données sur un bus I2C.

4.2.3 Gaz

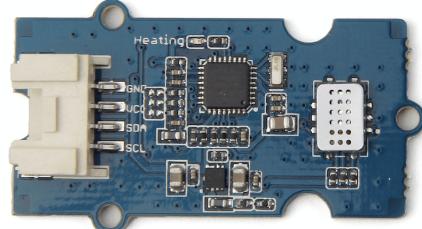


FIGURE 38 – Capteur de gaz

Il s'agit capteur de gaz multicanal de détection d'environnement avec un MiCS-6814 intégré qui peut détecter de nombreux gaz malsains, et trois gaz peuvent être mesurés simultanément grâce à sa fonction multicanal. Il peut donc nous aider à surveiller la concentration de plus d'un gaz. Dans le cadre de la SAE LoRaWAN, le gaz surveillé sera le NO₂ (dioxyde d'azote) qui est particulièrement présent dans les villes dû aux émissions de gaz des moteurs thermiques à combustion.

Selon la documentation, le capteur fonctionne sur une plage de tension de fonctionnement entre 3,3 VDC à 5 VDC. Il permet de mesurer trois gaz différents simultanément sur ces plages indiquées ci-dessous :

- Monoxyde de carbone CO 1 – 1000 ppm
- Dioxyde d'azote NO₂ 0,05 – 10 ppm
- Éthanol C₂H₆OH 10 – 500 ppm
- Hydrogène H₂ 1 – 1000 ppm
- Ammoniac NH₃ 1 – 500 ppm
- Méthane CH₄ >1000ppm
- Propane C₃H₈ >1000ppm
- Isobutane C₄H₁₀ >1000ppm

Afin d'atteindre une précision maximale, le capteur possède un élément chauffant (avant les mesures, il est conseillé d'attendre 10 minutes). Et enfin transmet ses données sur un bus I₂C. (ppm = parties par millions)

4.2.4 Localisation GPS



FIGURE 39 – Antenne et module GPS

Il s'agit d'un module Plug and Play possédant un boîtier SIM28, d'une antenne céramique et d'une configuration de communication série UART. Il dispose d'un récepteur GPS de 22 canaux de suivi/66 canaux d'acquisition. La sensibilité du suivi et de l'acquisition atteint toutes deux jusqu'à -160 dBm, ce qui en fait un excellent choix pour les projets éducatifs comme pour la SAE LoRa. (module pas cher et facile d'utilisation)

4.2.5 Identification RFID

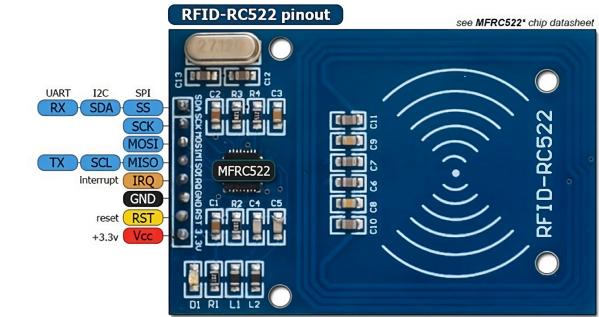


FIGURE 40 – Lecteur RFID

Le module RFID RC522 est un dispositif de communication sans contact qui permet de lire et d'écrire des données sur des cartes RFID. Il fonctionne à une fréquence de 13,56 MHz et prend en charge la communication avec les cartes MIFARE et ISO 14443A.

Le module est alimenté par une tension de 2,5 V à 3,3 V et consomme un courant de 13 à 26 mA en fonctionnement. Il dispose d'une interface SPI, d'une interface I2C et d'une interface UART série RS232, lors de la SAE LoRaWAN, nous utiliserons l'interface série SPI. La distance de fonctionnement typique en mode Lecture/Écriture est de 50 mm.

5 Programmation

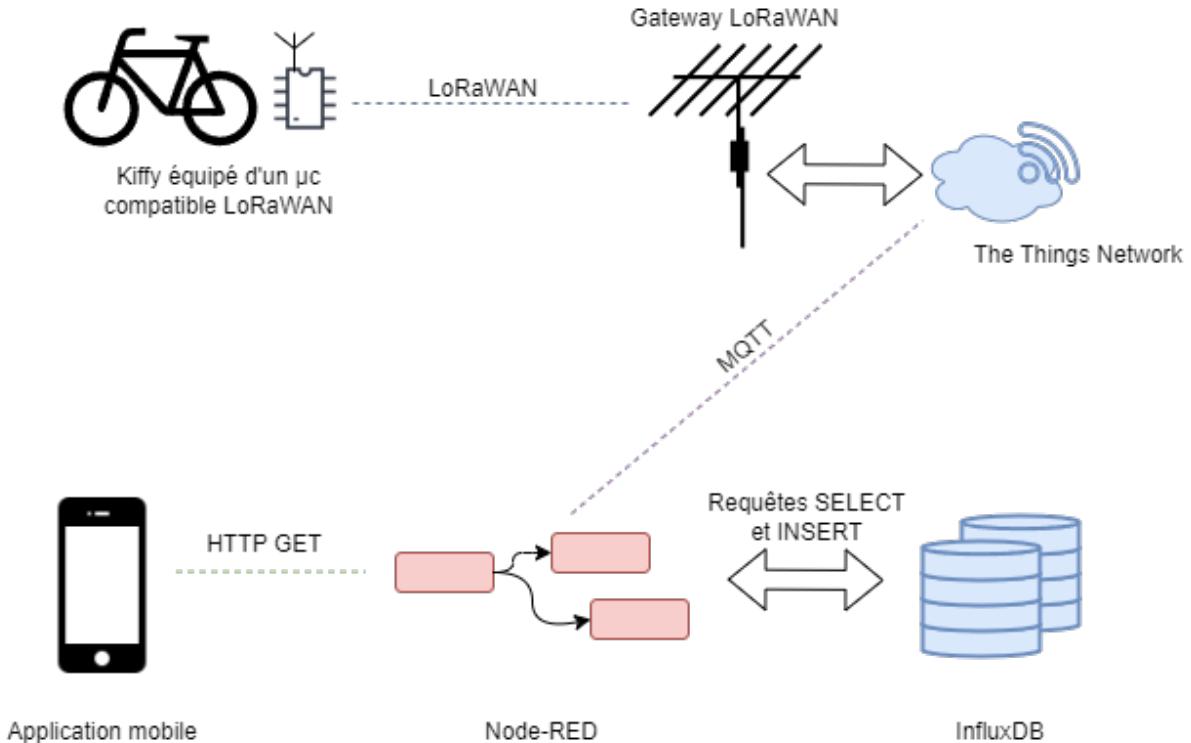


FIGURE 41 – La partie Programmation de la SAE LoRaWAN

5.1 Microcontrôleur

5.1.1 mBED

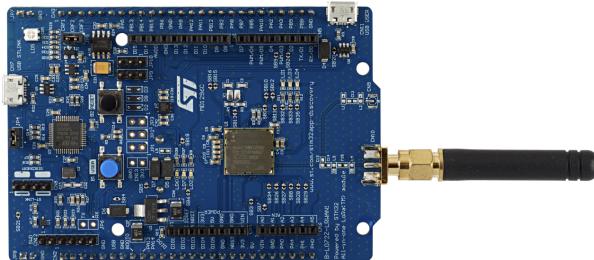


FIGURE 42 – Plateforme de développement B-L072Z-LRWAN1

Le microcontrôleur est le cerveau de ce projet. En effet, c'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

Dans notre projet, nous utilisons la plateforme de développement B-L072Z-LRWAN1 imposée, conçue par l'entreprise STMicroelectronics. Elle est équipée d'un micro-contrôleur Arm Cortex-M0+ de 48 MHz, 192 KB de mémoire flash (ROM), 20 KB de mémoire vive (RAM) et 20 KB de EEPROM. Elle est équipée d'un module LoRa/Sigfox fonctionnant entre 860 MHz - 930 MHz.

En utilisant le logiciel de développement intégré (IDE) Keil Arm Studio Cloud, nous avons pu développer notre code du contrôle du robot en langage C++. À la compilation, l'IDE Keil Studio fournit un fichier .bin qui est à placer à la racine du système de fichiers de la MBED.

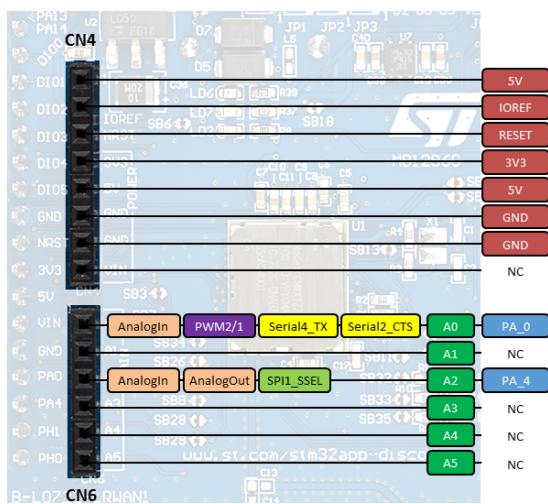


FIGURE 43 – Pinout mBED Gauche

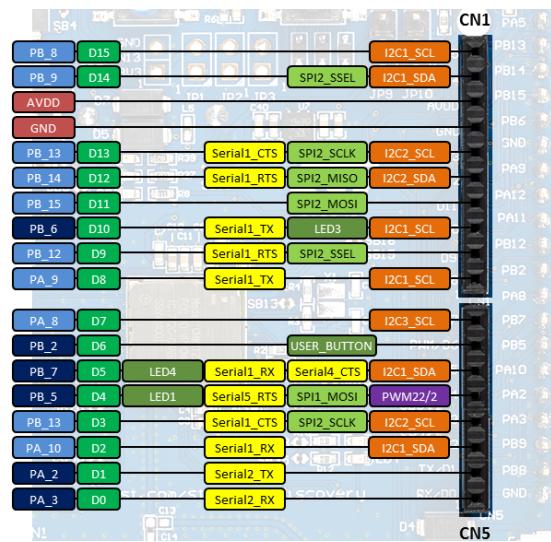


FIGURE 44 – Pinout mBED Droite

5.1.2 Configuration

Afin de pouvoir envoyer des données sur le réseau LoRaWAN et d'ainsi suivre le cahier des charges imposé, nous devons configurer et programmer notre microcontrôleur.

Comme évoqué précédemment, nous utilisons The Things Network pour la partie serveur. Mais avant que les données soient reçues, il faut que notre appareil soit reconnu et identifiable. Il existe deux méthodes permettant à notre mBED de s'identifier sur le réseau.

Activation par l'air (OTAA)

L'activation par l'air (OTAA) est la méthode d'activation la plus sécurisée pour les appareils LoRaWAN. Elle permet aux appareils de s'enregistrer auprès d'un réseau LoRaWAN sans avoir à être physiquement connectés à un ordinateur.

La procédure d'activation OTAA se déroule en deux étapes :

1. Demande d'association (Join-Request) : L'appareil émet un message de demande d'association sur un canal de join. Ce message contient les informations suivantes :
 - L'identifiant unique de l'appareil (DevEUI)
 - La clé de l'application (AppKey) (Clé secrète qui est unique pour chaque appareil. Elle est utilisée pour chiffrer les messages entre l'appareil et le serveur réseau)
 - Un nonce aléatoire (DevNonce)
2. Réponse d'association (Join-Accept) : Le serveur réseau répond à la demande d'association en envoyant un message de réponse d'association. Ce message contient les informations suivantes :
 - L'adresse de l'appareil (DevAddr) (Adresse unique qui est attribuée à l'appareil par le serveur réseau. Elle est utilisée pour identifier l'appareil sur le réseau)
 - Les clés de session (NwkSKey, AppSKey) (Clés secrètes qui sont utilisées pour chiffrer les messages entre l'appareil et le réseau)

Avantages de l'activation par l'air

- Sécurité renforcée : L'utilisation d'une clé d'application unique rend plus difficile pour les attaquants d'usurper l'identité ou d'écouter les appareils.
- Robustesse : Si la demande d'association est rejetée, l'appareil peut réessayer automatiquement la procédure d'association.

Activation par personnalisation (ABP)

L'activation par personnalisation (ABP) est une méthode d'activation moins sécurisée pour les appareils LoRaWAN. C'est celle que nous utiliserons lors de la SAE LoRaWAN. Elle permet aux appareils de s'enregistrer auprès d'un réseau LoRaWAN en partageant les informations suivantes avec le serveur réseau :

- L'identifiant unique de l'appareil (DevEUI)
- L'adresse de l'appareil (DevAddr)
- Les clés de session (NwkSKey, AppSKey)

Ces informations sont définies dans le fichier de paramétrage ABP dans le projet Keil Studio comme le montre la capture d'écran ci-dessous :

```

21     "lora-rf-switch-ctl2": { "value": "NC" },
22     "lora-txctl": { "value": "NC" },
23     "lora-rxctl": { "value": "NC" },
24     "lora-ant-switch": { "value": "NC" },
25     "lora-pwr-amp-ctl": { "value": "NC" },
26     "lora-tcxo": { "value": "NC" }
27   },
28   "target_overrides": {
29     "*": {
30       "mbed-trace.enable": false,
31       "platform.stdio-convert-newlines": true,
32       "platform.stdio-baud-rate": 115200,
33       "platform.default-serial-baud-rate": 115200,
34       "lora.over-the-air-activation": false,
35       "lora.duty-cycle-on": true,
36       "lora.phy": "EU868",
37
38       "lora.device-eui": "0x70, 0xB3, 0xD5, 0x7E, 0x00, 0x06, 0x2D, 0x00",
39       "lora.application-key": "0x70, 0xB3, 0xD5, 0x7E, 0x00, 0x02, 0x16, 0x8E",
40       "lora.appkey": "0x5E, 0xE6, 0x65, 0xC0, 0xBD, 0x28, 0x4D, 0x6F, 0xBE, 0x5A, 0x58, 0x28, 0xFF, 0xB3, 0xD9",
41       "lora.nwkskey": "0x14, 0x95, 0x00, 0x15, 0xE3",
42       "lora.device-address": "0x260B6576"
43     },
44
45     "K64F": {
46       "lora-spi-mosi": "D11",
47       "lora-spi-miso": "D12",
48     }
49   }
50 }

```

FIGURE 45 – Fichier de paramétrage ABP

5.1.3 Programme

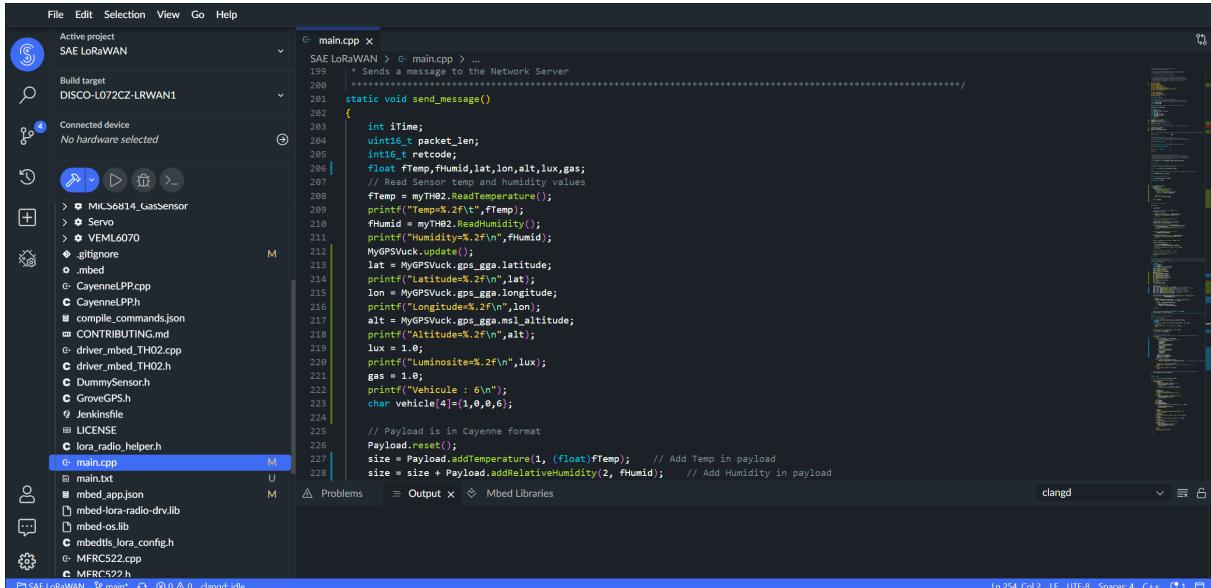


FIGURE 46 – Interface de l'IDE Cloud Keil Arm Studio

Le code est composé de deux fonctions essentielles : `send_message()` et `receive_message()`

La fonction `send_message()` lit les valeurs de température et d'humidité du capteur, ainsi que les coordonnées GPS, la luminosité et la concentration de gaz. Elle construit ensuite un message au format CayenneLPP en ajoutant toutes les données collectées. Enfin, elle envoie le message au réseau LoRaWAN.

Voici une explication détaillée des étapes du code :

1. Lire les données des capteurs :
 - La température est lue à l'aide de la fonction `myTH02.ReadTemperature()`.
 - L'humidité est lue à l'aide de la fonction `myTH02.ReadHumidity()`.
 - Les coordonnées GPS sont mises à jour à l'aide de la fonction `MyGPS.update()`.
 - Les coordonnées GPS actuelles (latitude, longitude et altitude) sont lues à partir de la structure `gps_gga` de la classe **MyGPS**.
 - La luminosité est lue grâce à la fonction `MyLumino.readUV()`.
 - La concentration de gaz NO2 est mise à jour à l'aide de la fonction `MyGas.getGas(NO2)`.
2. Construire le message CayenneLPP :
 - La fonction `Payload.reset()` initialise la structure **Payload**.
 - La température est ajoutée au message en utilisant la fonction `Payload.addTemperature(1, (float)fTemp)`.
 - L'humidité est ajoutée au message en utilisant la fonction `Payload.addRelativeHumidity(2, fHumid)`.
 - Les coordonnées GPS sont ajoutées au message en utilisant la fonction `Payload.addGPS(3, lat, lon, alt)`.
 - La luminosité est ajoutée au message en utilisant la fonction `Payload.addLux(5, lux)`.
 - La concentration de gaz est ajoutée au message en utilisant la fonction `Payload.addAnalogInput(6, gas)`.
 - Un identifiant de véhicule est ajouté au message en utilisant la fonction `Payload.addVEHICLE(7, vehicle)`.
3. Envoyer le message :
 - La fonction `lorawan.send()` envoie le message au réseau LoRaWAN en utilisant le port **MBED_CONF_LORA_APP_PORT**.
4. Gérer les erreurs :
 - Si l'envoi du message échoue, un message d'erreur est affiché et le message est réessayé dans 3 secondes.

```

1 static void send_message()
2 {
3     int iTime;
4     uint16_t packet_len;
5     int16_t retcode;
6     float fTemp ,fHumid ,lat ,lon ,alt ,lux ,gas ;
7     // Read Sensor temp and humidity values
8     fTemp = myTH02.ReadTemperature();
9     printf("Temp=% .2f \t",fTemp);
10    fHumid = myTH02.ReadHumidity();
11    printf("Humidity=% .2f \n",fHumid);
12    MyGPSVuck.update();
13    lat = MyGPS.gps_gga.latitude;
14    printf("Latitude=% .2f \n",lat);
15    lon = MyGPS.gps_gga.longitude;
16    printf("Longitude=% .2f \n",lon);
17    alt = MyGPS.gps_gga.msl_altitude;
18    printf("Altitude=% .2f \n",alt);
19    lux = MyLumino.readUV();
20    printf("Luminosite=% .2f \n",lux);
21    gas = MyGas.getGas(N02);
22    printf("Vehicule : 6\n");
23    char vehicle[4]={1,0,0,6}; // Groupe 1 - Table 6
24
25    // Payload is in Cayenne format
26    Payload.reset();
27    size = Payload.addTemperature(1, (float)fTemp); // Add Temp in
payload
28    size = size + Payload.addRelativeHumidity(2, fHumid); // Add
Humidity in payload
29    size = size + Payload.addGPS(3, lat, lon, alt); // Add GPS in
payload
30    size = size + Payload.addNFC(4, getNFC()); // Add luminosity in
payload
31    size = size + Payload.addLuminosity(5, lux); // Add luminosity in
payload
32    size = size + Payload.addAnalogInput(6, gas); // Add gas sensor in
payload
33    size = size + Payload.addVEHICLE(7, vehicle); // add vehicle
34
35    // Send complete message with cayenne format
36    retcode = lorawan.send(MBED_CONF_LORA_APP_PORT, Payload.getBuffer(),
Payload.getSize(),
37                                MSG_UNCONFIRMED_FLAG);
38
39    if (retcode < 0) {
40        retcode == LORAWAN_STATUS_WOULD_BLOCK ? printf("send - WOULD
BLOCK\r\n")
41        : printf("\r\n send() - Error code %d \r\n", retcode);
42
43        if (retcode == LORAWAN_STATUS_WOULD_BLOCK) {
44            //retry in 3 seconds
45            if (MBED_CONF_LORA_DUTY_CYCLE_ON) {
46                ev_queue.call_in(3000, send_message);
47            }
48        }
49    }
50 }
```

```

51     printf("\r\n %d bytes scheduled for transmission \r\n", retcode);
52     memset(tx_buffer, 0, sizeof(tx_buffer));
53 }

```

Listing 1 – Code de la fonction send_message()

La fonction *receive_message()* est responsable de la réception de messages sur les ports 3 et 4 et de l'exécution des actions appropriées en fonction du contenu des messages. Elle lit d'abord le nombre de ports, la position initiale, l'index et les états inconnu, pas de démarrage et démarrage. Ensuite, elle lit le port, les drapeaux et le code retour de la fonction *lorawan.receive()*. Si le code retour est inférieur à 0, elle affiche un message d'erreur. Sinon, elle affiche les données reçues sur le port sélectionné et traite le message en fonction du port.

Sur le port 3, la fonction vérifie si le premier octet du message est **0x30**, **0x31** ou **0x32**. Si c'est le cas, elle affiche le message "led = %x" et exécute la commande ASCII correspondante pour contrôler la LED. Les trois commandes possibles sont :

- 0x30 : Démarrage refusé : la LED rouge s'allume
- 0x31 : Démarrage autorisé : la LED verte s'allume
- 0x32 : Badge Inconnu : la LED orange s'allume

Elle met ensuite à jour les états des broches (activation/désactivation du MOSFET) en fonction de la commande exécutée.

Sur le port 4, la fonction convertit chaque octet du message en un chiffre décimal et les ajoute à la variable iPosition. Une fois tous les octets convertis, elle affiche le message "Servo position = %d" et utilise la valeur de iPosition pour mettre à jour la position du servomoteur. La valeur maximale de iPosition est 180, ce qui correspond à une position angulaire de 180 degrés. Nous n'avons pas pu utiliser cette fonctionnalité durant la SAE LoRaWAN.

Si le port est inconnu, la fonction affiche le message "port inconnu = %d". Enfin, la fonction efface le buffer *rx_buffer* pour la prochaine réception de message.

```

1 static void receive_message()
2 {
3     int num_port, iPosition=0,iIndex,etatInconnu, etatPasDem, etatDem;
4     uint8_t port;
5     int flags;
6     int16_t retcode = lorawan.receive(rx_buffer, sizeof(rx_buffer), port
7 , flags);
8     if (retcode < 0) {
9         printf("\r\n receive() - Error code %d \r\n", retcode);
10        return;
11    }
12    printf(" RX Data on port %u (%d bytes): ", port, retcode);
13    for (uint8_t i = 0; i < retcode; i++) {
14        printf("%02x", rx_buffer[i]);
15    }
16
17    switch (port) {
18        case 3: // control connu
19            printf("\n led=%x", (int)rx_buffer[0]);
20            if (rx_buffer[0]==0x30) {// ascii command for led
21                etatInconnu=0;
22                etatPasDem=1;
23                etatDem=0;
24            } else if (rx_buffer[0]==0x31) {
25                etatInconnu=0;
26                etatPasDem=0;
27                etatDem=1;
28            } else if (rx_buffer[0]==0x32) {
29                etatInconnu=1;
30                etatPasDem=0;
31                etatDem=0;
32            }
33            inconnu.write(etatInconnu);
34            pasdemarrage.write(etatPasDem);
35            demarrage.write(etatDem);
36            break;
37        case 4: // control servomotor
38            for (iIndex=0; iIndex<retcode; iIndex++) {
39                iPosition = iPosition*10 + (rx_buffer[iIndex]-0x30);
40            }
41            printf("\n Servo position =%d",iPosition);
42            Myservo.position ( iPosition-45 );
43            break;
44        default:
45            printf("\n port inconnu =%d", (int)port);
46            break;
47    }
48    memset(rx_buffer, 0, sizeof(rx_buffer));
49 }

```

Listing 2 – Code de la fonction receive_message()

5.2 NodeRED

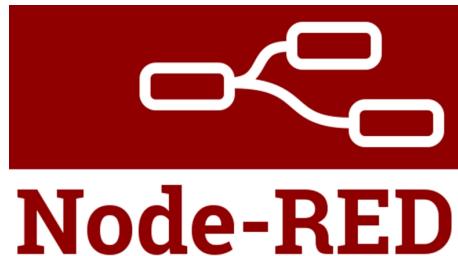


FIGURE 47 – Logo NodeRED

NodeRED est un outil de développement visuel (low-code) basé sur des flux pour connecter ensemble des périphériques matériels, des API et des services en ligne. Il est utilisé pour créer des applications IoT, des applications Web, des applications d'automatisation et bien plus encore.

Il est basé sur le langage de programmation JavaScript et utilise une approche de programmation visuelle pour créer des applications. Les applications NodeRED sont constituées de flux, qui sont des assemblages de noeuds. Les noeuds sont des blocs de construction de base qui peuvent être utilisés pour effectuer des tâches telles que la lecture de données d'un capteur, l'appel d'une API ou l'affichage de données sur un écran.

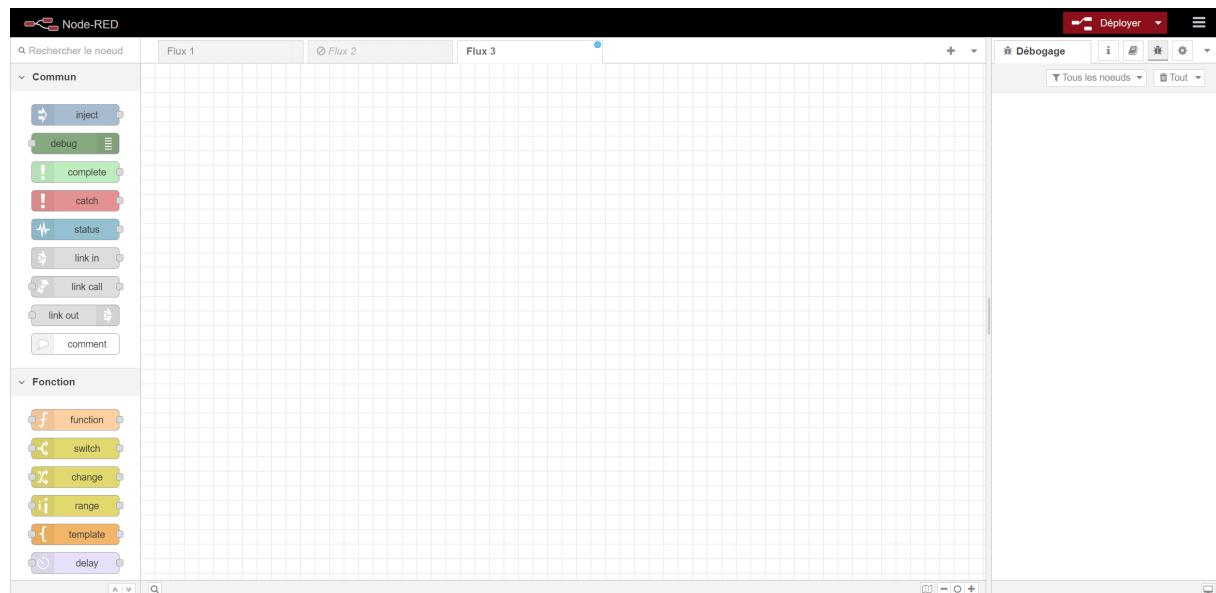


FIGURE 48 – Interface graphique NodeRED

NodeRED est un outil puissant et flexible qui peut être utilisé pour créer une grande variété d'applications. Il est facile à apprendre et à utiliser, même pour les personnes qui n'ont pas de connaissances en programmation.

Avantages de l'utilisation de NodeRED :

- Facile à apprendre et à utiliser : L'interface utilisateur de NodeRED est intuitive et facile à comprendre. Même les personnes qui n'ont pas de connaissances en programmation peuvent créer des applications avec NodeRED.
- Flexible et extensible : NodeRED dispose d'une large gamme de noeuds qui peuvent être utilisés pour effectuer une grande variété de tâches. Il existe également une large communauté d'utilisateurs de NodeRED qui développent des noeuds personnalisés.
- Open-source : NodeRED est un logiciel open source, ce qui signifie qu'il est gratuit et disponible à tous.

Dans le cadre de la SAE LoRaWAN, notre groupe a développé deux flux NodeRED permettant d'atteindre les objectifs demandés.

Le premier flux est le plus important, celui-ci permet de :

- Récupérer les données du site The Things Network via le protocole MQTT
- Traiter les données grâce à du code JavaScript
- Afficher dans un dashboard les données traitées
- Sauvegarder les données sur une base de données InfluxDB
- Et vérifier si le badge présenté est autorisé à démarrer le Kiffy

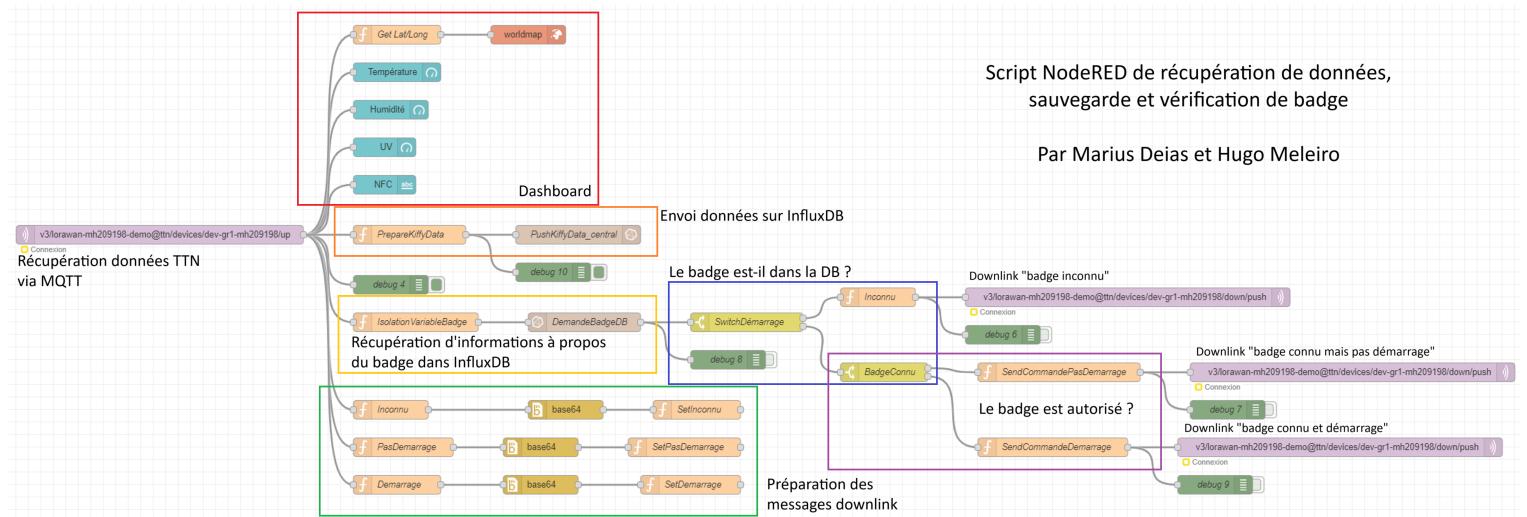


FIGURE 49 – Flux de récupération, sauvegarde et vérification des données

5.2.1 Récupération des données



FIGURE 50 – *Récupération des données MQTT*

Le protocole MQTT est un protocole de communication léger et efficace, conçu pour l'échange de données entre des appareils connectés. Il utilise un modèle d'éditeur/abonné, dans lequel les appareils publient des messages sur des sujets, et d'autres appareils peuvent s'abonner à ces sujets pour recevoir les messages. MQTT est souvent utilisé dans les applications IoT, où il permet aux appareils de communiquer entre eux et avec le cloud.

Lors de la SAE LoRaWAN, nous avons utilisé le protocole MQTT afin de recevoir/- transmettre des informations au serveur The Things Network. Il agit comme passerelle entre NodeRED et The Things Network.

Chaque fois que laMBED envoie un message, celui-ci sera relayé via MQTT sur NodeRED puis traduit en JSON pour pouvoir exploiter les informations.

5.2.2 Affichage des données

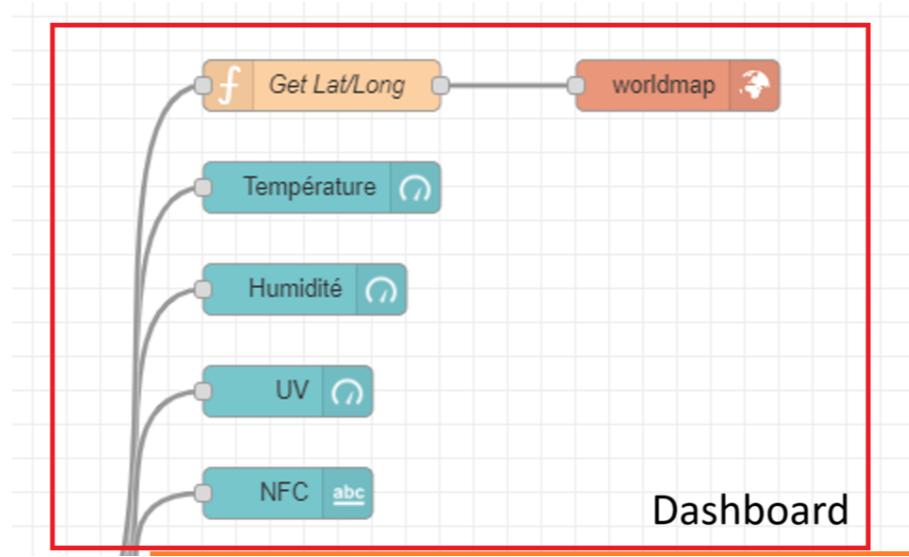


FIGURE 51 – Paramétrage du dashboard

Après la réception d'un message MQTT, nous affichons immédiatement les valeurs de température, humidité, ID RFID NFC et l'index UV sur un dashboard graphique disponible à l'adresse <http://localhost:1880/ui>.

L'intérêt d'utiliser un tableau de bord graphique est de pouvoir visualiser rapidement et facilement des données complexes. Les tableaux de bord graphiques utilisent des graphiques, des diagrammes et d'autres éléments visuels pour représenter les données de manière intuitive et facile à comprendre. En effet au lieu de lire du texte ou pire du JSON/XML, nous avons la possibilité de mettre en place facilement une interface intuitive et lisible.

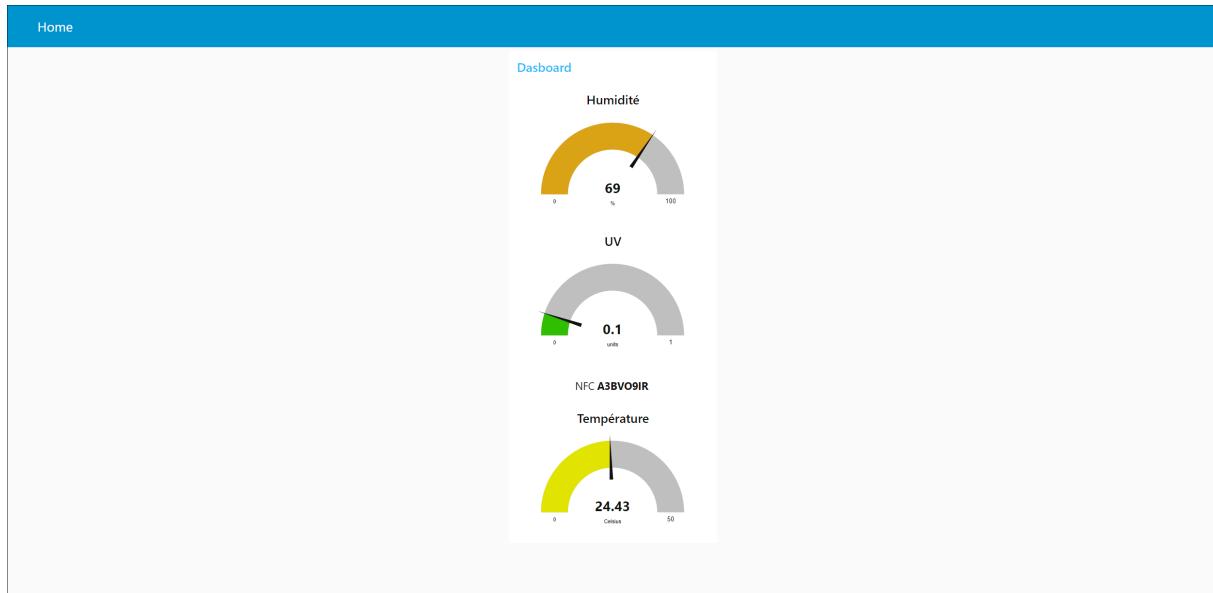


FIGURE 52 – Affichage du dashboard

Un autre module utile de la palette NodeRED : le WorldMap. Celui-ci permet de récupérer les données de latitude / longitude et d'afficher un beau pin rouge à l'emplacement sur une carte OpenStreetMap.

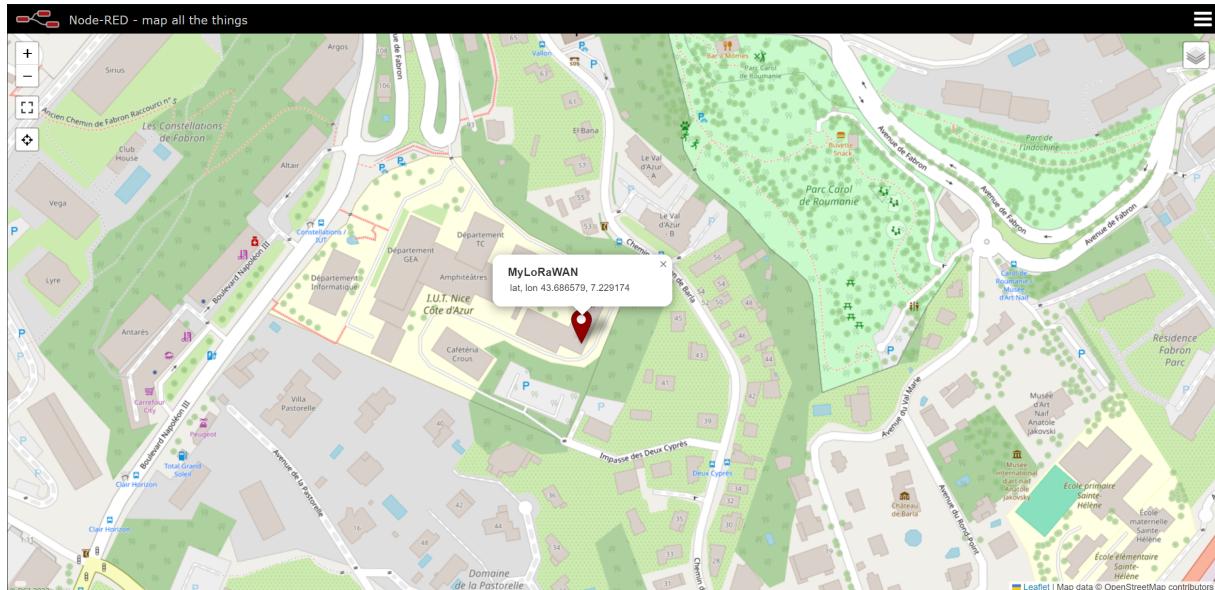


FIGURE 53 – Affichage de la worldmap avec le Kiffy

5.2.3 Sauvegarde des données

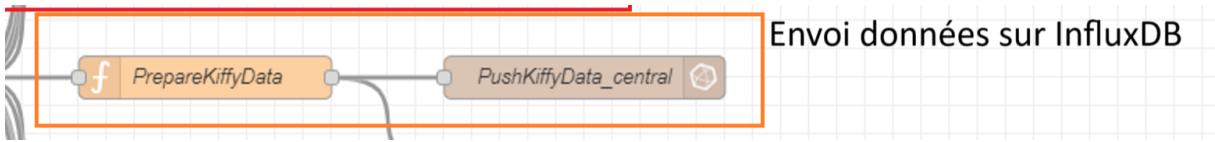


FIGURE 54 – Sauvegarde dans InfluxDB

Afin de pouvoir conserver un certain historique des données concernant la flotte de Kiffy's mais aussi pour libérer une charge de travail à NodeRED, nous utilisons InfluxDB qui reçoit les données parsées (voir parser ci-dessous) prêtes à être utilisées pour plus tard, notamment dans la vérification du badge et des requêtes du serveur HTTP.

```
1 msg.payload={  
2     "TEMPERATURE": msg.payload.uplink_message.decoded_payload.  
temperature,  
3     "HUMIDITE": msg.payload.uplink_message.decoded_payload.humidity,  
4     "LAT": msg.payload.uplink_message.decoded_payload.latitude,  
5     "LON": msg.payload.uplink_message.decoded_payload.longitude,  
6     "ALT": msg.payload.uplink_message.decoded_payload.altitude,  
7     "NFC": msg.payload.uplink_message.decoded_payload.RFID,  
8     "KIFFY_ID": 1,  
9     "UV": msg.payload.uplink_message.decoded_payload.luminosity,  
10    "GAS": msg.payload.uplink_message.decoded_payload.gas  
11 }  
12 return msg;
```

Listing 3 – Parser pour envoi InfluxDB

5.2.4 Vérification du badge

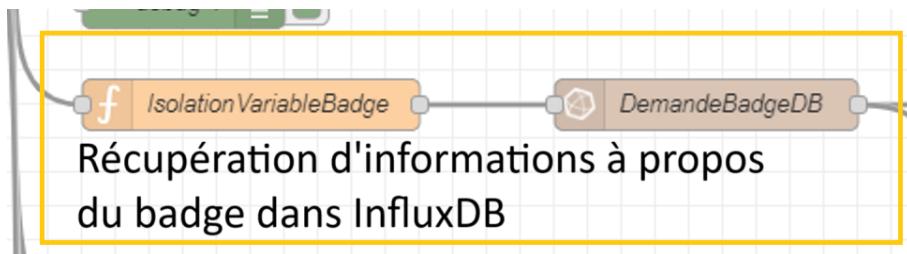


FIGURE 55 – Récupération d'infos sur le badge sur InfluxDB

La vérification du badge NFC se fait sur plusieurs étapes : la lecture du badge, la lecture des droits liés à ce badge et au downlink (envoi des informations) sur laMBED. Sur le script ci-dessous, on isole la variable RFID du Payload reçu par MQTT et on formate la requête InfluxDB sous *msg.query*.

```

1 var nfcTag = msg.payload.uplink_message.decoded_payload.RFID;
2 nfcTag = nfcTag.toString(); // On transforme l'objet JS en str
3 nfcTag = nfcTag.toUpperCase(); // On met tout en MAJ
4
5 // On formate la query InfluxDB
6 msg.query = "select * from BADGES where BADGE_ID='"+ nfcTag +"'";
7
8 return msg;

```

Listing 4 – Isolation de la variable RFID

Entre temps, on place dans des variables globales, les messages possibles sous forme d'une chaîne de caractères codé en base64 (0 → **MA==** pour PasDémarrage, 1 → **MQ==** pour Démarrage et 2 → **Mg==** pour Inconnu)

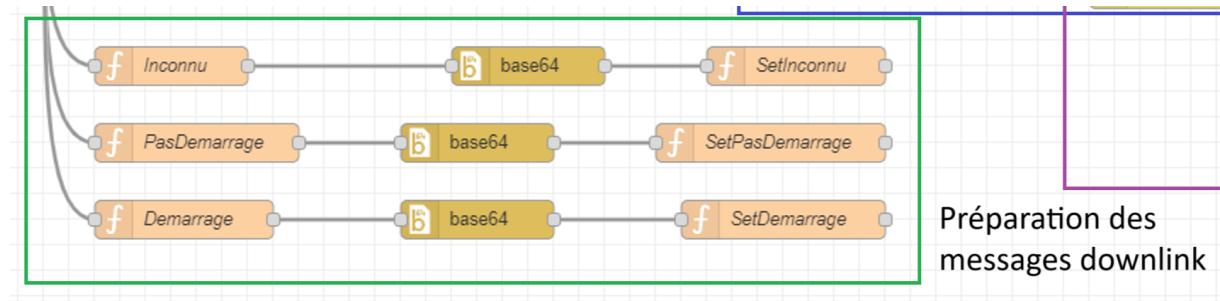


FIGURE 56 – Préparation des variables de Downlink

Puis, suite à la requête InfluxDB, on a les trois possibilités :

- Si InfluxDB ne retourne rien, c'est que le badge n'est pas connu et donc pas dans la base de données. Le démarrage est donc refusé et la LED orange s'allumera sur la mBED.

```
1     msg.payload = {
2         downlinks: [
3             {
4                 "f_port": 3,
5                 "frm_payload": flow.get("inconnu"),
6                 "priority": "NORMAL"
7             }
8         ]
9     }
10    return msg;
```

Listing 5 – Downlink Inconnu

- En revanche, si InfluxDB retourne un Object JavaScript, alors on obtient une information *DEM*. Si cette variable est à 0, alors le démarrage est refusé et la LED rouge s'allumera sur la mBED.

```
1     msg.payload = {
2         downlinks: [
3             {
4                 "f_port": 3,
5                 "frm_payload": flow.get("pasdemarrage"),
6                 "priority": "NORMAL"
7             }
8         ]
9     }
10    return msg;
```

Listing 6 – Downlink PasDémarrage

- Enfin, si la variable *DEM* est à 1, alors le démarrage est autorisé, la LED verte s'allumera sur la mBED et le MOSFET de démarrage allume le moteur électrique.

```
1     msg.payload = {
2         downlinks: [
3             {
4                 "f_port": 3,
5                 "frm_payload": flow.get("demarrage"),
6                 "priority": "NORMAL"
7             }
8         ]
9     }
10    return msg;
```

Listing 7 – Downlink Démarrage

Le badge est-il dans la DB ?

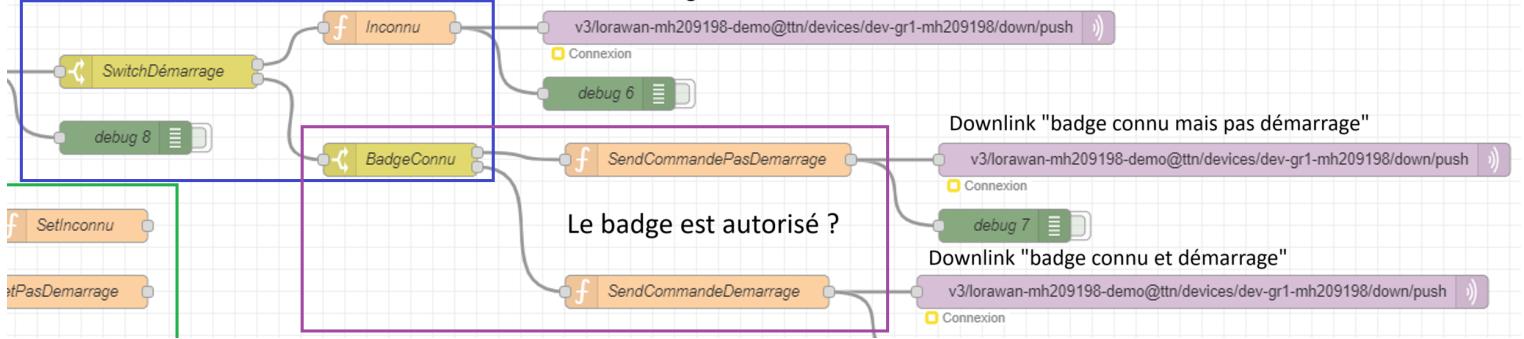


FIGURE 57 – Vérification du badge

Après toute la chaîne de traitement effectuée et le message Downlink prêt, on envoie celui-ci grâce au protocole MQTT à The Things Network qui gérera les temps de transmission après la réception. (voir Protocole LoRaWAN) Le flux NodeRED paraît complexe mais sa logique est simple comme le montre l'ordinogramme ci-dessous.

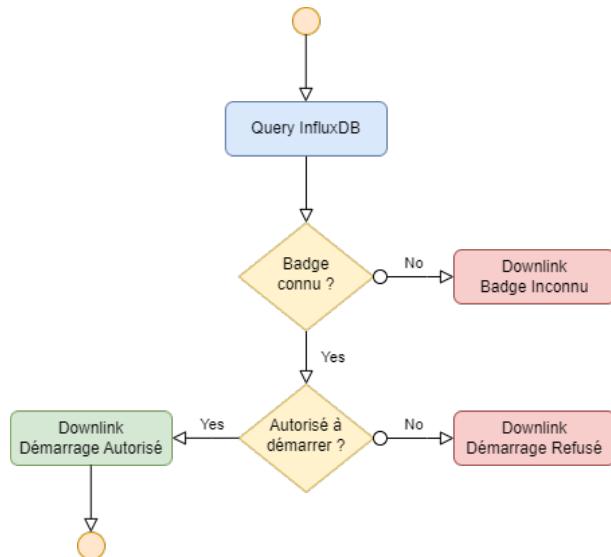


FIGURE 58 – Ordinogramme de la vérification

5.2.5 Service HTTP

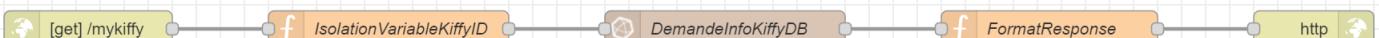


FIGURE 59 – Flux serveur HTTP

Enfin, nous avons créé un deuxième flux pour traîter la partie Serveur et Requêtes HTTP. NodeRED attend une requête HTTP GET sur l'URL `http://localhost:1880/mykiffy` avec une query `?query=myKiffyID`.

On isole ce paramètre et effectuons une requête SELECT sur la base de données InfluxDB. Si des données concernant ce Kiffy existent, alors on sélectionne la donnée la plus récente et on formate le JSON retourné par InfluxDB.

Ainsi, le client HTTP ayant effectué la requête reçoit une réponse HTTP OK 200 avec comme corps le JSON formatté. En revanche, si aucune donnée existe, le client recevra une réponse HTTP ERROR 404. Ce serveur HTTP sera utile lors du développement de l'application mobile. (Voir Application mobile)

5.3 InfluxDB



FIGURE 60 – Logo InfluxDB

InfluxDB est un système de gestion open source de base de données temporelles conçu pour stocker et analyser des données de séries temporelles. Elle est utilisée dans une grande variété d'applications, notamment la surveillance, la gestion des actifs, l'Internet des objets (IoT) et l'analyse en temps réel.

Les bases de données temporelles

Les bases de données temporelles sont conçues pour stocker des données qui sont mesurées à des intervalles réguliers. Ces données peuvent être des mesures physiques, telles que la température ou la pression, ou des mesures d'utilisation, telles que le nombre de requêtes HTTP reçues par un serveur.

Les avantages d'InfluxDB

InfluxDB offre un certain nombre d'avantages par rapport à d'autres bases de données temporelles, notamment :

- Performances : InfluxDB est conçu pour offrir des performances élevées pour les opérations d'écriture et de lecture de données.
- Scalabilité : InfluxDB est capable de gérer des volumes de données importants.
- Facilité d'utilisation : InfluxDB est une base de données facile à utiliser, avec une API REST complète et un langage de requêtes puissant.

InfluxDB peut être installé sur un serveur local ou sur un cloud. La configuration d'InfluxDB est simple et rapide.

```
root@a0d7b954-influxdb:/# influx -username homeassistant -password homeassistant
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> show databases
name: databases
name
-----
_internal
homeassistant
> use homeassistant
Using database homeassistant
> show measurements
name: measurements
name
-----
state
> select * from state
name: state
time          attribution_str
```

FIGURE 61 – Interface CLI InfluxDB

Pour commencer à utiliser InfluxDB, il faut créer une base de données et une table. Une base de données est un conteneur pour les données, et une table est un ensemble de données organisées. Les données sont insérées dans InfluxDB sous forme de séries temporelles. Une série temporelle est une collection de points de données, chacun ayant une valeur, une date et une heure.

Lors de la SAE LoRaWAN, nous avons utilisé un NAS Synology connecté sur le réseau et possédant un container Docker InfluxDB. Dans cette instance d’InfluxDB nous avons, grâce à NodeRED, publié des données sur la database GR1_2024 de type : TEMPERATURE, HUMIDITE, LAT, LON, ALT, NFC, KIFFY_ID, UV, GAZ.

5.4 Application mobile



FIGURE 62 – Logos Expo et React Native

Pour réaliser l’application, nous avons choisi d’utiliser le framework React Native avec une surcouche basée sur le SDK Expo 49. (la même architecture que dans les applications très populaires comme UniceNotes, Facebook, Uber, ...)

React Native est un framework d’applications mobiles open source créé par Facebook (Meta). Il est utilisé pour développer des applications pour Android, iOS et UWP (Windows) en permettant aux développeurs d’utiliser React avec les fonctionnalités natives de ces plateformes. La surcouche Expo SDK 49 permet d’utiliser un grand nombre de bibliothèques natives (caméra, GPS, puce 5G, ...) propres à l’appareil mobile sur une application React Native.

L’entièreté du code est rédigé en JavaScript et tient sur 54 lignes. L’utilisateur ouvre l’application qui est en majeure partie composé d’un plan sur lequel apparaît sa position à l’aide d’un point bleu. Puis, deux boutons en dessous de la carte permettent de récupérer des informations sur le Kiffy ou d’actualiser la position.

La première option s’appuie du serveur HTTP mis en place grâce à Node-RED. Dès que l’utilisateur appuie sur le bouton **Get Kiffy Data**, cela déclenche la fonction `getKiffyData(kiffyid)`. Cette fonction réalise une requête GET HTTP et reçoit une Promise JavaScript qui est transformée en JSON puis formatée pour récupérer les données utiles. Enfin, lorsque toutes les données utiles sont récoltées et formatées, un pin rouge apparaît sur la carte interactive sur la position du Kiffy choisi. Un appui sur ce pin affiche la température. (voir photos ci-dessous)

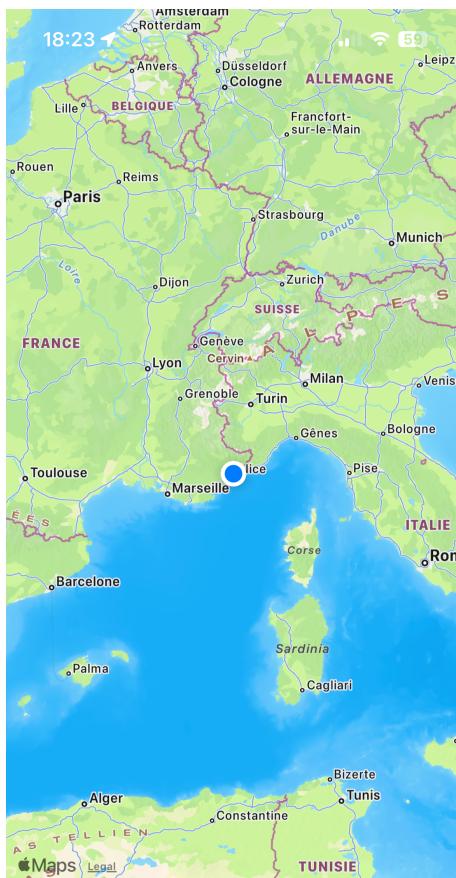
La seconde option s'appuie du module *expo-location* pour récupérer les coordonnées GPS de la puce intégrée au smartphone. La position de l'utilisateur est donc mise à jour lors de l'appui sur le bouton **Locate Me**.

```

51     getKiffyData('01000006')} /> // Bouton qui recupere les donnees du
52     Kiffy
53     <Button style={{flex:1}} title="Locate Me" onPress={() =>
54       getLocation()} /> // Bouton qui recupere la position de l'utilisateur
55   );
56 }

```

Listing 8 – Code de l'application MyKiffy



Get Kiffy Data

Locate Me



Get Kiffy Data

Locate Me

FIGURE 63 – Application ouverte

FIGURE 64 – Affichage du Kiffy

6 Conclusion

Le projet SAE LoRaWAN a été un projet très enrichissant pour nous. Il nous a permis de développer nos compétences en électronique, en informatique et en programmation. Nous avons également appris à travailler en équipe et à gérer un projet de A à Z.

Nous sommes satisfaits des résultats obtenus. L'antenne que nous avons conçue a des performances satisfaisantes et permet au tricycle électrique de communiquer avec le réseau LoRaWAN. La programmation que nous avons réalisée permet au tricycle de collecter des informations sur la qualité de l'air et de se géolocaliser.

Ce projet a un potentiel important. Il pourrait être utilisé pour équiper une flotte de tricycles électriques à Nice. Cela permettrait d'améliorer la qualité de l'air dans la ville et de faciliter la mobilité des habitants.

Voici quelques pistes d'amélioration pour le projet :

- L'antenne pourrait être améliorée pour augmenter son gain et sa directivité. Cela permettrait d'améliorer la portée et la fiabilité de la communication.
- La programmation pourrait être améliorée pour ajouter de nouvelles fonctionnalités, telles que la possibilité de suivre le parcours du tricycle ou de recevoir des notifications.

En ce qui concerne le déroulé du projet, il serait également pertinent de tester la carte dans un environnement réel, sur un kiffy électrique qui se déplacerait dans les conditions prévues. Cela permettrait de rendre le projet plus concret et de détecter d'éventuelles pistes d'amélioration qui ne seraient pas perceptibles en théorie.

7 Sources & Bibliographie

- Page Wikipedia sur le protocole LoRaWAN
- Documentation NodeRED
- Documentation InfluxDB
- Base C++ Mbed OS LoRaWAN
- Draw.io, utilisé pour réaliser les infographies
- GitHub, utilisé dans le cadre de la SAE
- The Things Network
- PathWave Advanced Design System

SAE LoRaWAN

I.U.T. Nice Côte d'Azur | GEII

2023-2024 | Conçu avec L^AT_EX

