

# Rapport de Travaux Pratiques - 2024.09.22

**Internet des objets - Protocoles LoRaWAN pour le Positionnement sans GPS  
par la Trilatération WiFi avec ESP8266 et modem LoRa E5**

## Table des matières

<b>Introduction.....</b>	<b>2</b>
<b>1. Calibrage de la mesure du RSSI en fonction de la distance.....</b>	<b>3</b>
1.1. Mise en place de points d'accès WiFi avec l'ESP8266.....	3
Figure 1.1.1 : Diagramme de séquence du déploiement de l'ESP en tant que point d'accès WiFi.....	3
Figure 1.1.2 : Visualisation du SSID depuis le gestionnaire WiFi d'un ordinateur.....	4
1.2. Estimation de la distance à partir du RSSI.....	4
Figure 1.2.1 : Diagramme de séquence du calcul des distances en fonction des RSSI collectés.....	5
Figure 1.2.2 : Graphique linéaire de la mesure du RSSI en fonction de la distance pour différentes contraintes environnementales.....	6
Figure 1.2.3 : Graphique semi-logarithmique de la mesure du RSSI en fonction de la distance pour différentes contraintes environnementales.....	7
Figure 1.2.4 : Graphique semi-logarithmique comparant la relation entre le RSSI et la distance, avec les valeurs mesurées dans un environnement peu contraint.....	8
<b>2. Positionnement estimé et transfert des données par protocole LoRaWAN vers le serveur.....</b>	<b>9</b>
2.1. Modèle de trilatération.....	9
Figure 2.1.1 : Diagramme de séquence du relevé et de l'envoi sur le réseau TTN des données collectées.....	10
Figure 2.1.2 : Support d'analyse des modèles de pondération en 3D sur GeoGebra.....	11
2.2. Architecture de l'envoi de la donnée.....	12
Figure 2.2.1 : Diagramme de séquence de l'envoi des données collectées sur le serveur.....	13
Figure 2.2.2 : Visualisation de la réception des relevés compactés et de leurs mise en forme sur le réseau TTN.....	14
Figure 2.2.3 : Visualisation de la mise en forme sur le réseau TTN des données appareils.....	15
<b>3. Traitement des données sur le serveur et visualisation de la position sur l'interface utilisateur.....</b>	<b>16</b>
<b>Conclusion.....</b>	<b>19</b>
<b>Annexe.....</b>	<b>20</b>
A. Mise en place de l'environnement de travail.....	20
A.1 Installation de Node-RED par étapes.....	20
A.2 Configuration de Arduino IDE pour échanger avec l'ESP.....	20
A.3 Configuration du matériel TTN par étapes.....	21
B. Brouillon du Relevé du RSSI en dB en fonction de la distance à la source dans différents environnements.....	25

## Introduction

Dans un environnement où les objets connectés, notamment les objets connectés mobiles comme les robots convoyeurs en entrepôts ou encore un simple smartphone, occupent une place de plus en plus importante, en particulier dans les environnements de travail fortement automatisés, il devient essentiel de pouvoir localiser des capteurs, des personnes et des objets mobiles. Ce système est appelé IPS ( système de géolocalisation en intérieur ).

En effet, dans des zones intérieures telles que les centres commerciaux, les parkings, les usines, ou même certaines rues étroites bordées de bâtiments hauts, comme le Paris d'avant [les grands travaux d'élargissement des rues menés par le préfet Haussmann](#), ou encore dans des zones forestières denses comme à Saclay, les signaux haute fréquence, à courte longueur d'onde des GPS satellites, dont la portée dépend d'une ligne de vue dégagée, sont souvent trop faibles pour traverser les bâtiments aux murs épais en béton.

Ainsi, pour guider avec précision une flotte de cyclistes en livraison ou pour orienter une personne dans un centre commercial, une nouvelle technologie de positionnement en intérieur est nécessaire. Nous avons exploré une méthode de trilatération par WiFi Sniffing qui permet de détecter les BSSID ( adresses MAC uniques de l'interface radio ) et leurs niveaux RSSI ( indicateurs de puissance du signal ) pour trianguler une position avec des balises de positionnement.

Après avoir établi une relation entre la distance à un émetteur WiFi et la puissance du signal reçu, les données seront transmises via le protocole LoRaWAN à un serveur où elles seront croisées avec une base de données telle que [Geolocalisation API de Google](#) ou [WIGLE](#).

Pour une meilleure gestion, portativité et communication de notre projet, nous avons, sur les conseils de Thibault, mis en place un suivi avec [GitHub](#) et une remise en ligne via [Google Doc](#).

Aide rédactionnelle, au débogage, au code, soutien et remerciement : Daniel, Yulin, Maxime, Ayman, Victor, Quentin, Karlitou, Sekouba, Nicolas, ChatGPT, HARIAN Elyoth, DOUZE Yann, Benjamin, l'incroyablissime Sylvain VIATEUR.

Lien utile : [https://fr.wikipedia.org/wiki/Système\\_de\\_positionnement\\_en\\_intérieur](https://fr.wikipedia.org/wiki/Système_de_positionnement_en_intérieur)

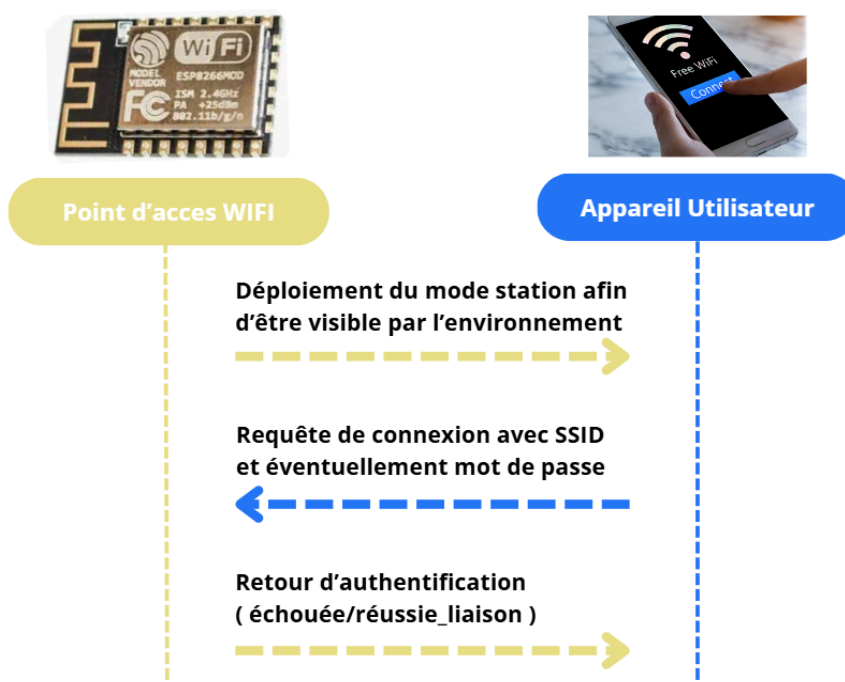
# 1. Calibrage de la mesure du RSSi en fonction de la distance

Dans cette première partie, nous chercherons à développer une méthode pour estimer la distance entre un appareil et une source émettant un signal WiFi. Cette estimation permettra, par la suite, de trianguler la position d'un nouvel appareil.

## 1.1. Mise en place de points d'accès WiFi avec l'ESP8266

Après avoir configuré l'environnement de travail disponible dans l'annexe [A. Mise en place de l'environnement de travail](#), nous allons, avec nos copromotionnaires, déployer un réseau d'ESP en mode AP ( point d'accès WiFi ) dont les emplacements seront connus. Cette technologie, fondée sur les normes IEEE 802.11, permet la communication de manière non filaire entre différents appareils, facilitant l'accès à Internet et aux réseaux locaux sur des [distances modérées](#). Elle offre des vitesses de transmission de plus en plus élevées grâce à l'amélioration de la bande passante et à l'utilisation de hautes fréquences, bien que celles-ci entraînent un coût énergétique croissant avec l'évolution des générations WiFi demandant une puissance plus élevée pour maintenir un signal stable sur de longues distances.

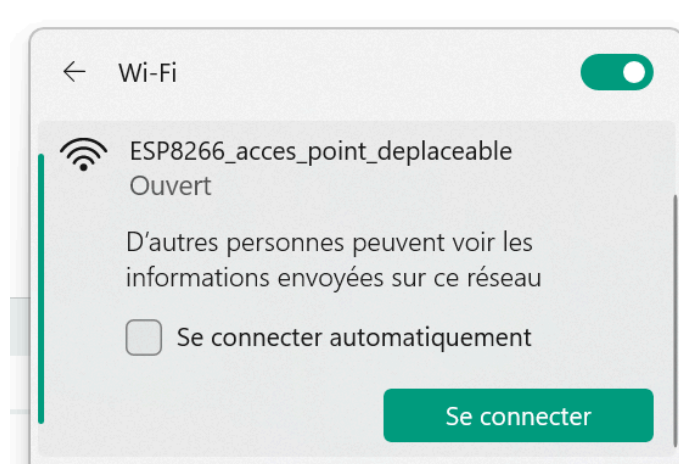
*Figure 1.1.1 : Diagramme de séquence du déploiement de l'ESP en tant que point d'accès WiFi.*



Nous avons choisi un débit de transmission légèrement plus lent que celui des affichages typiques, afin de pouvoir visualiser progressivement l'apparition des informations transmises. Cependant, en cas de besoin d'échanges plus lourds ou de communications en temps réel, il serait nécessaire d'augmenter le baudrate.

De plus, nous avons mis en place un système d'affichage des logs dans le terminal Serial Monitor, permettant d'afficher de manière récurrente l'adresse MAC de l'ESP et son SSID associé. Avec une légère modification, ces logs pourraient, dans une éventuelle future application, être affichés sur un mini module d'écran LCD, ce qui permettrait de retrouver l'adresse MAC d'un point d'accès sans avoir à passer par le Serial Monitor.

*Figure 1.1.2 : Visualisation du SSID depuis le gestionnaire WiFi d'un ordinateur.*



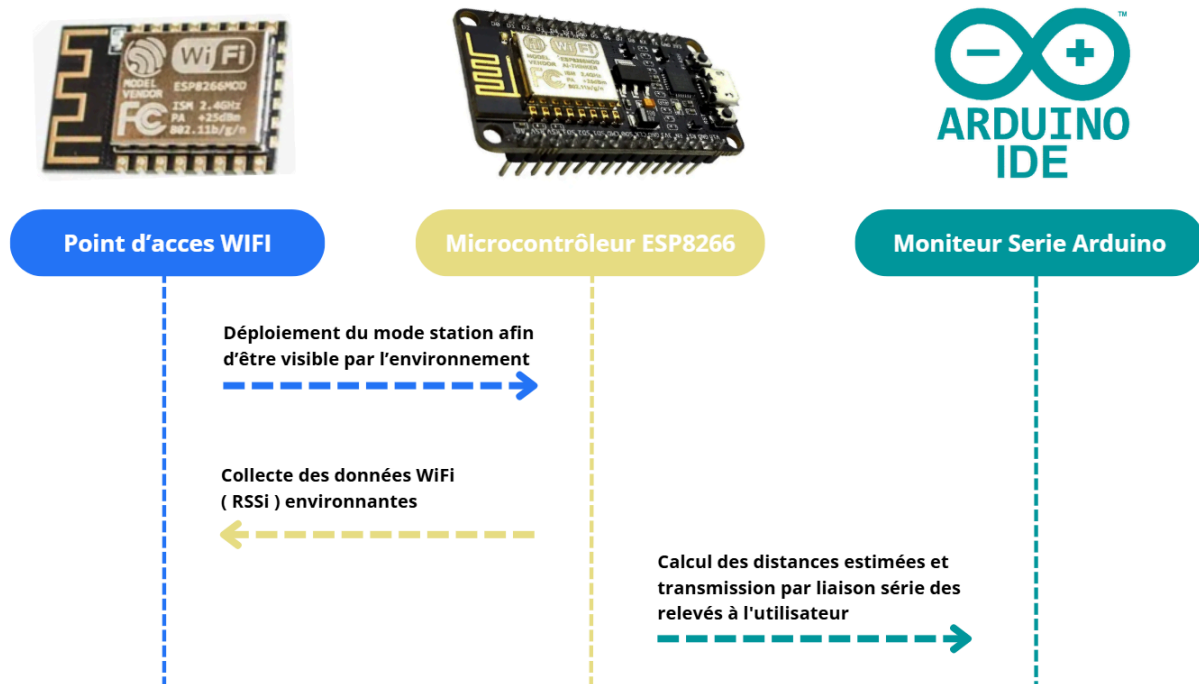
Nous constatons ainsi la bonne publication de notre réseau WiFi en mode point d'accès, ce qui conclut cette section.

## 1.2. Estimation de la distance à partir du RSSI

Dans cette partie, nous cherchons à établir la relation entre la puissance du signal reçu par notre ESP et sa distance par rapport aux appareils émettant un signal WiFi, tels que les ESP en mode point d'accès.

Le Wi-Fi est ici une technologie pertinente, car il permet non seulement de transmettre des données, mais aussi de fournir des informations de localisation à partir de l'intensité du signal reçu ( RSSI ) et des identifiants uniques des points d'accès WiFi ( adresses MAC ). En connaissant l'emplacement de chaque appareil connecté, un système de positionnement efficace et peu coûteux peut être mis en place. Bien que le WiFi soit énergivore, sa disponibilité dans les environnements urbains en fait une solution viable.

Figure 1.2.1 : Diagramme de séquence du calcul des distances en fonction des RSSI collectés.



Dans le cadre de ce projet, nous avons évalué la portée du signal en WiFi pour estimer les distances. Nous posons initialement l'hypothèse d'une relation linéaire entre la distance et le RSSI, selon la formule :

$$d = C_{P_{\text{linéique}}} \times RSSI + C^{ST0}$$

où  $d$  est la distance entre le récepteur et l'émetteur en mètres,

$C_{P_{\text{linéique}}}$  est le coefficient de puissance linéique du signal reçu en  $m. dB^{-1}$ ,

$RSSI$  est la force du signal reçu en  $dB$ ,

$C^{ST0}$  une constante correspondant à l'offset en mètres.

Nous proposons de réaliser les mesures dans trois conditions distinctes pour observer l'effet des obstacles sur la puissance du signal. Ainsi, à plusieurs distances ( de 0 m, avec les appareils collés l'un à l'autre, jusqu'à 10 m ), nous mesurerons :

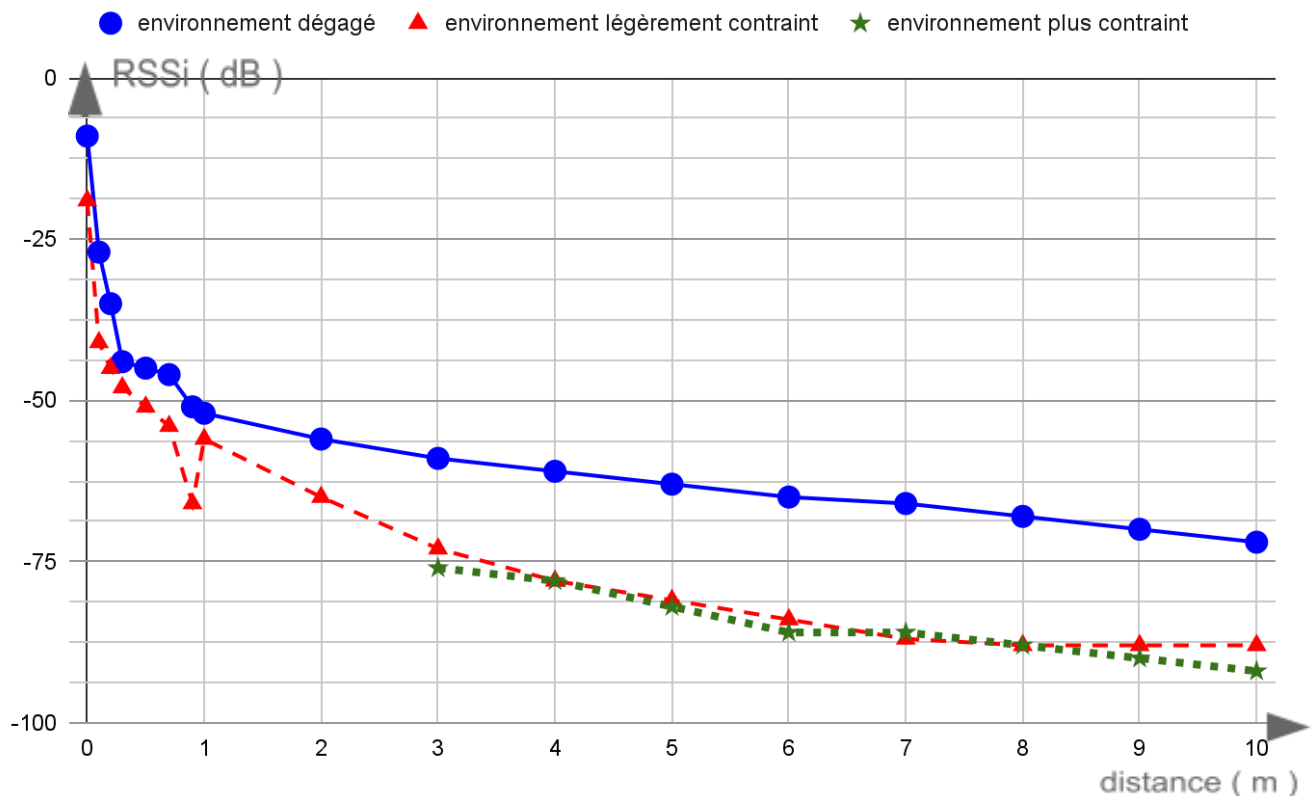
v ) Dans un environnement dégagé, sans obstacle entre l'émetteur et le récepteur.

vv ) Dans un environnement légèrement contraint ( présence d'obstacles mineurs tels qu'un écran d'ordinateur, une table, une chaise, du matériel de bureau, ... ).

vvv ) Dans un environnement plus contraint, avec des obstacles significatifs, tels qu'un ou plusieurs murs voir un plancher.

Ces mesures nous permettront d'évaluer si l'impact des obstacles entre la source et le récepteur est suffisamment important pour être pris en compte ou s'il peut être négligé.

*Figure 1.2.2 : Graphique linéaire de la mesure du RSSI en fonction de la distance pour différentes contraintes environnementales.*



*Graphique accessible aux daltoniens - 4% population française*

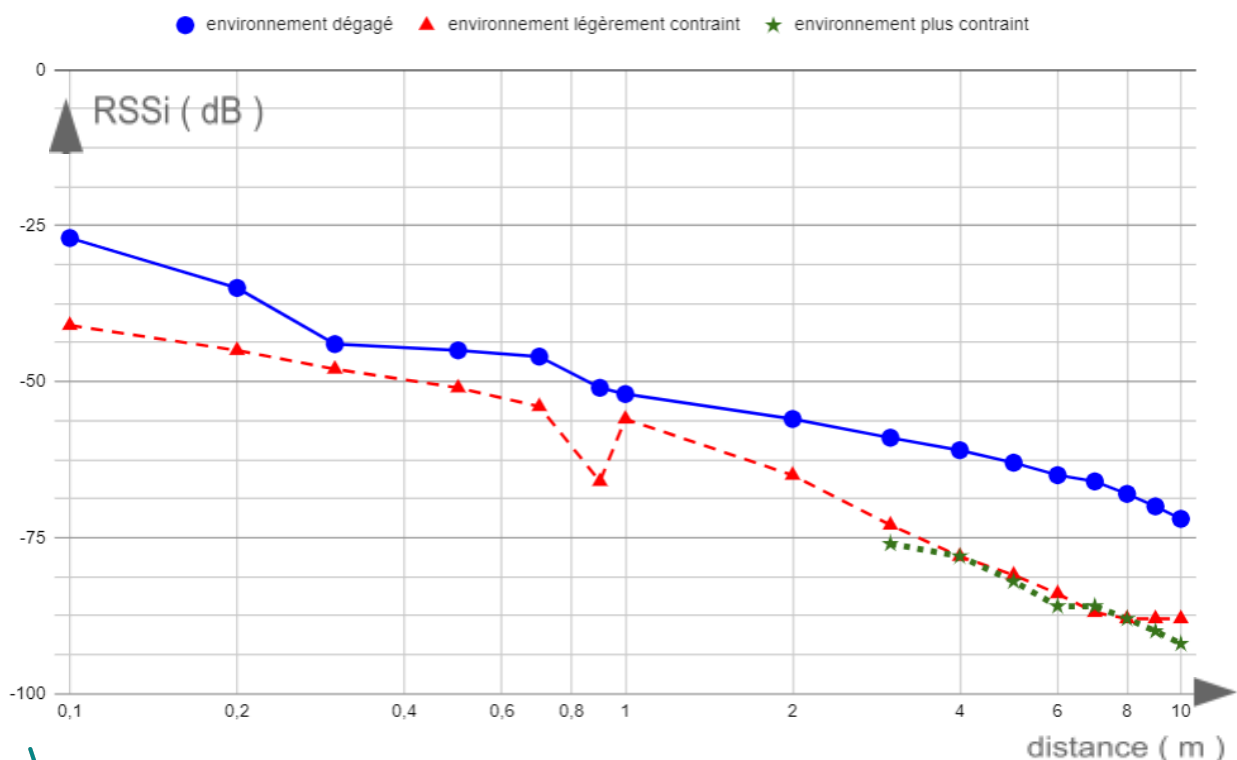
À la précision des relevés de mesure près, nous obtenons, malgré quelques anomalies (notamment un décalage à 1 m dans l'environnement dégagé, dû à une reprise des mesures à une distance plus proche) une tendance significative qui valide partiellement notre modèle. En effet, il semble acceptable d'utiliser une relation linéaire pour une distance allant de 1 à plusieurs mètres, en négligeant les très petites distances. Cela peut être suffisant pour des cas de localisation approximative, comme pour localiser un magasin dans un centre commercial, où une marge de 3 à 5 mètres dans la largeur des couloirs n'aura pas d'impact significatif. Cette approche peut également s'appliquer à un environnement comme une forêt dense, avec des points de connexion WiFi dispersés.



Ainsi, nous obtenons une loi relativement correcte pour les distances allant de 1 à 10 mètres, avec un coefficient de décroissance linéaire de  $-2 \text{ m} \cdot \text{dB}^{-1}$  et un offset raisonnable de  $-60 \text{ dB}$ . Nous postulons qu'il est préférable de surestimer la distance plutôt que de la sous-estimer, afin de créer des rayons de distance autour des émetteurs qui se superposent, délimitant ainsi une zone probable de localisation.

De plus, nous constatons que les différences de contraintes environnementales ( peu ou fortement contraint ) n'ont pas d'impact significatif pour des distances de quelques mètres.

*Figure 1.2.3 : Graphique semi-logarithmique de la mesure du RSSI en fonction de la distance pour différentes contraintes environnementales.*



Graphique accessible aux daltoniens - 4% population française

Dans un second temps, pour une meilleure approximation de la relation entre le RSSI et la distance, nous observons que cette relation semble plutôt logarithmique. En représentant la distance sur une échelle semi-logarithmique, nous obtenons des droites plus marquées et cohérentes, indiquant une relation plus fidèle entre le RSSI et la distance. Une relation logarithmique permettrait de couvrir les courtes distances, ce qui est particulièrement pertinent pour les applications de localisation plus précise.

Nous utiliserons donc la courbe rouge-triangle dans un environnement légèrement contraint pour trouver une relation plus fidèle à la réalité, et telle que :

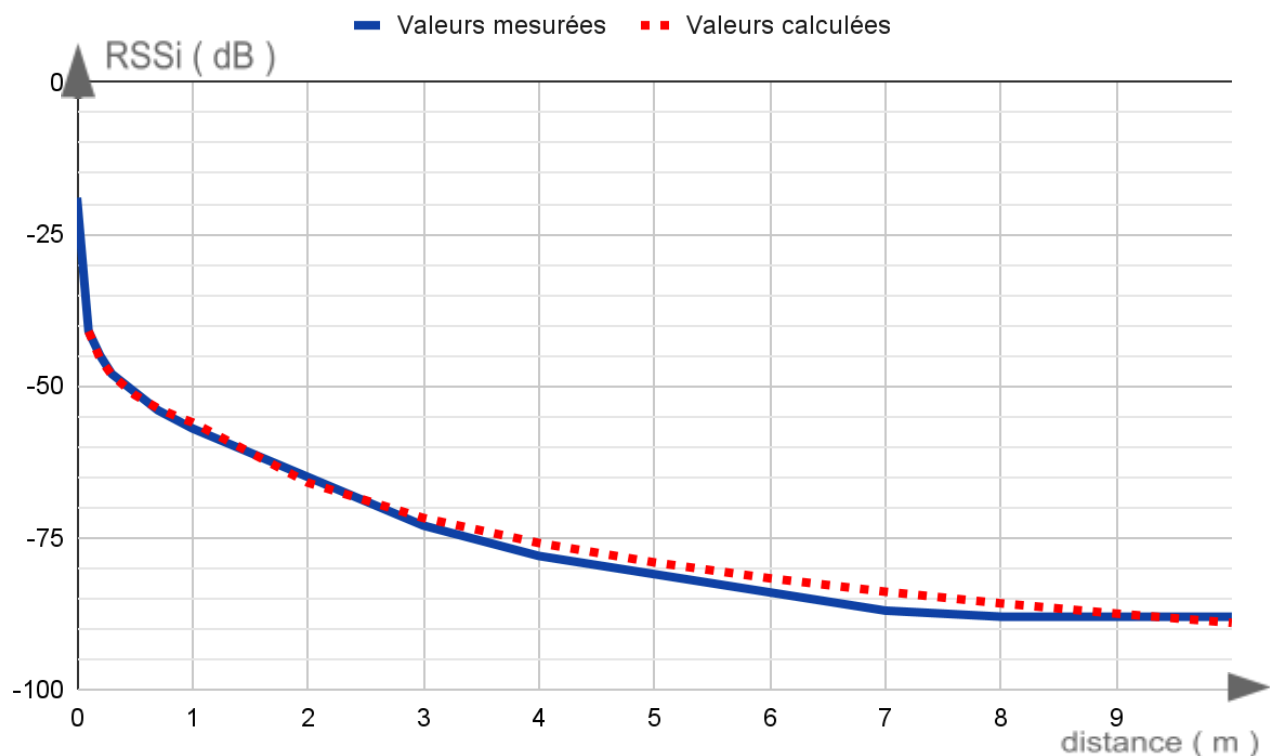
$$RSSi = (-41 - 15\log_{10}(10d)) \prod_{[0,1[} + (-56 - 33\log_{10}(d)) \prod_{[1,\infty[}$$

La fonction a deux principales parties :

a ) Pour les distances de 0,1 m à 1 m avec - 41dB d'offset et une pente de -15dB par décade ( où  $\prod_{[0,1[}$  est une fonction porte valant 1 de 0 à 1 exclu et 0 ailleurs ).

aa ) Pour les distances de 1 m à 10 m avec - 56dB d'offset et une pente de -33dB par décade ( où  $\prod_{[1,\infty[}$  est une fonction porte valant 1 de 1 à l'infini et 0 ailleurs ).

*Figure 1.2.4 : Graphique semi-logarithmique comparant la relation entre le RSSI et la distance, avec les valeurs mesurées dans un environnement peu contraint.*



Ainsi, nous obtenons la relation suivante :

$$\forall RSSi \in [ - 56 ; + \infty[ : d (RSSi) = 10^{-\left(\frac{RSSi + 56}{15}\right)}$$

$$\forall RSSi \in ] - \infty ; - 56[ : d (RSSi) = 10^{-\left(\frac{RSSi + 56}{33}\right)}$$



Après de nouvelles expérimentations, nous avons ajusté la première partie de la relation RSSI-distance en augmentant la pente pour les courtes distances. Ce choix repose sur l'observation que, dans des environnements dégagés, le signal est souvent de meilleure qualité à proximité immédiate des émetteurs, ce qui permet d'obtenir des relevés de RSSI plus précis et cohérents.

Nous avons également affiné notre formule d'estimation de distance de manière empirique, en nous concentrant sur la plage de 0,1 à 1 mètre sans obstacle. Cette portion de la courbe a montré une meilleure stabilité et précision. En revanche, les expérimentations en extérieur, bien que plus limitées, ont révélé une forte variabilité des résultats, probablement due à une dégradation du signal dans ces conditions. Les mesures extérieures doivent donc être interprétées avec précaution, car elles peuvent entraîner une imprécision accrue dans l'estimation des distances.

## 2. Positionnement estimé et transfert des données par protocole LoRaWAN vers le serveur

Dans cette deuxième partie, nous allons proposer un modèle pour approcher les coordonnées gps d'un récepteur par les relevés reçu en première partie et de les envoyer pour leurs exploitation graphique.

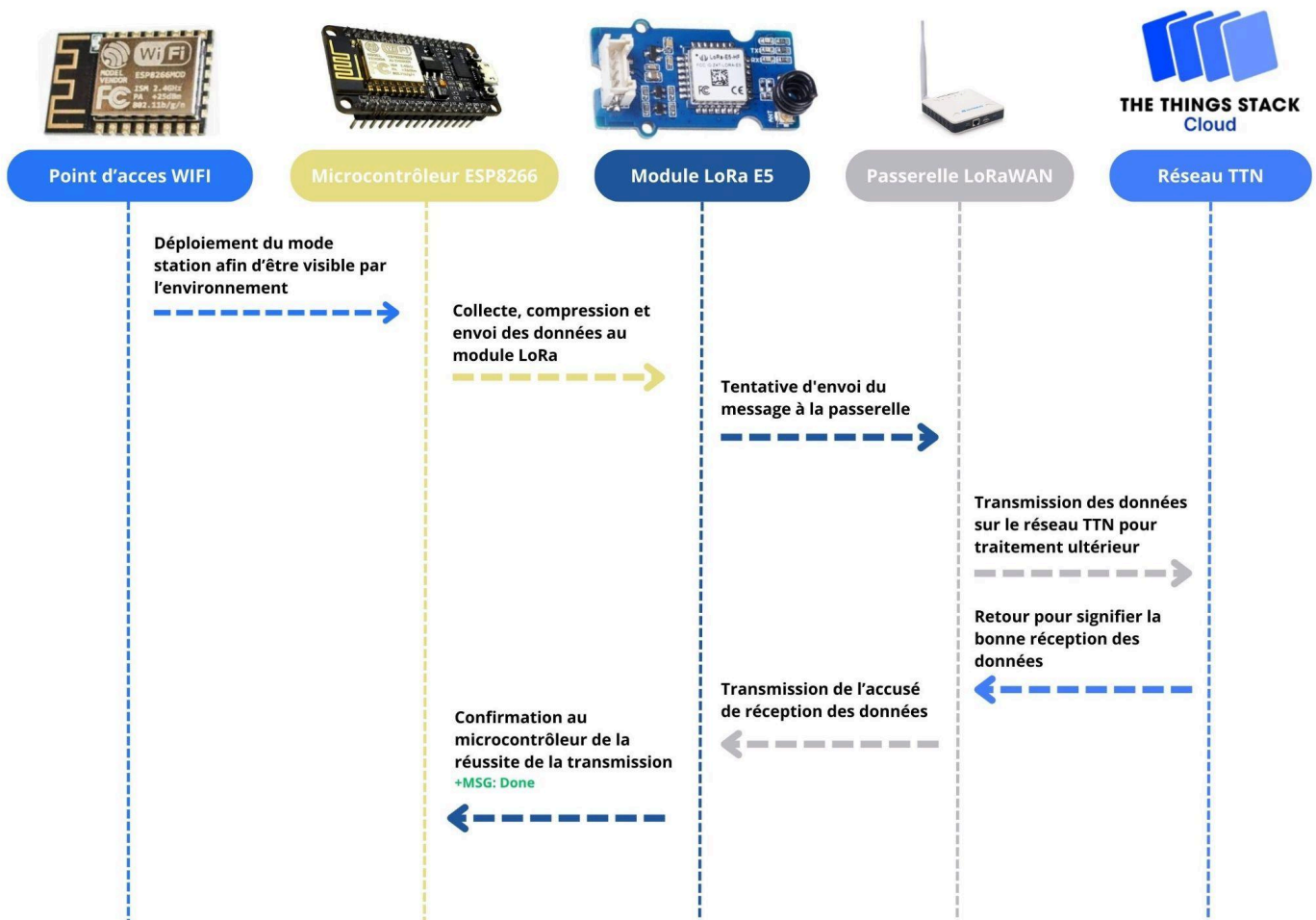
### 2.1. Modèle de trilatération

Pour estimer la position GPS basée sur les données WiFi et les relevés de RSSI, nous proposons ici un modèle de trilatération. Ce modèle repose sur l'idée que les valeurs les plus faibles de RSSI ( signal fort ) indiquent une proximité relative avec l'émetteur, car en intérieur ou en milieu urbain avec des obstacles, seuls des récepteurs proches peuvent capter un signal suffisamment fort.

Les mesures de RSSI pour un même point et réseau peuvent fluctuer dans le temps, en fonction des légers changements d'environnement, comme le passage d'une personne ou l'orientation du récepteur. Par conséquent, nous attribuons un poids plus élevé aux RSSI les plus faibles, car ils sont davantage représentatifs de courtes distances dans ces contextes. En revanche, des RSSI élevés ne garantissent pas systématiquement une grande distance, car des interférences peuvent fausser les mesures pour les points plus éloignés.

Idéalement, il aurait été préférable de disposer de relevés de RSSI projetés sur un plan en 2D pour minimiser les perturbations dans l'estimation des positions.

Figure 2.1.1 : Diagramme de séquence du relevé et de l'envoi sur le réseau TTN des données collectées.



Pour ce modèle, nous utilisons une moyenne pondérée des distances calculées, en affectant un poids plus élevé aux distances courtes (RSSI fortement proche de zéro) et un poids plus faible aux distances longues (RSSI faiblement proche de zéro). Ce système permet d'obtenir une estimation plus stable des coordonnées GPS en intégrant la variabilité des RSSI dans le calcul de distance. Ainsi nous obtenons comme 1er modèle :

$\forall$  point émetteur  $p_n$ , de coordonnées  $(x_n, y_n, z_n)$  où  $x_n$  la latitude,  $y_n$  la longitude et  $z_n$  l'altitude, et de distance estimée  $d_n$  où  $N$  le nombre de points en compte.

Les coordonnées  $(x_{recepteur}, y_{recepteur})$  sont alors :

$$x_{recepteur} = \frac{\sum_{n=1}^N \left( x_n \times \frac{1}{d_n} \right)}{\sum_{n=1}^N \left( \frac{1}{d_n} \right)} \quad \text{et} \quad y_{recepteur} = \frac{\sum_{n=1}^N \left( y_n \times \frac{1}{d_n} \right)}{\sum_{n=1}^N \left( \frac{1}{d_n} \right)}$$

Ce modèle présente certaines limites, notamment parce qu'il ne prend pas en compte la coordonnée d'altitude  $z$ , étant donné l'affichage en 2D sur une carte. Cela peut entraîner des erreurs d'estimation pour les points situés en hauteur ou en contrebas par rapport au récepteur, notamment en environnement intérieur où les signaux doivent traverser plusieurs couches de plancher, réduisant ainsi la force du signal WiFi.

Le modèle utilise un coefficient de pondération fractionnaire de la forme :

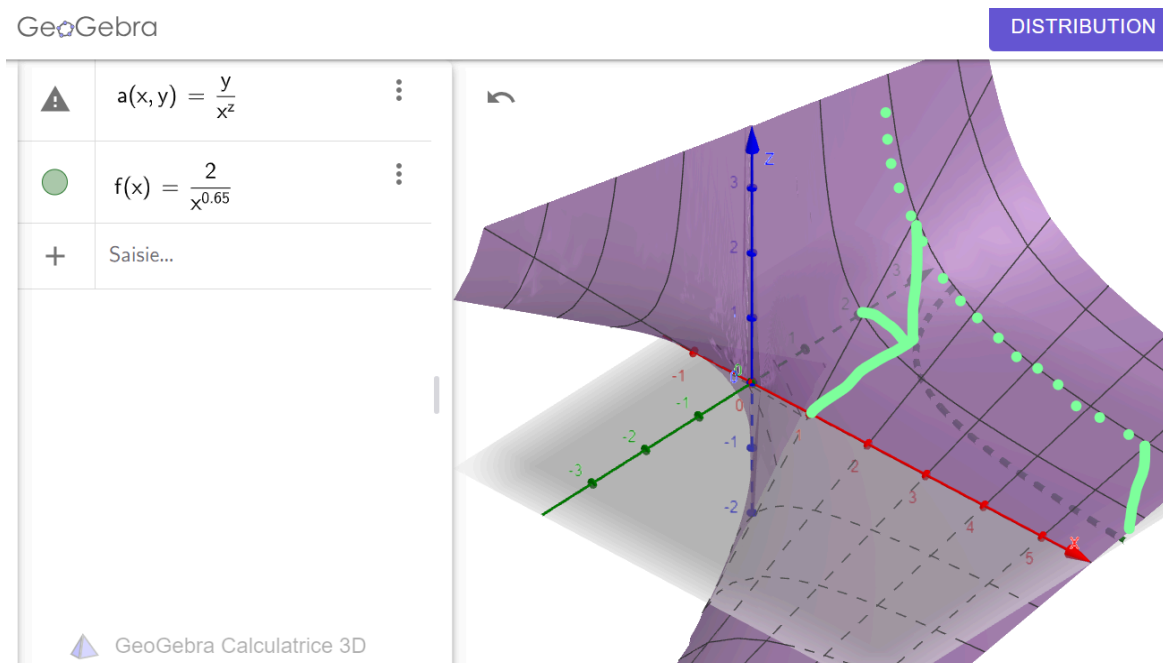
$$\frac{f(d_n)}{g(d_n)} = \frac{1}{(d_n)^1} \text{ où ici } f(d_n) = 1 \text{ et } g(d_n) = (d_n)^1.$$

Cette approche est pertinente, car elle atténue fortement les longues distances par rapport aux courtes, particulièrement utile en contexte urbain dense ou en intérieur. De plus, ce modèle présente l'avantage de ne pas supprimer totalement la pondération des longues distances, même si elle est affaiblie, ce qui permet de tenir compte des relevés avec un RSSI élevé, souvent associés à des distances plus grandes.

Une plus forte pondération veut être donnée aux distances proches de 1 mètre. Nous allons alors chercher un modèle similaire où la pondération prend la forme d'une fonction Constante  $f(d_n) = Cst$  où  $Cst \in \mathbb{R}$  et d'une fonction

$$g(d_n) = (d_n)^v \text{ où } v \in \mathbb{R}$$

Figure 2.1.2 : Support d'analyse des modèles de pondération en 3D sur GeoGebra



Après une analyse des différentes fonctions disponibles dans une représentation 3D, pour observer et comparer les courbes plus aisément, nous avons retenu le modèle suivant :

$$\frac{f(d_n)}{g(d_n)} = \frac{2}{(d_n)^{0,65}} \text{ où ici } f(d_n) = 2 \text{ et } g(d_n) = (d_n)^{0,65}$$

Ce modèle présente l'avantage d'une pente plus douce, garantissant que les distances très courtes, comme 0,4 m, 0,5 m et 1 m, conservent un poids élevé, tout en renforçant également le poids des distances un peu plus éloignées, comme 1 m, 2 m et 3 m, les rendant moins négligeables.

Ainsi l'équation finalement retenue est :

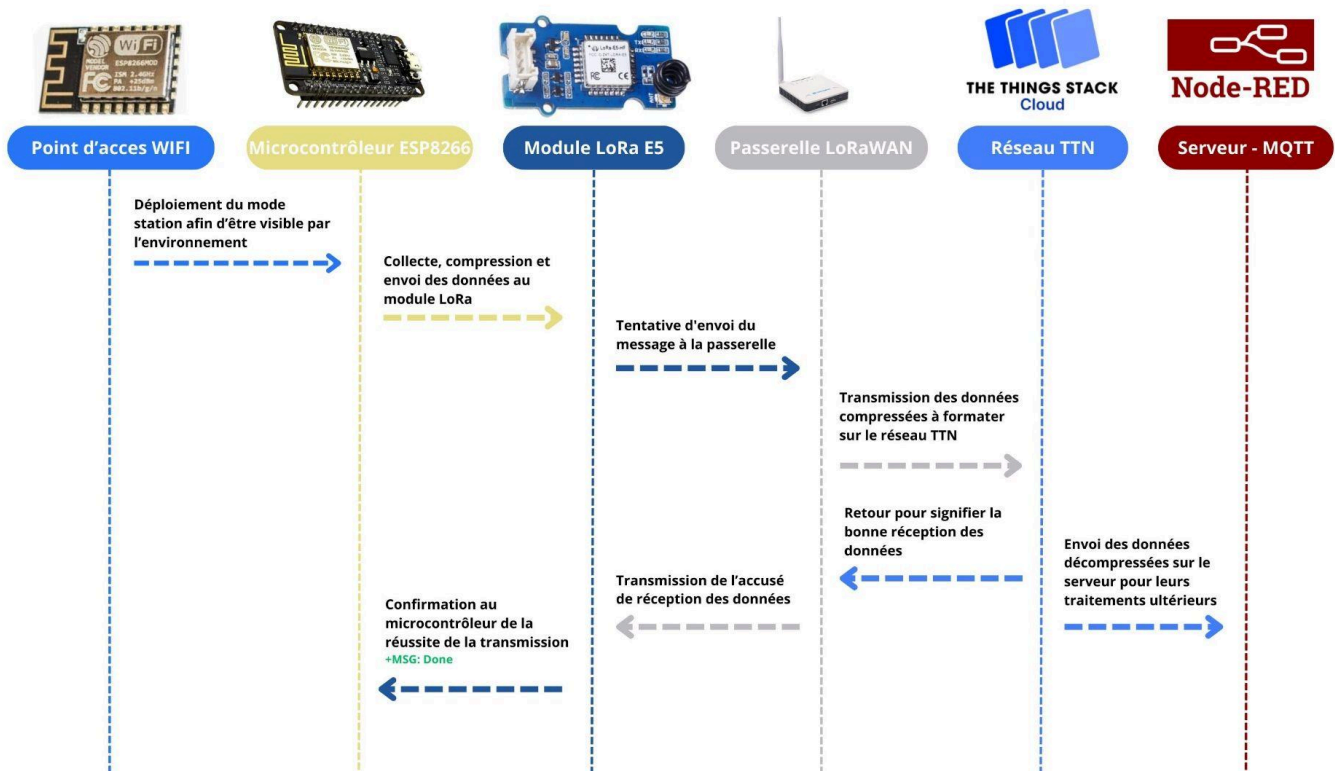
$$x_{recepteur} = \frac{\sum_{n=1}^N \left( x_n \times \frac{2}{(d_n)^{0,65}} \right)}{\sum_{n=1}^N \left( \frac{2}{(d_n)^{0,65}} \right)} \quad \text{et} \quad y_{recepteur} = \frac{\sum_{n=1}^N \left( y_n \times \frac{2}{(d_n)^{0,65}} \right)}{\sum_{n=1}^N \left( \frac{2}{(d_n)^{0,65}} \right)}$$

Ainsi, après avoir mis en place un modèle de trilatération pour estimer les positions, il est essentiel de transmettre ces données via le réseau LoRaWAN, tout en respectant ses contraintes spécifiques en termes de taille et de format de messages. Dans cette prochaine section, nous détaillerons l'architecture choisie pour l'envoi des données, la structuration du message, et les étapes nécessaires pour réceptionner, décoder, et exploiter ces informations efficacement sur le serveur.

## 2.2. Architecture de l'envoi de la donnée

Dans notre projet, le transfert des données de localisation et des valeurs de RSSI (indicateur de force de signal) vers le serveur se fait via le réseau LoRaWAN. Ce réseau impose certaines contraintes, notamment en termes de taille de message. Dans notre cas, chaque message est limité à 51 octets, ce qui impacte la quantité de données que nous pouvons envoyer en une seule transmission. Il a donc été nécessaire d'optimiser le format d'envoi des données pour en maximiser le contenu dans ces conditions.

Figure 2.2.1 : Diagramme de séquence de l'envoi des données collectées sur le serveur.



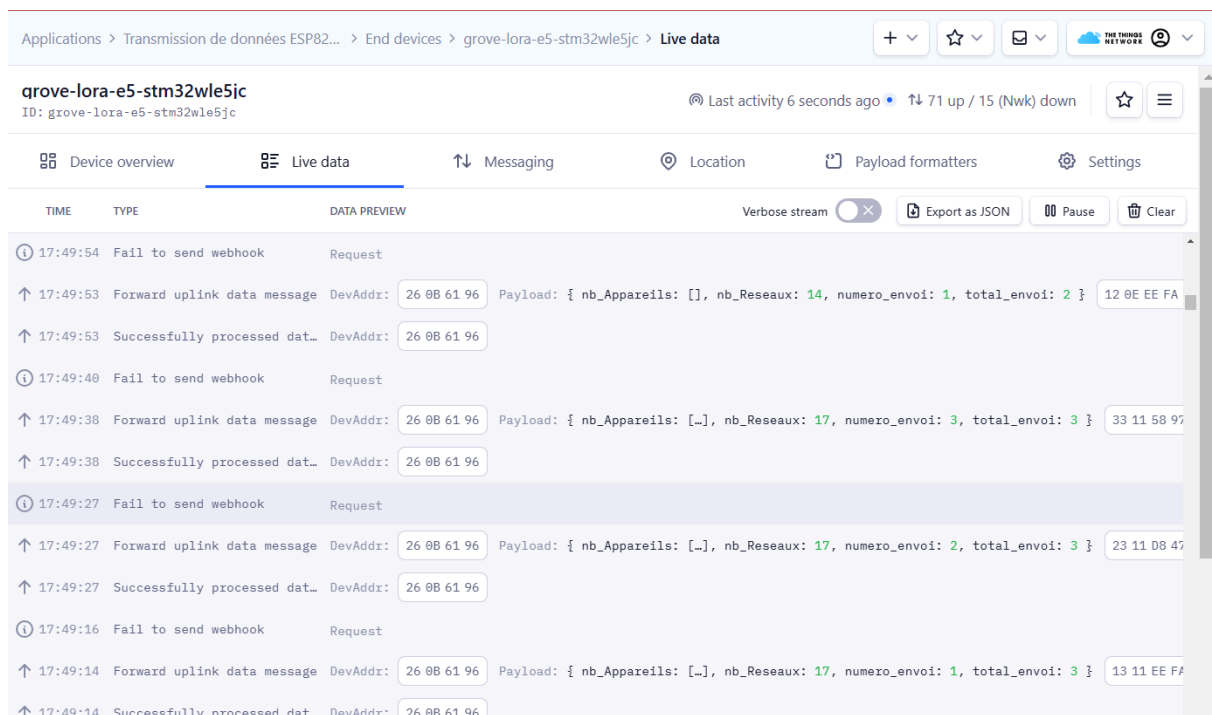
Dans un premier temps, nous avons envisagé un envoi en code ASCII où chaque caractère occuperait 8 bits. Cependant, cette méthode limite la capacité totale de transmission à seulement 51 caractères, ce qui s'est révélé insuffisant pour notre application. Afin d'augmenter la densité de l'information dans chaque message, nous avons opté pour une transmission en code hexadécimal. Dans ce format, chaque caractère occupe seulement 4 bits, doublant ainsi la quantité de caractères par message : nous pouvons envoyer jusqu'à 102 caractères dans les mêmes 51 octets.

Pour maximiser la quantité de données transférées dans ces 51 octets, nous avons conçu une architecture très compacte. En premier lieu, chaque adresse MAC, qui est cruciale pour identifier les appareils détectés dans la base de données, est envoyée sous forme de code hexadécimal et occupe donc 6 octets, contre 12 en ASCII. Ensuite, pour chaque appareil, nous envoyons également la valeur de RSSI, qui est nécessaire pour l'estimation de la distance. Au lieu d'envoyer directement la distance ( nombre à virgule ), nous transmettons uniquement le RSSI sous un format compact de 1 octet. Ce choix permet de simplifier l'envoi et de réduire le nombre d'octets requis, car la conversion du RSSI en distance sera effectuée au niveau du serveur, dans le payload formatter.

Avec cette structure, chaque appareil nécessite donc 7 octets : 6 pour l'adresse MAC et 1 pour le RSSI. Cela permet d'inclure jusqu'à 7 appareils par message, occupant ainsi 49 octets. Les 2 octets restants sont utilisés pour envoyer des informations supplémentaires afin d'assurer le bon déroulement de la transmission et de la reconstruction des données au niveau du serveur.

Nous avons utilisé les deux octets restants pour garantir que les messages sont bien regroupés et reconstitués au niveau du serveur. Ainsi, le deuxième octet indique le nombre total d'appareils détectés, encodé sur 8 bits. Ce choix est important, car nous avons fréquemment une quarantaine d'appareils à transmettre, et un codage sur 4 ou 5 bits serait insuffisant. Le deuxième octet est ensuite encodé avec deux informations : d'une part, 4 bits de poids fort indiquent le numéro de l'envoi dans la séquence, et les 4 bits de poids faible contiennent le nombre total de messages qui seront envoyés. Cela facilite le regroupement des messages fragmentés au niveau du serveur.


*Figure 2.2.2 : Visualisation de la réception des relevés compactés et de leur mise en forme sur le réseau TTN.*





Ainsi, cette approche compacte et organisée permet de maximiser la transmission des données dans la limite des 51 octets, tout en assurant l'intégrité des informations lors de la réception. Une fois les données envoyées, le décryptage et le traitement des informations sont réalisés par le payload formatter sur TTN. Le code du décodeur est disponible dans l'annexe [A.3 Configuration du matériel TTN par étapes](#) en fin d'étape 6. Celui-ci reçoit et interprète les messages encodés en hexadécimal pour les recomposer en objets JavaScript structurés. Chaque adresse MAC est représentée par un nom artificiel, tel que "MAC\_1", "MAC\_2", "MAC\_3" ce qui permet de déterminer facilement la numérotation de chaque appareil reçu. Les objets incluent également les informations cruciales telles que l'adresse MAC de l'appareil, les valeurs RSSI et la distance estimée.

*Figure 2.2.3 : Visualisation de la mise en forme sur le réseau TTN des données appareils.*

 **Latest decoded payload**
×

SOURCE: LIVE DATA Received 31 sec. ago

```

1  {
2    "Donnees_Appareils": [
3      {
4        "Distance_Estime": 3.51,
5        "MAC": "58-97-BD-CD-92-62",
6        "Num": "MAC_8",
7        "RSSI": -74
8      },
9      {
10       "Distance_Estime": 3.51,
11       "MAC": "58-97-BD-CD-92-61",
12       "Num": "MAC_9",
13       "RSSI": -74
14     },
15     {
16       "Distance_Estime": 3.51,
17       "MAC": "58-97-BD-CD-92-63",
18       "Num": "MAC_10",
19       "RSSI": -74
20     }
21   ],
22   "nb_Reseaux": 10,
23   "numero_envoi": 2,
24   "total_envoi": 2
25 }

```

Bien que notre format de transmission atteigne un haut niveau de compacité, il pourrait être optimisé en économisant certains bits, notamment celui indiquant le nombre total de messages, qui peut être déduit du nombre total d'appareils modulo 7. Cela permettrait de libérer des bits pour ajouter par exemple un multiplexeur, permettant ainsi de définir différents formats de payload et de faciliter la réception des données dans divers contextes.



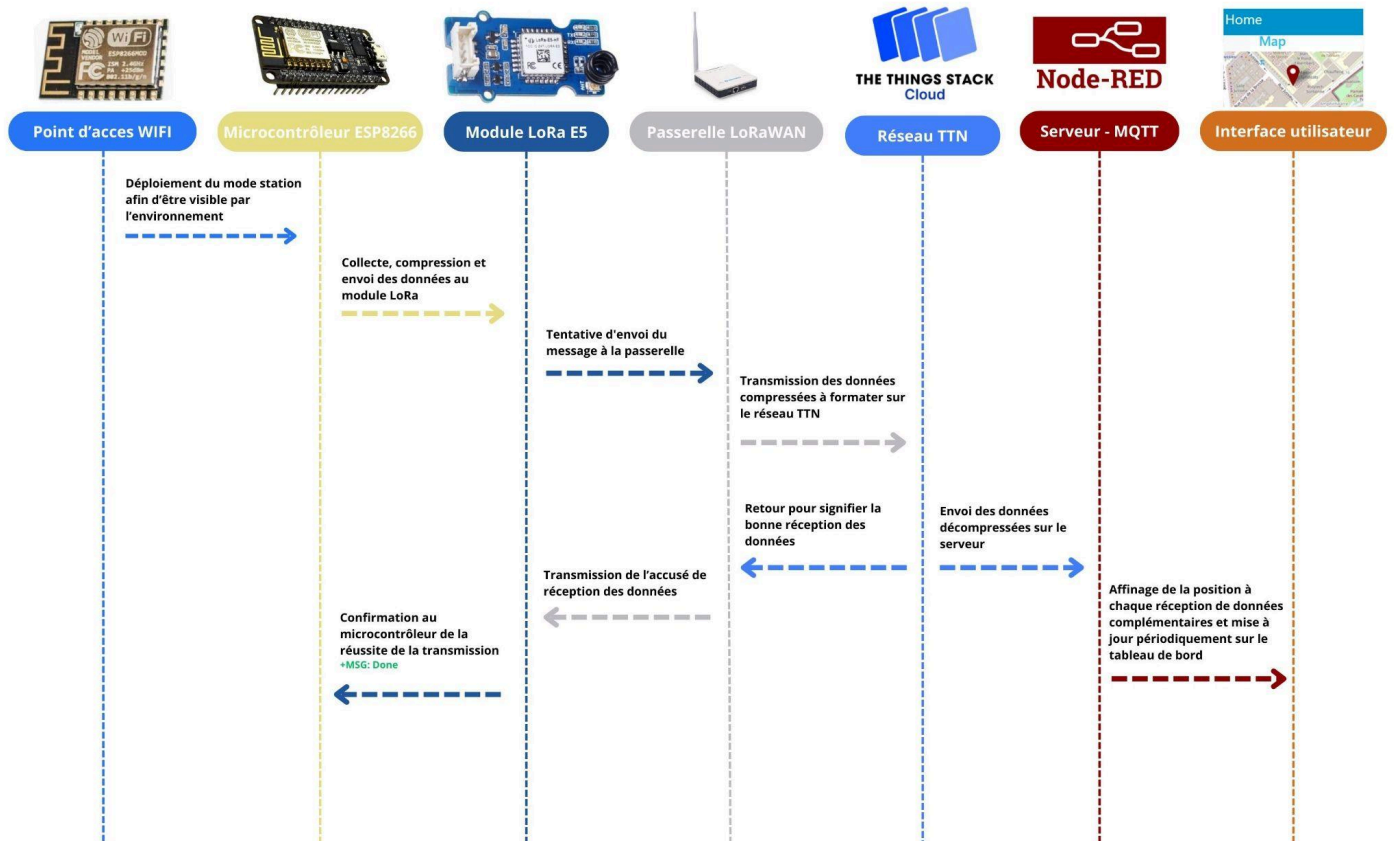
Enfin, pour traiter et afficher les données de positionnement sur notre serveur, nous avons configuré une connexion MQTT via The Things Network TTN. Dans la console TTN, nous avons configuré l'intégration MQTT et utilisé l'API Key comme mot de passe pour la connexion. Ensuite, dans Node-RED, un simple node MQTT In reçoit les messages en provenance de TTN avec l'aide de la source suivante [https://youtu.be/9H6GFXatOCY?si=2tuFL5xH\\_2H9Yp83](https://youtu.be/9H6GFXatOCY?si=2tuFL5xH_2H9Yp83), reprise, complétée et adaptée. En choisissant cette architecture, nous bénéficions d'un format de transmission léger, bien adapté aux réseaux IoT, qui simplifie le transfert et l'affichage des données sur une carte en temps réel.

### 3. Traitement des données sur le serveur et visualisation de la position sur l'interface utilisateur

Pour conclure ce projet, cette dernière partie se concentre sur l'architecture du traitement des données sur le serveur, en décrivant comment celles-ci sont interrogées et visualisées via une interface utilisateur dynamique. Ce processus permet de mettre à jour en continu les informations de localisation et d'afficher, par intervalles de 10 secondes, la position estimée de manière de plus en plus précise au fur et à mesure que le serveur reçoit toutes les données liées à une position.

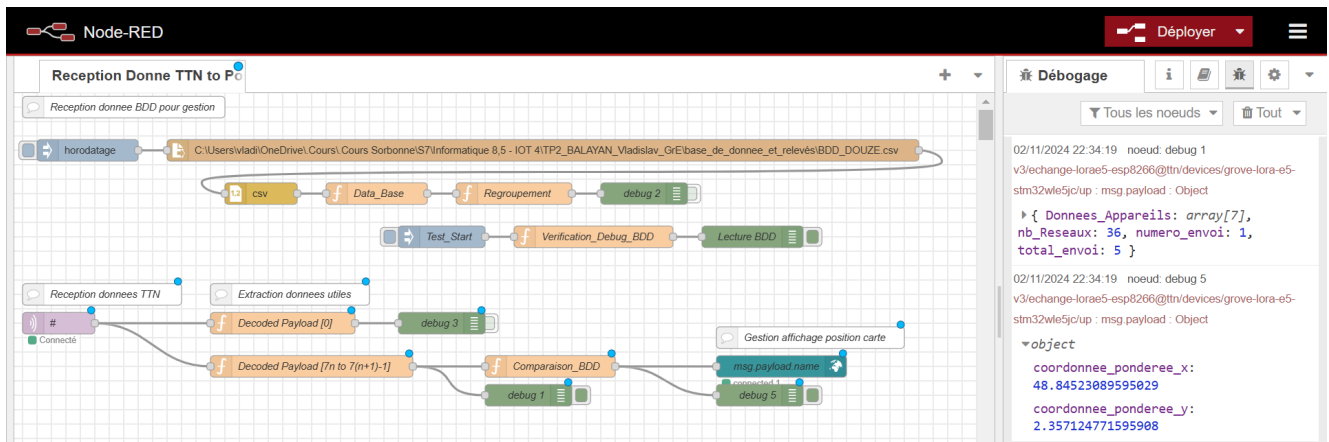
Pour assurer le traitement, le stockage, et la visualisation des données en temps réel, nous avons choisi un serveur Node-RED dont l'installation est disponible à l'annexe [A.1 Installation de Node-RED par étapes](#) . Ce serveur offre une architecture simple et modulaire pour la gestion des flux de données IoT. Node-RED est particulièrement adapté à notre projet en raison de son intégration avec les protocoles de communication comme MQTT, utilisé pour la transmission des données depuis le réseau LoRa via TTN. Node-RED facilite également la connexion à des bases de données et la gestion dynamique des informations de localisation.

Figure 3.1 : Diagramme de séquence de l'envoi des données collectées sur le serveur et sa visualisation sur l'interface utilisateur.



Nous avons configuré Node-RED pour recevoir les paquets de données sous forme de messages JSON, faciles à comprendre pour l'utilisateur. Les données reçues, une fois décompressées, sont comparées à une base de données locale au format CSV, contenant les adresses MAC connues et leurs coordonnées de latitude et de longitude. Cette comparaison permet d'extraire sous forme d'objets les éléments utiles, comme les adresses MAC et les coordonnées, afin de les comparer à la distance des appareils environnants. À chaque réception d'un paquet, les adresses MAC contenues sont recherchées dans cette liste, et les coordonnées de latitude et de longitude correspondantes sont extraites pour chaque appareil détecté. De plus, la distance estimée, transmise depuis TTN, est également récupérée pour le calcul de la position.

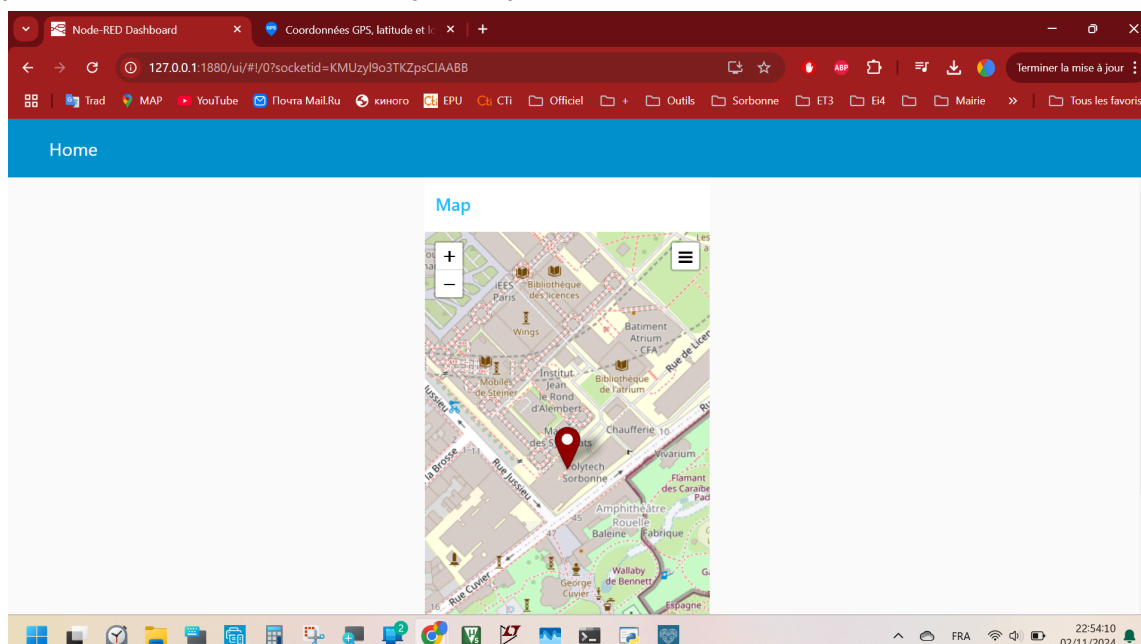
Figure 3.2 : Schéma des flux du serveur Node-RED pour la gestion des données en MQTT.



Le calcul de la position se base sur la formule de trilatération définie précédemment. Les données étant transmises toutes les 10 secondes, chaque paquet contient jusqu'à 7 appareils détectés ( ou moins dans le cas du dernier paquet de la série ). Le calcul de la position est ainsi mis à jour dynamiquement à chaque nouveau paquet reçu, permettant d'affiner progressivement la localisation estimée.

Node-RED met ensuite à jour automatiquement les coordonnées calculées et les transmet à l'interface utilisateur pour la visualisation des coordonnées affinées sur la carte. Grâce à la mise à jour continue des données, la position estimée de l'appareil s'actualise toutes les 10 secondes.

Figure 4.3 : Interface Utilisateur visualisant la communication à l'utilisateur de la position estimé de manière dynamique.



## Conclusion

Ce projet a permis de démontrer le potentiel de localisation d'objets connectés sans recours aux systèmes GPS, en s'appuyant sur la puissance et la disponibilité des signaux WiFi associés à la technologie LoRaWAN. En établissant une relation entre la puissance du signal RSSI et la distance, nous avons réussi à développer un modèle de positionnement flexible, efficace même dans des environnements contraints. L'utilisation du WiFi présente un avantage majeur dans ce contexte : sa large disponibilité dans les espaces urbains et intérieurs, ce qui rend cette technologie particulièrement adaptée pour des solutions de localisation abordables et accessibles.

Bien que la précision des mesures WiFi soit parfois influencée par les obstacles ou les interférences, l'adaptabilité de la technologie permet de maintenir une bonne précision dans de nombreux cas d'usage. Contrairement aux solutions GPS, souvent inefficaces dans les zones couvertes ou denses, le WiFi assure une couverture quasi continue et exploitable pour des applications de positionnement, tout en facilitant la compatibilité avec les infrastructures existantes.

L'architecture modulaire, fondée sur Node-RED et MQTT, renforce la réactivité du système en assurant un traitement en temps réel des données, ce qui permet une visualisation dynamique de la position des appareils. Ce projet illustre également les limites d'une approche basée sur le RSSI. Cependant, pour des applications nécessitant une grande précision, comme le suivi en temps réel dans des environnements industriels complexes, tel qu'un robot convoyeur, il est envisageable d'améliorer la performance en combinant le WiFi avec d'autres technologies, comme l'intelligence artificielle, en utilisant des caméras pour appuyer la précision du robot dans un entrepôt.

En somme, ce travail démontre les avantages de solutions IoT fondées sur le WiFi pour le positionnement en intérieur, offrant une infrastructure pratique, peu coûteuse et fiable. Avec des optimisations continues, ce système pourrait s'adapter à une variété d'applications, ouvrant ainsi des perspectives intéressantes pour le positionnement en temps réel dans des environnements connectés.

## Annexe

### A. Mise en place de l'environnement de travail

#### A.1 Installation de Node-RED par étapes

Pour explorer la communication bidirectionnelle de manière plus visuellement agréable, avec le protocole HTTP REST ( envoie requête serveur ) MQTT ( intermédiaire d'un serveur Broker gérant la réception et renvoie des sujets ) et nous aurons besoin d'un serveur ici Node-RED installable tel que :

1 ) Installation de Node.js depuis :

<https://nodejs.org/dist/v20.17.0/node-v20.17.0-x64.msi>

En cochant l'installation automatique des outils.

2 ) Vérification de l'installation depuis un terminal de commande : `node -v; npm -v`

3 ) Installation de Node-RED depuis le terminal de commande : `npm install -g --unsafe-perm node-red`

4) Lancement depuis le terminal de commande avec : `node-red`

5) Ouverture du lien dans un navigateur sans fermer le terminal de commande : <http://127.0.0.1:1880/>

6) Installation du Dashbord : `Gérer la palette-> installer -> DashBordNodeRed`

#### A.2 Configuration de Arduino IDE pour échanger avec l'ESP

1) Installation de Arduino IDE depuis :

<https://downloads.arduino.cc/arduino-ide/arduino-ide-windows.exe>

2) Dans Arduino IDE, indiquer le module à utiliser depuis `File->Preferences...->` et indiquer dans `« Additional boards manager URLs : »` l'url depuis recherche internet « esp8266 url for arduino ide » :

[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_index.json) / [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json),  
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

sur le site

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/> ou celui de l'ESP8266

3) Installation de la carte ESP depuis `Tools->Board->Boards Manager...` : `« Arduino Nano ESP32 »` et dans notre cas `ESP8266`

4) Après avoir relié le capteur avec le microcontrôleur, **Tools->Board : Node MCU 1.0 (ESP-12E Module) -> esp8266 -> Node MCU 1.0 (ESP-12E Module)**

5) Installation librairie LoRa :

a )

<https://www.gotronic.fr/art-module-lora-e5-grove-113020091-33673.htm>

b ) Fiche technique :

[https://wiki.seeedstudio.com/Grove\\_LoRa\\_E5\\_New\\_Version/](https://wiki.seeedstudio.com/Grove_LoRa_E5_New_Version/)

c) Téléchargement de la librairie au format zip

[Seed\\_LoRa\\_librairie](#) » -> code -> download zip

d ) Inclusion de la librairie téléchargée : **Sketch -> Include Library -> Add .ZIP Library... -> LoRa-E5-main.zip**

5 ) Installation du driver ( si le port n'est pas détecté ) depuis

<https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/> ->

<https://randomnerdtutorials.com/install-esp32-esp8266-usb-drivers-cp210x-windows/> ->

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads> ->

[https://www.silabs.com/documents/public/software/CP210x\\_Windows\\_Drivers.zip](https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip)

6) Connecter le Port : **Tools -> Port...-> COM3**

### A.3 Configuration du matériel TTN par étapes

Pour garantir une communication correcte entre nos données et notre serveur, il est essentiel de configurer le réseau TTN (The Things Network).

1) Création d'une application depuis le réseau communautaire The Things Network :

<https://eu1.cloud.thethings.network/console/applications>

2) Ajout des modules utilisés : **End devices -> Top end devices -> Register end device**

<https://eu1.cloud.thethings.network/console/applications/echange-lorae5-esp8266/devices/add>

3) Selection du **type du module** tel que :

End device type

Input method ⓘ

- ☒ Select the end device in the LoRaWAN Device Repository
- ☐ Enter end device specifics manually

End device brand ⓘ *	Model ⓘ *	Hardware Ver. ⓘ *	Firmware Ver. ⓘ *	Profile (Region) *
<input type="text" value="Seeed Technolog..."/>	<input type="text" value="LoRaWAN Dev Kit"/>	<input type="text" value="1.0"/>	<input type="text" value="1.0"/>	<input type="text" value="EU_863_870"/>

4) Choix de la **bande de fréquence** utilisée ici standard européen :

Frequency plan ⓘ \*

5) Generation d'identifiants réseau aléatoire pour pouvoir se connecter à l'application :

**Provisioning information**

JoinEUI ⓘ \*

<input type="text" value="A0 A0 A0 A0 A0 A0 A0 A0"/>	<input type="button" value="Reset"/>
------------------------------------------------------	--------------------------------------

This end device can be registered on the network

DevEUI ⓘ \*

<input type="text" value="70 B3 D5 7E D0 06 59"/>	<input type="button" value="Generate"/>	1/50 used
---------------------------------------------------	-----------------------------------------	-----------

AppKey ⓘ \*

<input type="text" value="B3 C2 F5 B9 D8 7A 0B 02 B2 D8 74"/>	<input type="button" value="Generate"/>
---------------------------------------------------------------	-----------------------------------------

End device ID ⓘ \*

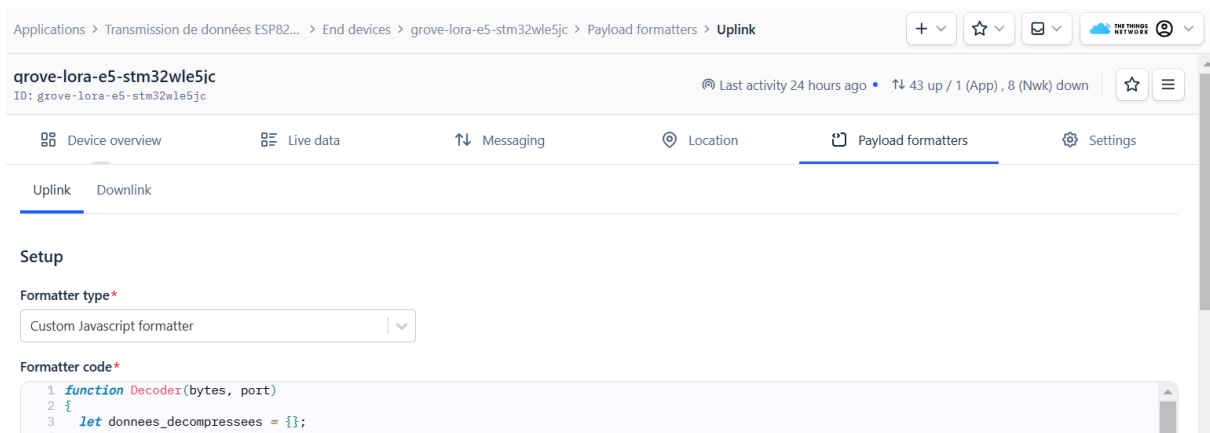


## 6 ) Configuration du **formatage des données** reçues sur le réseau :

Pour visualiser les données reçues depuis le réseau LoRa avec un format compacté, nous allons effectuer une conversion des données brutes. Nous allons effectuer depuis Payload formatters une déconversion rendant les données exploitables et facilement interprétables pour l'utilisateur final et sa transmission ultérieure.

**Applications -> Transmission de données ESP8266 to Grove\_Lora\_E5 ->**

**End devices -> grove-lora-e5-stm32wle5jc -> Payload formatters -> Uplink**



Pour notre utilisation, nous avons opté pour un formatage avec un code en JavaScript disponible ci-après :

```

○○○

function Decoder(bytes, port)
{
    let donnees_decompressees = {};

    //2 premiers octet d information
    let envoi_total = bytes[0];
    let numero_envoi = (envoi_total >> 4) & 0x0F;    //4 bits de poids fort : le 1er quartet
                                                    // contient le numero de l envoi courent
    let total_envoi = envoi_total & 0x0F;    //4 bits de poids faible : le 2eme quartet
                                                    //le nb d envoi total programme

    let nb_envois_restants = total_envoi - numero_envoi;
    let nb_WiFi = bytes[1];

    donnees_decompressees.numero_envoi = numero_envoi;
    donnees_decompressees.total_envoi = total_envoi;
    donnees_decompressees.nb_Reseaux = nb_WiFi;
    donnees_decompressees.Donnees_Appareils = [];

    let nb_de_donnees_a_traiter = (total_envoi-numero_envoi)>0 ? 7 : nb_WiFi%7;

    for (let i = 0; i < nb_de_donnees_a_traiter; i++)
    {
        let start = 2 + i * 7; //Decalage du au 2 premiers octet d information
        let MacHex = bytes.slice(start, start + 6).map(b => b.toString(16).padStart(2,
'0')).join('-').toUpperCase();

        // Interpreter le RSSI en signed 8-bit
        let RssiHex = bytes[start + 6] > 127 ? bytes[start + 6] - 256 : bytes[start + 6];

        // Calculer la distance en fonction du RSSI
        let distance_estime;
        if (RssiHex > -56)
        {
            distance_estime = parseFloat(Math.pow(10, -((RssiHex + 52) / 25.0)).toFixed(2));
//Pour avoir un nombre avec 2 decimales
        }
        else
        {
            distance_estime = parseFloat(Math.pow(10, -((RssiHex + 56) / 33.0)).toFixed(2));
//Pour avoir un nombre avec 2 decimales
        }

        donnees_decompressees.Donnees_Appareils.push
        ({
            Num    : `MAC_${(i + 1) + ((numero_envoi - 1) * (7))}`,
            MAC    : MacHex,
            RSSI   : RssiHex,
            Distance_Estime: distance_estime
        });
    }

    return donnees_decompressees;
}

//Co coder avec Chat GPT

```

## B. Brouillon du Relevé du RSSI en dB en fonction de la distance à la source dans différents environnements

Distance	ESP 327 mur	ESP 324 obstacles	Google Tél environnement à vide :	ESP319 Inconnu
0m	/	-38 qq cm -19 dessus	-31 à qq cm -9 dessus	/
1m	/	-56	-49 à -52	/
2m	/	-65	-56	/
3m	-76	-73	-59	/
4m	-78	-78	-61	/
5m	-82	-81	-63	/
6m	-86	-84	-65	-92
7m	-86 à 92	-87	-66	-90
8m	-86 à 92	-88	-68	-90
9m	-86 à 92	-87 à 88	-70	-91
10m	-86 à 92	-87 à 88	-72 à -64	/
0,1m	/	-41	-27	/
0,2	/	-45	-26 à 35	/
0,3	/	-48	-44	/
0,5	/	-51	-45	/
0,7	/	-54	-46	/
0,9	/	-66	-51	/

