



POLYTECH[®]
SORBONNE



SORBONNE
UNIVERSITÉ

POLYTECH SORBONNE UNIVERSITÉ

INTERNET DES OBJETS
WiFi LOCATION AVEC ESP ET LoRAWAN
RAPPORT

TP2 : Géolocalisation avec WiFi

Élève(s) :

Daniel FERREIRA LARA

Enseignant(s) :

Yann DOUZE

Elyoth HARIAN

Thibault HILAIRE

4 novembre 2024

Table des matières

1	Introduction General	2
1.1	Les protocoles Wi-Fi et LoRaWAN	2
2	Déroulement du TP	4
2.1	Schémas de l'architecture	4
2.2	Les premières étapes	4
2.3	Transmission de données via LoRa	5
2.4	Consommation et persistance des données du TTN	8
2.5	Trilateration	9
2.6	Visualisation des positions	10
3	Conclusion	11

1 Introduction General

Ce rapport couvrira le développement de mon projet de géolocalisation avec LoRaWAN et WiFi aussi succinctement et objectivement que possible. Plusieurs heures ont été passées à essayer de rendre la solution évolutive et facile d'accès, c'est pourquoi je vous conseille vivement d'accéder au site web créé pour visualiser les résultats.

Je pense que le résultat obtenu dans la création d'une solution de bout en bout était satisfaisant, et ce fut une bonne expérience de pratiquer toutes les étapes du développement, de l'optimisation de bas niveau au déploiement de systèmes web. Voici des liens importants pour visualiser les endroits concernant du projet :

1. **Repo GitHub** : [IoT Projects Repo](#)
2. **Site Web** : [Geolocation WiFi](#)

Les étapes suivantes expliqueront les protocoles de communication sans fil utilisés dans le projet, les étapes de développement impliquant les défis rencontrés, l'architecture et la norme de développement, les résultats obtenus. Enfin, il y a une conclusion générale sur l'expérience et sur ce qui a été appris au cours du processus.

1.1 Les protocoles Wi-Fi et LoRaWAN

Le protocole WiFi est largement utilisé depuis les années 2000, la liberté d'utiliser le WiFi sur les appareils mobiles est l'une des raisons de la popularité de cette technologie. Normalisé par la norme IEEE 802.11, il a connu plusieurs évolutions au fil du temps jusqu'à la technologie dite WiFi 6 (802.11ax). Les principales caractéristiques du WiFi sont qu'il permet un significatif data rate sur une distance relativement courte, de l'ordre de 150 mètres, une connexion constituée de points d'accès reliés à Internet par câble ou qui répliquent le signal dans un réseau mesh.

Nous pouvons simplifier le protocole en le ramenant à une architecture dispositif <-> point d'accès, permettant aux appareils mobiles de se connecter à l'internet, en agissant comme un intermédiaire. Nous pouvons le classer dans la norme OSI au niveau le plus bas, en tant que connexion physique (matériel lui-même) et connexion de liaison (adresse MAC). D'un point de vue physique, le WiFi fonctionne sur les fréquences de 2,4 GHz, 5 GHz et 6 GHz, avec une norme de modulation numérique DSSS ou OFDM et un cryptage WPA2/3. En termes de fonctionnement et d'adressage, l'appareil découvre les points d'accès, les authentifie avec un utilisateur et d'un mot de passe et fournit leurs données, puis associe l'adresse MAC au lien du AP.

LoRaWAN (Long Range Wide Area Network) est un protocole qui permet d'atteindre de longues distances tout en consommant peu d'énergie. Conçu pour les appareils IoT dans de vastes zones, il fonctionne sur des fréquences de 868 MHz en Europe. La technologie de modulation LoRa basée sur le CSS, qui permet de communiquer sur des distances allant jusqu'à 15 kilomètres dans les zones rurales et de 2 à 5 kilomètres dans les environnements urbains, est largement utilisée pour envoyer de petites données sur de longues distances.

Les données sont généralement envoyées à partir d'appareils alimentés par des piles qui nécessitent une consommation d'énergie minimale et envoient des données occasionnellement, en quelques dizaines de secondes ou quelques minutes. L'architecture LoRaWAN est basée sur des appareils connectés à des passerelles, qui transmettent des données à des endroits tels que The Things Network. Il est également possible d'établir une connexion lora

point à point entre les appareils, en fait il peut s'agir d'un réseau d'appareils connectés les uns aux autres.

Critère	WiFi	LoRa	LoRaWAN
Débit de données	Jusqu'à 1 Gbps, >1 Gbps (WiFi 6)	0.3 à 50 kbps (selon config)	0.3 à 50 kbps
Distance	100 m en intérieur, 300 m en extérieur	Jusqu'à 15 km rural, 2-5 km ville	Jusqu'à 15 km rural, 2-5 km ville
Fréquence	2,4 GHz et 5 GHz (WiFi 6 : 6 GHz)	433 MHz, 868 MHz (Europe), 915 MHz (Amériques)	433 MHz, 868 MHz (Europe), 915 MHz (Amériques)
Consommation d'énergie	Élevée	Très faible	Très faible
Topologie de réseau	AP/Station	Point-à-point ou étoile	Réseau étoile
Usage principal	Internet, réseaux locaux, streaming, etc.	IoT (sensors, monitoring)	IoT à longue portée, smart city, agriculture, etc.

TABLE 1 – Comparaison des caractéristiques entre WiFi, LoRa et LoRaWAN

Le fonctionnement des réseaux LoRa et LoRaWAN peut être schématisé comme suit :

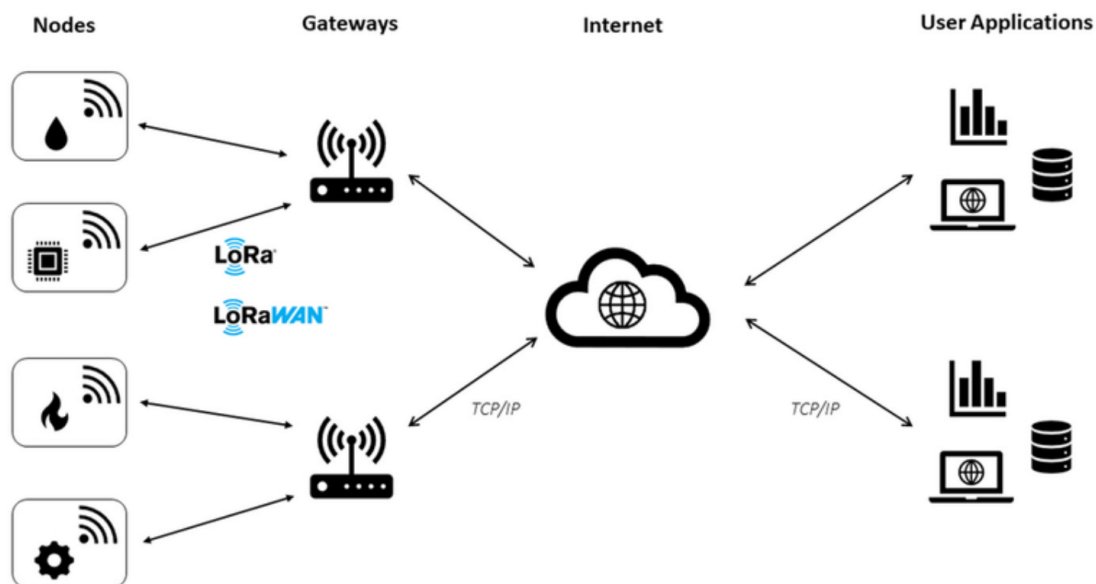


FIGURE 1 – Schéma complet

2 Déroulement du TP

2.1 Schémas de l'architecture

Après avoir compris l'ensemble du développement de la solution, nous pouvons démontrer et simplifier l'architecture dans le diagramme fonctionnel suivant :

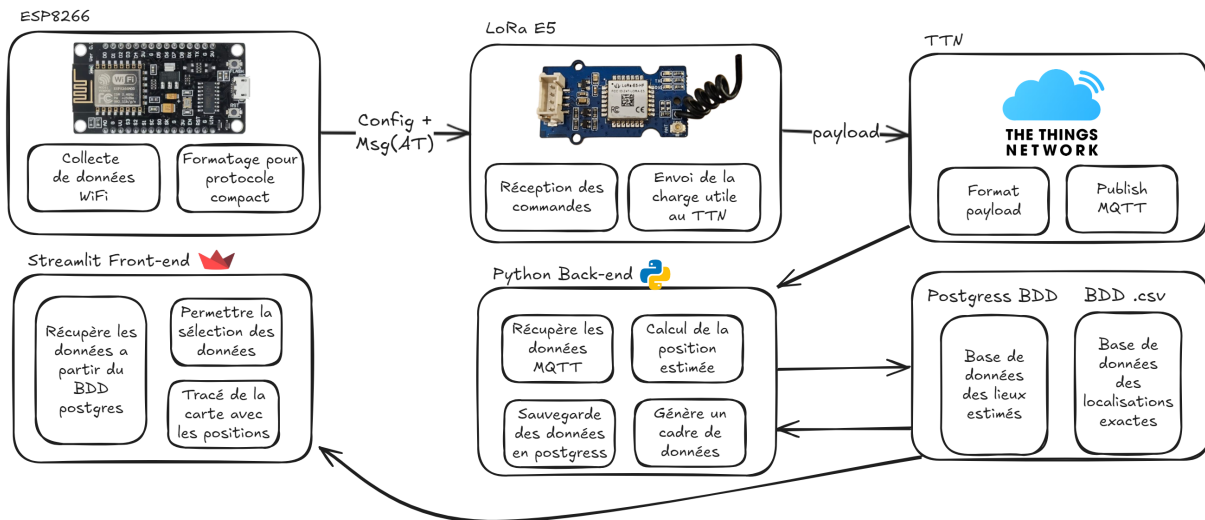


FIGURE 2 – Schéma complet

Je décris ci-dessous les étapes du développement, qui ont nécessité un certain nombre de remaniements et d'optimisations afin d'obtenir une solution aussi modulaire que possible et facile à reproduire et à améliorer.

2.2 Les premières étapes

Les premières étapes (1,2,3) ont consisté à gérer les points d'accès disponibles, en créant des points d'accès dans des pièces spécifiques avec des ESP8266 dispersés dans le bâtiment. Ensuite, j'ai développé le code pour découvrir les points d'accès et estimer leur distance en fonction du RSSI, c'est-à-dire de la force du signal.

$$\text{distance} = 10^{\frac{\text{rssi}_0 - \text{rssi}}{10 \cdot n}}$$

$$n = 2.5$$

$$\text{rssi}_0 = -30 \text{ dB}$$

Pour calculer la distance tant sur l'ESP que sur le serveur python, j'ai utilisé une solution de base consistant à convertir la puissance du signal en une valeur en mètres. Pour ce faire, j'ai pris la distance de 0 mètre comme -30 dB , d'où l'élément n , qui indique le facteur d'atténuation du signal, une valeur que j'ai trouvée raisonnable est de 2,5. Idéalement, il serait intéressant de modéliser une fonction plus complexe pour le calcul, en mesurant l'atténuation pour différents scénarios, mais d'après mon analyse, la formule utilisée est efficace car elle est généraliste et convient à presque tous les cas.

Le résultat final de ces étapes est la création d'une bibliothèque C++ pour gérer la recherche de points d'accès WiFi ([Biblio WiFi](#)). Un des problèmes de la bibliothèque

originale était que les réseaux ne pouvaient pas être triés, et dans notre cas, il est crucial de choisir les réseaux avec le plus de puissance, ce qui signifie qu'ils sont plus proches et qu'ils ont une meilleure précision dans les calculs de position futurs.

J'ai donc créé une méthode pour trier les objets par RSSI et je l'ai ajoutée à la bibliothèque. Elle permet également de générer un JSON avec un nombre limité de réseaux, qui servira de validateur de données dans les étapes suivantes. Voici une image du résultat de l'analyse du réseau WiFi :

```
19:30:40.255 -> 29: Redmi Note 11[(79.43 m), (-78 dBm)] MAC: 0E:6E:E4:C8:66:C9
19:30:40.255 -> 30: ISCD[(89.13 m), (-79 dBm)] MAC: E8:65:D4:72:F9:69
19:30:40.255 -> 31: CONGRES[(89.13 m), (-79 dBm)] MAC: 58:97:BD:07:D6:62
19:30:40.286 -> 32: eduspot[(89.13 m), (-79 dBm)] MAC: 58:97:BD:07:D6:61
19:30:40.286 -> 33: eduroam[(100.00 m), (-80 dBm)] MAC: 00:F6:63:CA:DD:A4
19:30:40.286 -> 34: eduroam[(125.89 m), (-82 dBm)] MAC: 00:F6:63:DD:3C:F4
19:30:40.286 -> 35: eduspot[(141.25 m), (-83 dBm)] MAC: 00:F6:63:DD:3C:F1
19:30:40.286 -> 36: eduspot[(141.25 m), (-83 dBm)] MAC: 00:F6:63:C7:2A:71
19:30:40.286 -> 37: CONGRES[(141.25 m), (-83 dBm)] MAC: E4:AA:5D:E7:F3:11
19:30:40.319 -> 38: CONGRES[(158.49 m), (-84 dBm)] MAC: 00:F6:63:DD:3C:F2
19:30:40.319 -> 39: CONGRES[(158.49 m), (-84 dBm)] MAC: 00:F6:63:C7:2A:72
19:30:40.319 -> 40: scaiteam[(158.49 m), (-84 dBm)] MAC: 58:D9:D5:41:50:32
19:30:40.319 -> 41: PPIAdmin[(158.49 m), (-84 dBm)] MAC: 04:D9:F5:74:E0:B0
19:30:40.319 -> 42: fablabstaff[(177.83 m), (-85 dBm)] MAC: 08:5A:11:28:8A:10
19:30:40.319 -> 43: ISIR[(223.87 m), (-87 dBm)] MAC: E8:65:D4:72:AC:61
19:30:40.351 -> 44: ISIR[(223.87 m), (-87 dBm)] MAC: E8:65:D4:72:A0:D1
19:30:40.351 -> 45: CONGRES[(223.87 m), (-87 dBm)] MAC: 00:F6:63:E6:2E:D2
19:30:40.351 -> 46: eduspot[(223.87 m), (-87 dBm)] MAC: 00:F6:63:E6:2E:D1
19:30:40.351 -> 47: ISIR[(251.19 m), (-88 dBm)] MAC: E8:65:D4:72:A0:D9
19:30:40.351 -> 48: CONGRES[(251.19 m), (-88 dBm)] MAC: E4:AA:5D:AC:41:21
```

FIGURE 3 – Résultat Scan WiFis

2.3 Transmission de données via LoRa

Les premières étapes de la configuration du Lora E5 ont consisté à utiliser le module avec un convertisseur série et à concevoir et tester la chaîne de commandes « AT » nécessaire à la mise en place et à l'envoi de données sur le réseau LoRaWAN. J'ai eu un petit problème de connexion avec le convertisseur série usb, je l'ai résolu en installant le pilote que j'ai trouvé sur le lien référencé comme [1].

Pour les tests, le circuit ci-dessous a été assemblé et à partir de celui-ci, il a été possible de créer un code pilote pour l'ESP afin de commander le système LoRa.

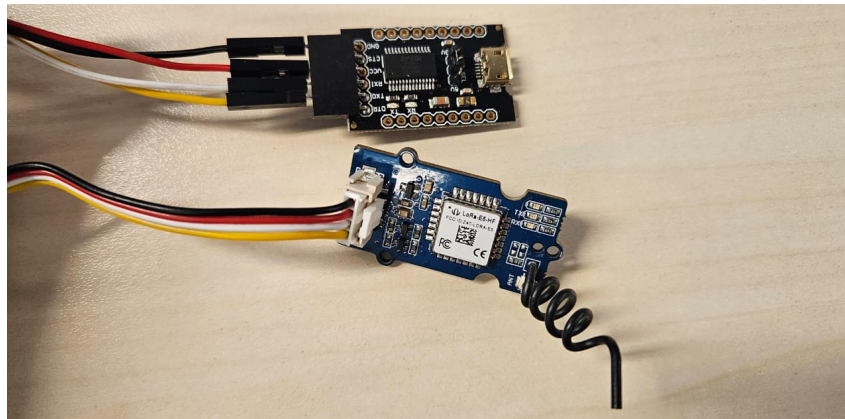


FIGURE 4 – Circuit

```

---- Sent utf8 encoded message: "AT+ID" ----
+INFO: Input timeout
+ID: DevAddr, 32:30:27:B7
+ID: DevEui, 70:B3:D5:7E:D0:06:AE:E5
+ID: AppEui, C4:B2:07:4C:38:5B:7F:6A
---- Sent utf8 encoded message: "AT+DR=EU868" ----
+INFO: Input timeout
+DR: EU868
---- Sent utf8 encoded message: "AT+ID=DevEui,\"70B3D57ED006AEE5\"" ----
+INFO: Input timeout
+ID: DevEui, 70:B3:D5:7E:D0:06:AE:E5
---- Sent utf8 encoded message: "AT+ID=AppEui,\"C4B2074C385B7F6A\"" ----
+INFO: Input timeout
+ID: AppEui, C4:B2:07:4C:38:5B:7F:6A
---- Sent utf8 encoded message: "AT+KEY=APPKEY,\"2BD9D746199C68C2011BA6250B4B5EF2\"" ----
+INFO: Input timeout
+KEY: APPKEY 2BD9D746199C68C2011BA6250B4B5EF2
---- Sent utf8 encoded message: "AT+MODE=LWOTAA" ----
+INFO: Input timeout
+MODE: LWOTAA
---- Sent utf8 encoded message: "AT+DR=DR3" ----
+INFO: Input timeout
+DR: DR3
+DR: EU868 DR3 SF9 BW125K
---- Sent utf8 encoded message: "AT+JOIN" ----
+INFO: Input timeout
+JOIN: Start
+JOIN: NORMAL
+JOIN: Join failed
+JOIN: Done

```

FIGURE 5 – Setup

```

---- Sent utf8 encoded message: "AT+JOIN" ----

+INFO: Input timeout
+JOIN: Start
+JOIN: NORMAL
+JOIN: Join failed
+JOIN: Done
---- Sent utf8 encoded message: "AT+JOIN" ----

+INFO: Input timeout
+JOIN: Start
+JOIN: NORMAL
+JOIN: Network joined
+JOIN: NetID 000013 DevAddr 26:0B:D5:FD
+JOIN: Done
---- Sent utf8 encoded message: "AT+MSG=\"Test\"" ----

+INFO: Input timeout
+MSG: Start
+MSG: Done
|

```

FIGURE 6 – Join

À la suite de ces tests, une autre bibliothèque C++ a été créée pour la gestion des données, la configuration et l'envoi d'informations via LoRa. [Biblio LoRa](#)

Pour compresser les données, j'ai créé le protocole suivant et j'ai également configuré le TTN pour qu'il decode les informations envoyées par le module.

$$\langle N/\text{MAC}_1, \text{RSSI}_1; \text{MAC}_2, \text{RSSI}_2; \dots; \text{MAC}_N, \text{RSSI}_N \rangle$$

Où N est le nombre de réseaux à envoyer

Ce protocole est suffisamment compact puisque le calcul de la trilatération sera effectué par le serveur et que les SSID se trouvent dans la base de données. C'était la manière la plus optimisée d'envoyer l'information avec la garantie qu'elle serait reçue correctement par le réseau TTN. Il y avait une limitation pour chaque débit de données, comme je l'avais déjà fait, et je n'ai donc capté que les N signaux maximaux les plus forts.

Data Rate	Réseaux codés max.
DR1	Impossible
DR2	2
DR3	6
DR4	13
DR5	14

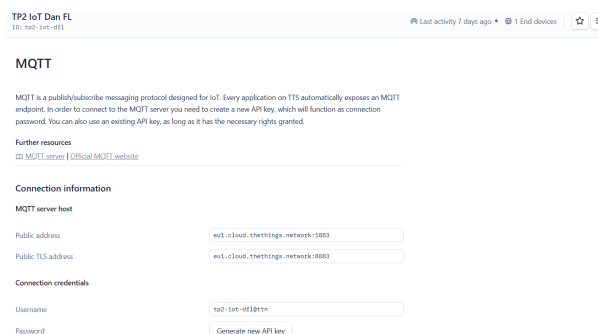
Par conséquent, le réseau TTN decode la charge utile comme suit, après avoir configuré une fonction de décodage personnalisée :


```
▼ decoded_payload: object
  ► Networks: array[1]
    Quantity: 3
```

FIGURE 7 – Décodage Personnalisée

2.4 Consommation et persistance des données du TTN

Pour consommer les données reçues dans le TTN et permettre leur persistance, un serveur MQTT se connecte au projet dans le TTN, puis traite les données et les compare avec la base de données du réseau déjà définie. Il calcule ensuite la position estimée et enregistre enfin toutes ces informations dans une base de données Postgres déployée sur [Vercel](#). Trouvez ci-dessous l'image de l'intégration MQTT :



The screenshot shows the 'MQTT' configuration page in the TTN (The Things Network) interface for a project named 'TP2 IoT Dan FL'. The page includes a description of MQTT, further resources, and a 'Connection information' section with fields for 'MQTT server host', 'Public address', 'Public TLS address', 'Username', and 'Password'. The 'Public address' and 'Public TLS address' fields are both set to 'eu1.cloud.thethings.network:1883'. The 'Username' field is set to 'tp2-iot-efl@ttn'. There is a 'Generate new API key' button next to the password field.

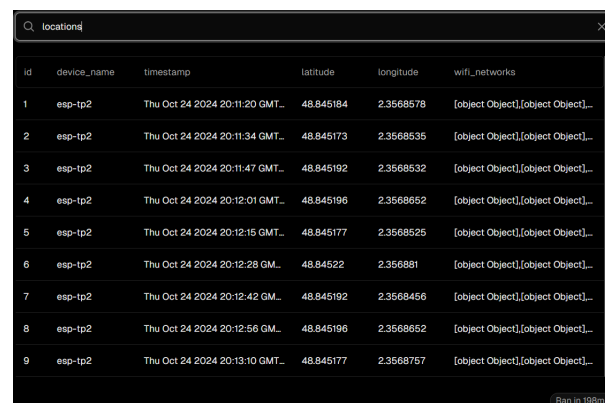
FIGURE 8 – integration MQTT

Le serveur qui calcule la position est abonné au topic principale du MQTT du projet et reçoit le JSON ci-dessous, capturant les informations dont il a besoin pour faire son travail.

```
09/10/2024, 16:49:57  no: debug 1
v3/tp2-iot-dfi@ttn/devices/esp-tp2/up : msg.payload : Object
▼ object
  ▶ end_device_ids: object
  ▶ correlation_ids: array[1]
    received_at: "2024-10-09T14:49:57.959432518Z"
  ▼ uplink_message: object
    session_key_id: "AZJxv19TLjLrLkb+fWpgTA=="
    f_port: 9
    f_cnt: 3
    frm_payload: "PDMvQjY1RjRGQjY2RDZBLC01OT4="
  ▼ decoded_payload: object
    ▶ Networks: array[1]
      Quantity: 3
    ▶ rx_metadata: array[2]
    ▶ settings: object
      received_at: "2024-10-09T14:49:57.753445868Z"
      consumed_airtime: "0.071936s"
    ▶ version_ids: object
    ▶ network_ids: object
```

FIGURE 9 – JSON reçu par le serveur

Enfin, après le processus de trilatération, le serveur accède à la base de données postgres et enregistre les informations qu'elle contient à l'aide d'une commande SQL, ce qui donne la base de données suivant :



id	device_name	timestamp	latitude	longitude	wifi_networks
1	esp-tp2	Thu Oct 24 2024 20:11:20 GMT...	48.845184	2.3568578	[object Object],[object Object]...
2	esp-tp2	Thu Oct 24 2024 20:11:34 GMT...	48.845173	2.3568535	[object Object],[object Object]...
3	esp-tp2	Thu Oct 24 2024 20:11:47 GMT...	48.845192	2.3568532	[object Object],[object Object]...
4	esp-tp2	Thu Oct 24 2024 20:12:01 GMT...	48.845196	2.3568652	[object Object],[object Object]...
5	esp-tp2	Thu Oct 24 2024 20:12:15 GMT...	48.845177	2.3568525	[object Object],[object Object]...
6	esp-tp2	Thu Oct 24 2024 20:12:28 GMT...	48.84522	2.356881	[object Object],[object Object]...
7	esp-tp2	Thu Oct 24 2024 20:12:42 GMT...	48.845192	2.3568456	[object Object],[object Object]...
8	esp-tp2	Thu Oct 24 2024 20:12:56 GMT...	48.845196	2.3568652	[object Object],[object Object]...
9	esp-tp2	Thu Oct 24 2024 20:13:10 GMT...	48.845177	2.3568757	[object Object],[object Object]...

FIGURE 10 – BDD Postgres

Le processus de trilatération est extrêmement important pour la solution, peut être trouvé dans le fichier de [trilatération](#), et sera décrit ensuite.

2.5 Trilateration

Avant d'effectuer la trilatération, j'ai joint les wifis trouvés à ceux de la base de données, afin d'utiliser le RSSI des données trouvées et de compléter les informations de latitude, de longitude et d'altitude. Il a été nécessaire d'étendre la base de données initiale fournie afin de mieux calculer les points à l'intérieur du campus de l'UPMC. Malheureusement, le calcul de l'altitude a été compromis car les nouvelles données ne contiennent pas l'altitude, de sorte que seules la latitude et la longitude ont été calculées.

Après avoir mesuré les distances et obtenu les emplacements des points d'accès, j'ai d'abord calculé le point estimé à partir des moyennes des positions des points d'accès.

$$\text{Moyenne initiale} = \left(\frac{\sum \text{Latitude}_i}{n}, \frac{\sum \text{Longitude}_i}{n}, \frac{\sum \text{Altitude}_i}{n} \right)$$

Cependant, la simple moyenne ne donne pas la position réelle, car la puissance varie entre les points. J'ai donc créé une fonction de correction d'erreur en utilisant la méthode des moindres carrés.

$$E(x, y, z) = \sum_i \left(\sqrt{(x - \text{Latitude}_i)^2 + (y - \text{Longitude}_i)^2 + (z - \text{Altitude}_i)^2} - d_i \right)^2$$

Enfin, j'ai utilisé la fonction de minimisation pour trouver la position qui minimise l'erreur entre toutes les mesures

$$(x, y, z) = \arg \min_{(x,y,z)} E(x, y, z)$$

En conséquence, la position estimée est un vecteur de 3 positions, dans lequel l'altitude n'est pas prise en compte parce que les données de la colonne dans la base de données sont erronées, de sorte que nous n'utilisons que la latitude et la longitude.

2.6 Visualisation des positions

La dernière étape du projet a consisté à créer un serveur web en Python avec une interface frontale pour la visualisation des données. Ce serveur est complètement autonome et disponible 7 jours sur 7. Le site a été développé sur de la bibliothèque python [Streamlit](#), avec une recherche constante dans la base de données pour obtenir les mises à jour des appareils. L'interface dispose d'un cache pour sauvegarder les données jusqu'à ce qu'elles soient modifiées dans la base de données, vous permet de sceller des appareils, de sélectionner une heure spécifique pour capturer un appareil donné et une plage de dates pour filtrer les données.

Une fois les données correctement sélectionnées, la solution affiche à l'écran la position estimée de l'appareil et les positions des points d'accès, le tout sur une carte interactive. En ce qui concerne les problèmes rencontrés dans le développement, il est impossible de connecter un service MQTT comme backend intégré à Streamlit, il y a donc deux serveurs qui tournent séparément. J'ai effectué un test ostensible et je suis arrivé à la même conclusion que dans l'article [2].

Vous trouverez ci-dessous une image de la solution finale, je vous conseille vivement de l'essayer, il suffit de cliquer sur l'image.

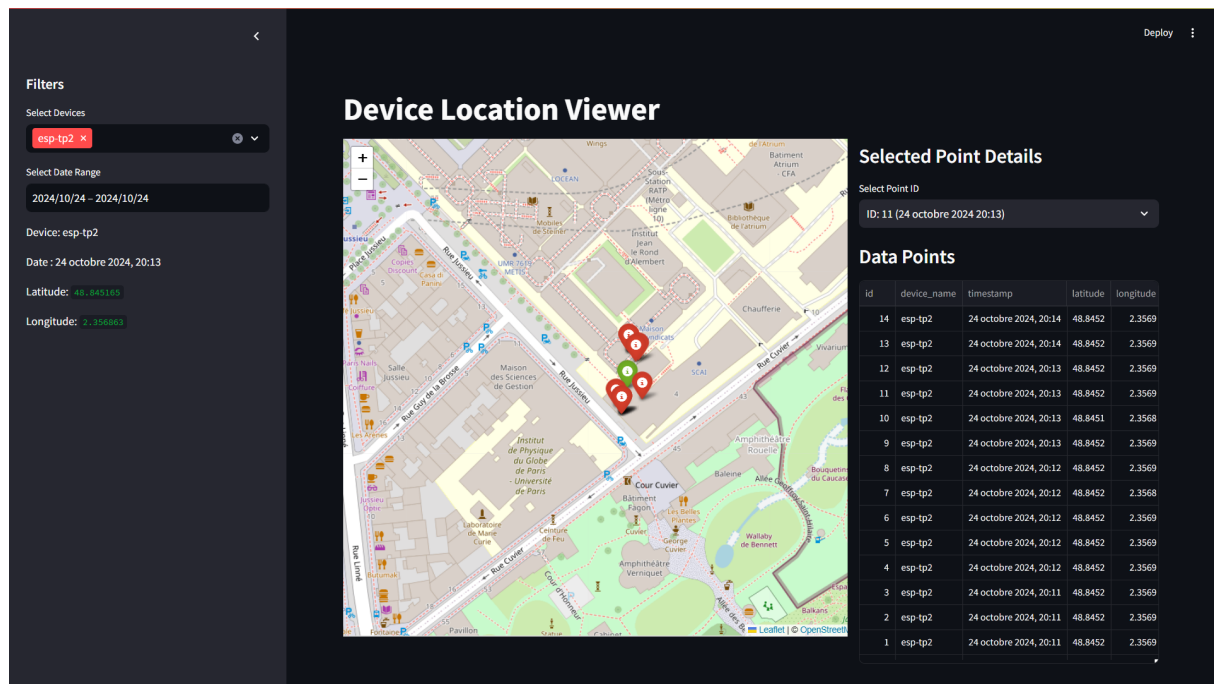


FIGURE 11 – Interface d'utilisateur

3 Conclusion

Ce projet de géolocalisation utilisant LoRaWAN et WiFi a permis de développer une solution robuste, évolutive et accessible. Les différentes étapes et les différents niveaux d'abstraction ont permis d'acquérir une grande expérience professionnelle, en travaillant sur les microprogrammes, la gestion des bases de données, le déploiement de systèmes web back-end et front-end.

C'était une excellente occasion de mettre à l'épreuve les modèles de développement de logiciels que j'ai appris au cours de ma formation, ainsi que l'architecture logicielle, afin de trouver un bon moyen de résoudre la question proposée.

En plus de travailler avec un large éventail de technologies, j'ai également dû relever un certain nombre de défis pour rendre la solution cohérente et modulaire, car la codépendance des étapes a rendu le processus de modularisation plus complexe, nécessitant un certain remaniement dans chaque module afin d'obtenir le bon résultat obtenu.

L'architecture modulaire et l'implémentation des bibliothèques C++ facilitent la réutilisation et l'adaptation de cette solution à d'autres solutions, par exemple, certains des modules développés ont été réutilisés dans le TP3, comme la bibliothèque de gestion du module LoRa E5. L'intégration de l'interface Streamlit pour la visualisation en temps réel permet de suivre efficacement la localisation des appareils, sans avoir à exécuter de commandes. Il suffit qu'un appareil esp connecté à un réseau LoRa se connecte correctement au TTN, à l'aide du module intégré, pour que le pipeline se mette en place.

Références

- [1] Campbell Scientific. Pilote usb vers série (jeu de puces ftdi). [Campbell Scientific](#), 2024. Consulté le 4 novembre 2024.
- [2] Communauté Discuss Streamlit. Streamlit avec paho mqtt - est-ce possible ? [Discuss Streamlit](#), 2024. Consulté le 4 novembre 2024.