

Rapport

TL Système

Tracker GPS LoRa

BÉLIS Pierre
MJABRI Sofiane
BARON Paul

Janvier 2019

Sommaire

Introduction

I - Contexte

- 1) État initial du projet
- 2) Interface utilisateur
- 3) Objectifs

II - Ajout de fonctionnalités

- 1) Position de l'utilisateur
- 2) Fonction recherche
- 3) Mesure de la batterie

III - Amélioration de la portée

- 1) Paramètres de communication LoRa
- 2) Choix du module LoRa
- 3) Choix de l'antenne
- 4) Test en extérieur
- 5) Test en intérieur

IV - Réduction de la consommation

- 1) Améliorations mises en place
- 2) Autonomie estimée

Conclusion

Introduction

Dans le cadre du TL Système, nous continuons notre projet de balise de géolocalisation commencé lors de la mineure IoT. L'objectif de cette balise est de pouvoir localiser un animal de compagnie. Pour cela, elle communique avec une station de base fixe et la position mesurée est stockée sur le cloud.

Nous souhaitons donc améliorer notre dispositif en augmentant sa portée, sa compacité et son autonomie. Nous ajouterons également de nouvelles fonctionnalités.

Ce TL s'inscrit dans notre parcours car il lie des notions de télécommunications et d'électronique embarquée.



I - Contexte

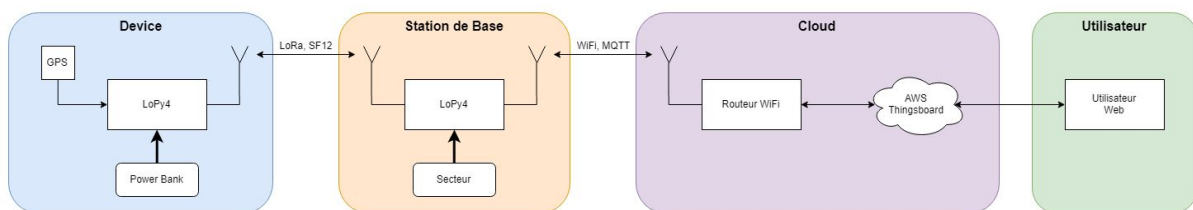
Nous allons détailler ici les éléments issus de notre précédent prototype ainsi que les améliorations souhaitées.

1) État initial du projet

Nous avons repris le projet que nous avons commencé en Mineure IoT. Le dispositif était déjà fonctionnel mais il y avait des aspects à améliorer.

La balise, ou device, est basée sur un LoPy4 de Pycom. Un GPS y est connecté en UART et l'alimentation est assurée par une power bank solaire.

Le device envoie périodiquement la position mesurée par le GPS à la station de base, également constituée d'un LoPy4 qui relaie ces informations, via MQTT sur un réseau WiFi, à un serveur AWS qui héberge un broker MQTT ainsi qu'un serveur HTTP gérant l'interface utilisateur. Ce service est ThingsBoard.



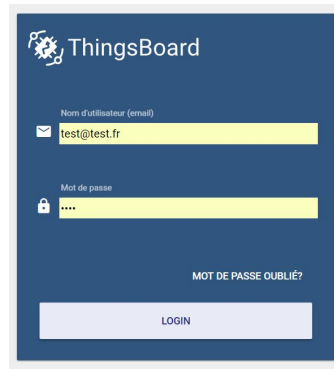
Architecture initiale du système

2) Interface utilisateur

Nous continuerons à utiliser l'interface ThingsBoard pour la poursuite de ce projet. Ce service assure une interface web à l'utilisateur, un système d'authentification, différents widgets pour mettre en forme les données stockées dans une base de données SQL. Il communique avec les différents objets connectés grâce à différents protocoles, nous utiliserons cependant uniquement le broker MQTT inclus.

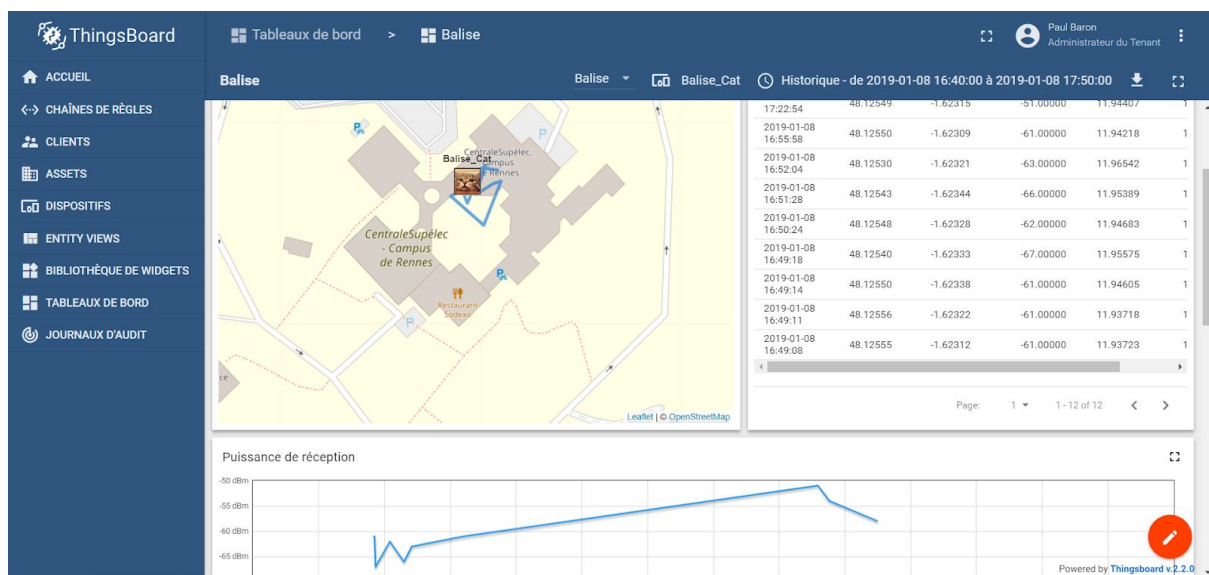
Ce service est hébergé sur une instance Amazon Web Service de taille minimale. Elle est gratuite durant un an. Il est possible par la suite d'héberger le service ThingsBoard sur tout autre serveur d'hébergement, ou même sur une Raspberry Pi connectée à Internet.

Nous avons choisi ce service car l'un d'entre nous avait déjà de l'expérience sur son utilisation. De plus, il est libre et open-source.



Page d'authentification de ThingsBoard

Nous pourrions ainsi afficher en ligne une carte avec la position de la balise et son historique sur une période réglable, une table des mesures à vocation de debug, différentes courbes montrant l'évolution de paramètres variés. Nous ajouterons également d'autres fonctionnalités détaillées par la suite.

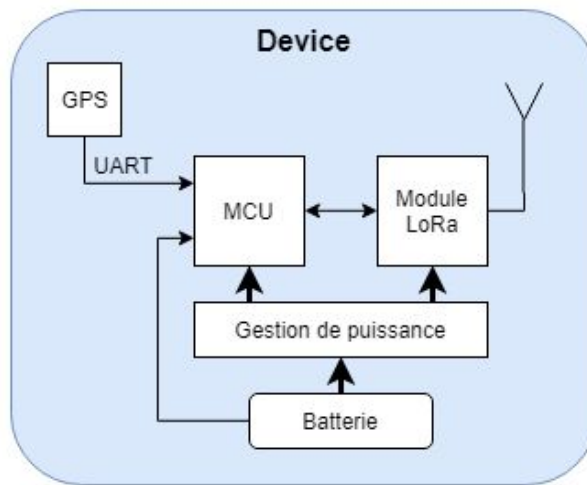


Aperçu de l'interface de la balise

3) Objectifs

Les objectifs de ce TL sont d'améliorer le dispositif afin de diminuer l'encombrement, le coût, augmenter la portée et diminuer la consommation de la balise GPS.

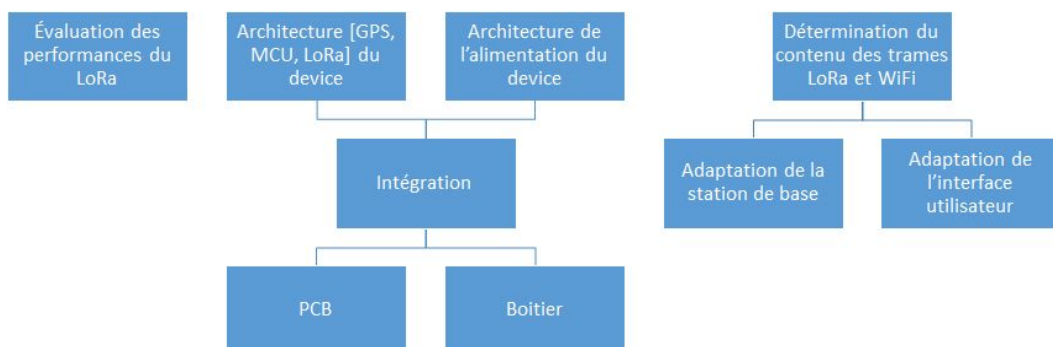
Pour cela, nous allons modifier l'architecture du device.



Architecture désirée de la balise

Nous ajouterons également une fonctionnalité permettant de calculer la distance entre la balise et la station de base et nous implémenterons une fonction “Recherche” permettant de localiser rapidement le dispositif.

Nous souhaitons également concevoir un circuit imprimé et modéliser un boîtier à imprimer afin d’obtenir un prototype final complet



Organisation initiale pour la réalisation du projet

II - Ajout de fonctionnalités

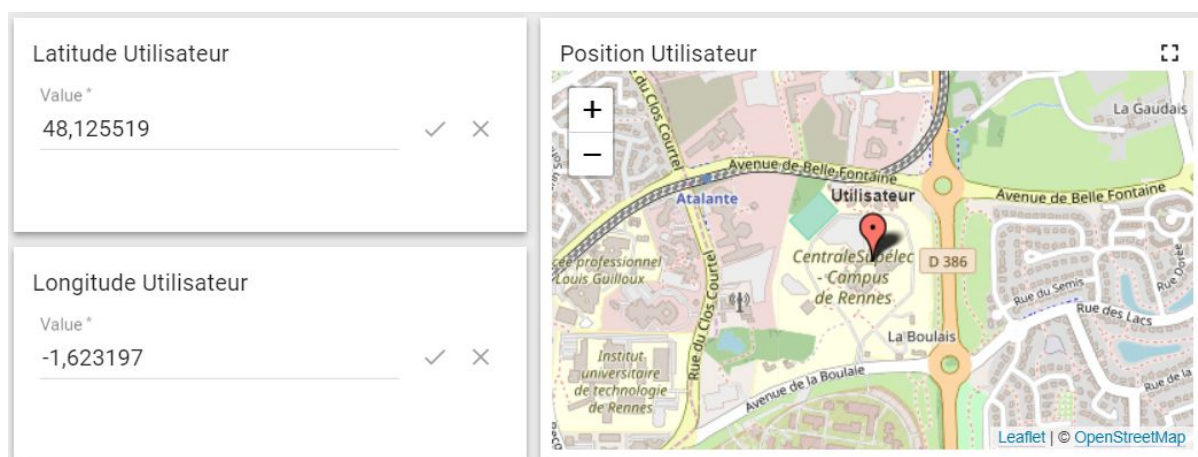
L'ajout de fonctionnalités répond à un double objectif : rendre l'application plus complète et permettre une meilleure gestion de la consommation au niveau du device. Pour ce faire, nous avons ajouté la possibilité de localiser l'utilisateur (et de connaître la distance avec le device), une fonction recherche activable manuellement, et l'indication du niveau de batterie du device.

L'ensemble des calculs se fait au niveau de la station de base pour réduire la consommation du device.

L'échange d'informations entre la station de base et le cloud se fait grâce à la fonctionnalité publish/subscribe du protocole MQTT qui permet de "s'abonner" à des paramètres pour avoir accès aux modifications effectuées par l'utilisateur, et de publier les données de certains paramètres.

1) Position de l'utilisateur

Depuis l'interface utilisateur Thingsboard, nous avons ajouté la possibilité de renseigner manuellement la latitude et la longitude de la station de base de manière à pouvoir localiser l'utilisateur sans modification du code de la station.



The screenshot displays the 'Position Utilisateur' section in the Thingsboard interface. On the left, there are two input fields: 'Latitude Utilisateur' with a value of 48,125519 and 'Longitude Utilisateur' with a value of -1,623197. Both fields have a 'Value *' label and a small 'x' icon for clearing the input. On the right, there is a map titled 'Position Utilisateur' showing a red pin marker at the 'Centrale Supélec - Campus de Rennes'. The map includes street names like 'Avenue de Belle Fontaine', 'Rue du Clos Courtel', and 'Rue des Lacs'. The map is powered by Leaflet and OpenStreetMap.

Champs à remplir pour positionner la station et aperçu sur une carte

Ces informations sont nécessaires pour le calcul en temps réel de la distance entre l'utilisateur (u) et le device (d), qui se fait au niveau de la station de base par la formule de distance orthodromique :

$$d = 60 \arccos(\sin(\text{lat_u})\sin(\text{lat_d}) + \cos(\text{lat_u})\cos(\text{lat_d})\cos(\text{long_u} - \text{long_d}))$$

Cette formule est adaptée pour le calcul de petites distances, cadre d'utilisation le plus courant de l'application.

L'évolution de la distance entre l'utilisateur et le device s'affiche ensuite sous forme de courbe au niveau de l'interface utilisateur

2) Fonction recherche

Une autre nouveauté et l'ajout d'une fonction "recherche", modélisée par un bouton au niveau de l'utilisateur, et un indicateur lumineux qui s'allume lorsque la recherche est activée:



Bouton d'activation de la fonction "Recherche" et témoin lumineux

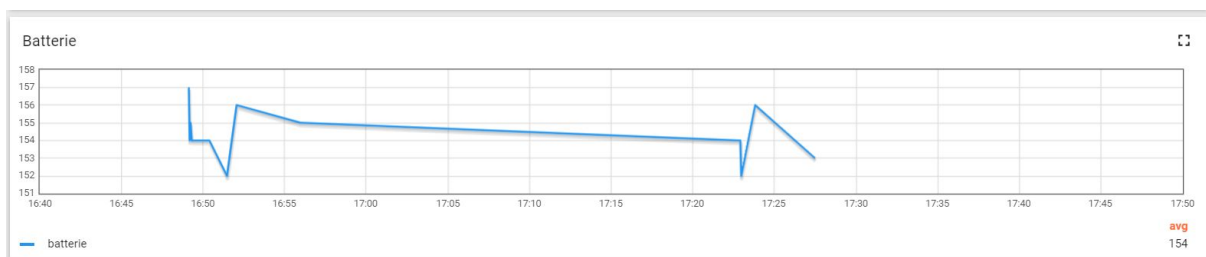
Lorsque l'utilisateur décide d'activer la recherche, la station de base et le device communiquent avec des délais plus rapides, permettant ainsi un accès plus rapide et fréquent à la position du device.

A titre d'information, lorsque la recherche est activée, la station de base demande au device un délai de 5 secondes, contre 10 à 1080 secondes suivant les cas d'utilisation lorsque cette fonction n'est pas activée.

3) Mesure de la batterie

Dans l'objectif de réduire la consommation du device, il est utile d'avoir accès au niveau de batterie restant. Cette information est communiquée par le device à la station de base en LoRa sous forme de valeur normalisée comprise entre 0 et 255 (valeurs respectives de charge nulle et de pleine charge).

Cette valeur est ensuite utilisée par la station de base pour modifier le délai d'envoi des informations du device suivant l'état de la batterie. Elle est également publiée sur Thingsboard et il est possible d'observer son évolution sous forme de courbe :



Évolution du niveau de batterie (modifié via un potentiomètre lors des tests)

III - Amélioration de la portée

Une difficulté majeure rencontrée lors du projet effectué en mineure IoT était la portée. En effet, lors du test effectué avec la station de base en intérieur, nous recevions un signal jusqu'à une distance de 400m (au niveau de la station de métro).

Nous avons essayé d'améliorer la portée de communication en jouant sur plusieurs facteurs, les paramètres de communication, le choix du module LoRa, ainsi que le choix des antennes pour le device et la station de base.

Deux tests ont ensuite été effectués, un en extérieur dans les conditions les plus proches possibles du LOS, et un en intérieur avec de mauvaises conditions météorologiques.

1) Paramètres de communication LoRa

La portée d'une communication LoRa est principalement déterminée par la bande passante, la puissance d'émission, et le spreading factor (SF), aussi appelé facteur d'étalement. Ce dernier paramètre, dont la valeur est comprise entre 7 et 12, augmente la portée en contrepartie d'une diminution du débit (et donc une plus grande consommation).

$$Rb = SF \times \frac{BW}{2^{SF}} \times CR$$

Rb = débit binaire utile, SF = Spreading Factor, BW = Bandwidth, CR = Coding Rate

L'intervalle de bande passante disponible le plus faible possible, 125 kHz, a été choisi. Enfin, le facteur de puissance le plus élevé, 14, a été retenu.

Les paramètres de communication LoRa choisis sont donc les suivants:

SF: 12

Tx Power : 23dBm

BP : 125 kHz

Preamble: 4

Freq : 868.85 MHz

Coding Rate: %

Débit : 293 b/s

2) Choix du module LoRa

Le choix du module devait répondre à plusieurs objectifs:

- une forte compacité permettant de diminuer la taille du boîtier de manière à rendre le dispositif fixable sur un animal
- une puissance d'émission plus importante que le LoPy4 grâce à un Power Amplifier plus puissant que celui normalement utilisé pour LoRa permettant d'atteindre une puissance d'émission de 23 dBm au lieu de 14 dBm.
- un coût réduit

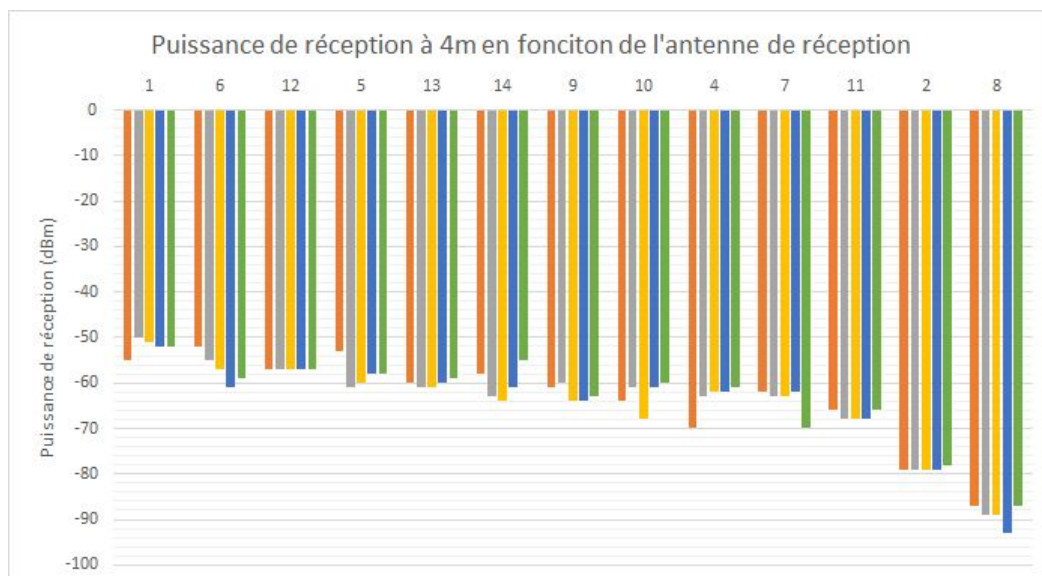
Dans ces conditions, parmi les modules disponibles, c'est le modèle RFM96W qui a été retenu.



Module LoRa RFM96W

3) Choix des antennes

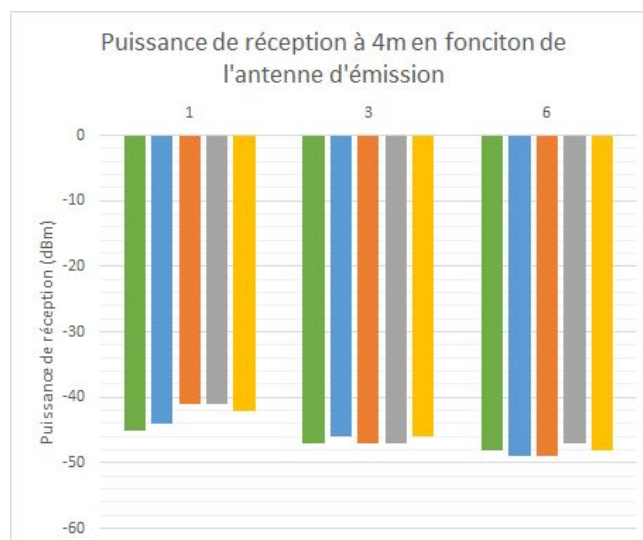
Pour le choix des antennes, il y avait deux critères : le gain et la taille (pour l'antenne de la balise). Nous avons fait des tests pour comparer le gain des différentes antennes que nous avons à disposition. Nous avons choisi la combinaison d'antennes qui assure un gain maximum pour permettre une augmentation de la portée.





Antenne n°2 (à gauche) à n°13 (à droite)
 n°1 : Antenne de base fournie avec le LoPy4
 n°14 : Antenne WiFi

Une fois l'antenne de réception choisie, nous avons réalisé le même test sur l'antenne d'émission en comparant l'antenne utilisée pour les tests de réception aux 2 meilleures mesurées précédemment.



La connectique des antennes nous a posé problème. Les feeders utilisés peuvent être sources de pertes importantes dans la puissance transmise.

4) Test en extérieur

Afin de tester la portée du dispositif, nous avons effectué un test dans un champ sans encombrement entre la balise et la station de base. Nous avons perdu le signal après une distance de 1,3 km et une perte de ligne direct à cause d'une colline après 600m. Le signal est perdu lorsque sa puissance de réception chute sous -140 dBm. La mesure de position est alors très précise, de l'ordre du mètre.



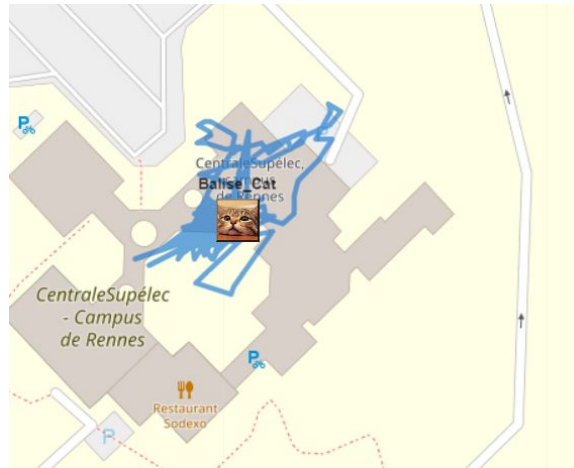
Parcours effectué lors du test, mesure de la distance maximale atteinte et évolution de la puissance reçue



Balise en déplacement, et station de base

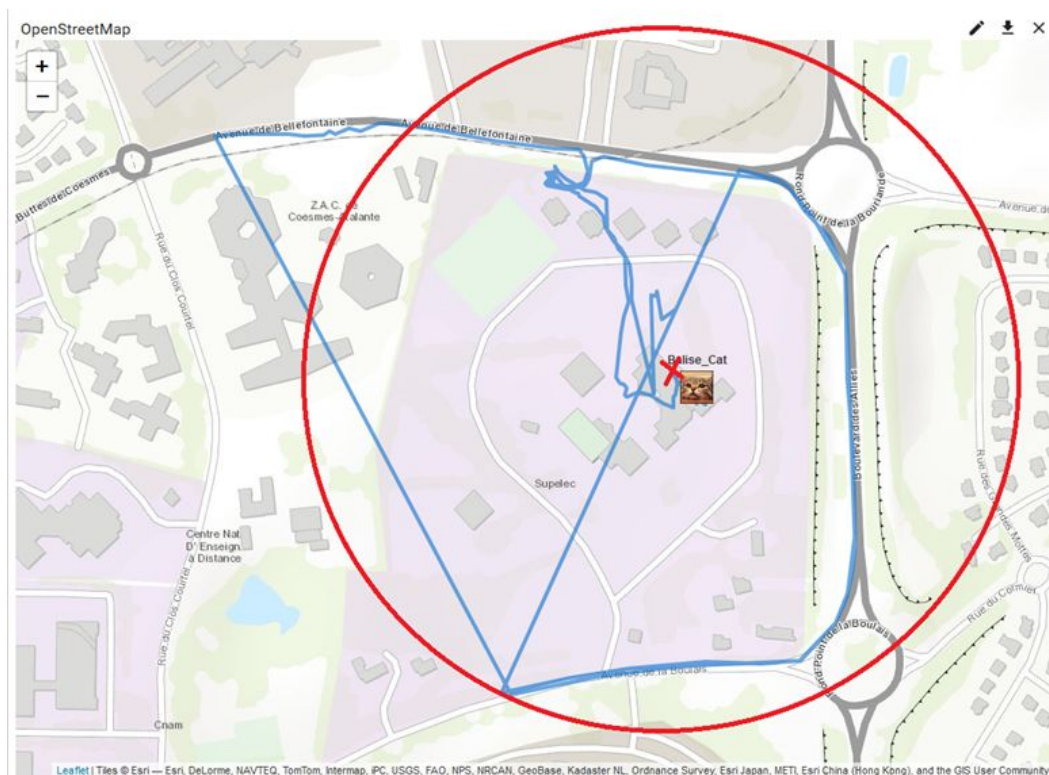
5) Test en intérieur

En intérieur, la localisation est moins précise : immobile, la précision est d'environ 10 mètres en général.

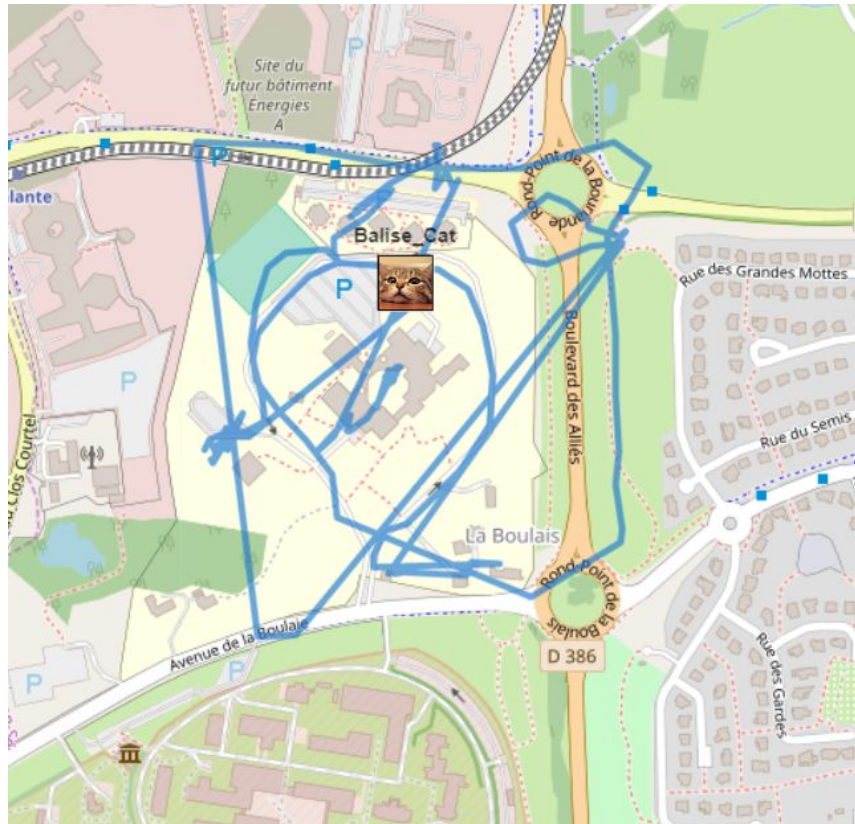


Évolution de la position immobile en intérieur sur une après-midi

Nous avons voulu faire un test dans les mêmes conditions que le test fait lors de la mineure IoT afin d'observer les progrès possiblement effectués. Nous avons observé les mêmes résultats que précédemment. Cependant, il pleuvait, ce qui a pu influencer les mesures.



Trajet effectué lors du test avec la station de base à l'intérieur de l'école et la balise dans une voiture (mineure IoT)



Trajet de la balise lors du test en TL système

IV - Réduction de la consommation

Notre système doit être autonome en énergie afin de ne pas gêner l'animal. La balise fonctionnera donc sur batterie et son autonomie doit être suffisamment longue pour que sa recharge ne soit pas contraignante pour l'utilisateur. Une autonomie d'au minimum une semaine serait idéale. La consommation de la balise doit donc être la plus faible possible afin de maximiser l'autonomie du système tout en limitant l'encombrement causé par la taille de la batterie.

1) Améliorations mises en place

Nous avons pu mettre en place différentes améliorations visant à réduire la consommation d'énergie du dispositif durant les différentes phases de son fonctionnement.

Les principaux composants consommateurs optimisés sont :

- ◆ Le microcontrôleur, ESP32
- ◆ Le module LoRa, RFM96W
- ◆ Le module GPS, UBX-M8030-KT

Ces différents composants peuvent être mis en veille afin de minimiser leur consommation lors de leurs phases d'inactivité. Cette stratégie est la principale source d'économie d'énergie.

La station de base calcule la durée de mise en veille de la balise à partir de la position de l'animal, de sa vitesse, du niveau de batterie et de l'état de la fonction de la fonction recherche. Cette valeur varie entre 5s et 18min, elle est codée linéairement sur 1 octet et transmise de la station de base à la balise après chaque transmission de la position de l'animal afin de profiter du temps d'éveil de la balise.

Le module LoRa est l'élément le plus consommateur lors de l'émission d'un message. A titre indicatif, la datasheet et les mesures s'accordent sur une consommation d'environ 120mA en émission. Le message à émettre a donc été condensé afin de limiter la durée de cette consommation. La latitude et la longitude sont codées en virgule fixe sur 4 octets chacun. Le niveau de batterie est codé sur 1 octet. Afin de vérifier l'intégrité du message reçu, nous avons ajouté un checksum sur un octet en fin de message.

Lat_0	Lat_1	Lat_2	Lat_3	Lng_0	Lng_1	Lng_2	Lng_3	Bat	Check
-------	-------	-------	-------	-------	-------	-------	-------	-----	-------

Message émis par la balise

Après sa phase d'émission, le module est passé en mode RX_SINGLE afin de potentiellement recevoir le message émis par la station de base durant une durée maximale de 1,8s. Cette durée permet de recevoir le message indiquant le temps à passer en veille dans la majorité du temps en limitant le temps d'activité. Après réception ou à la fin du timeout, le module LoRa bascule en mode SLEEP jusqu'au réveil du microcontrôleur.

Symbol	Description	Conditions	Min	Typ	Max	Unit
IDDSL	Supply current in Sleep mode		-	0.2	1	uA
IDDIDLE	Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
IDDST	Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
IDDFS	Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
IDDR	Supply current in Receive mode	<i>LnaBoost</i> Off, higher bands <i>LnaBoost</i> On, higher bands Lower bands	- - -	10.8 11.5 12.1	- - -	mA
IDDT	Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = +7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	- - - -	mA mA mA mA

Consommation du module LoRa dans ses différents modes de fonctionnement

Le mode d'inactivité par défaut est le mode STAND_BY, qui consomme plus que le mode SLEEP. Malheureusement, nous n'avons pas pu vérifier le bon passage dans ce mode étant donné le matériel utilisé pour mesurer la puissance consommée. En effet, nous n'avons qu'une mesure de la consommation globale du système avec une précision affichée à 100µA près probablement surestimée.

Nous avons également réalisé des essais avec le module LoRa E32-TTL-100 qui permet un contrôle de ses modes de fonctionnement plus simple. Cependant, nous n'avons pas réussi à établir de communication avec la station de base constituée d'un LoPy4.

Le module GPS possède également deux modes de fonctionnement permettant d'optimiser sa consommation d'énergie.

Le mode ON/OFF permet de mettre le module en veille pour une durée supérieure à 10s. A son réveil, le GPS cherche à nouveau les satellites et détermine sa position durant une période déterminée. Puis, il se rendort pour la période d'inactivité indiquée.

Le second mode, dit cyclic tracking, permet de mettre le module en repos durant moins de 10s. Après avoir déterminé sa position, le GPS se place dans un état où il continue de suivre les satellites trouvés afin de se fixer plus rapidement à son réveil. Sa consommation en veille est plus élevée que le mode précédent, mais le temps d'initialisation est grandement réduit, ce qui permet de rester actif moins longtemps.

Diagram of ON/OFF operation

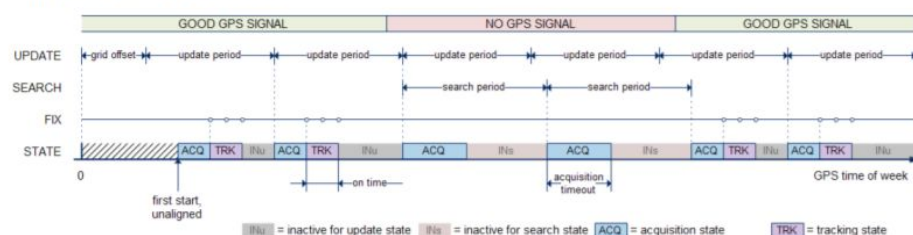


Diagram of cyclic tracking operation

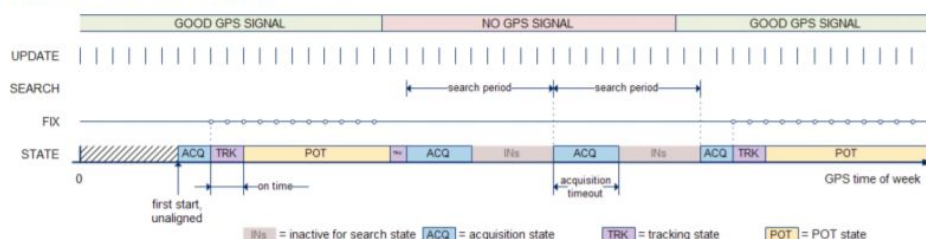
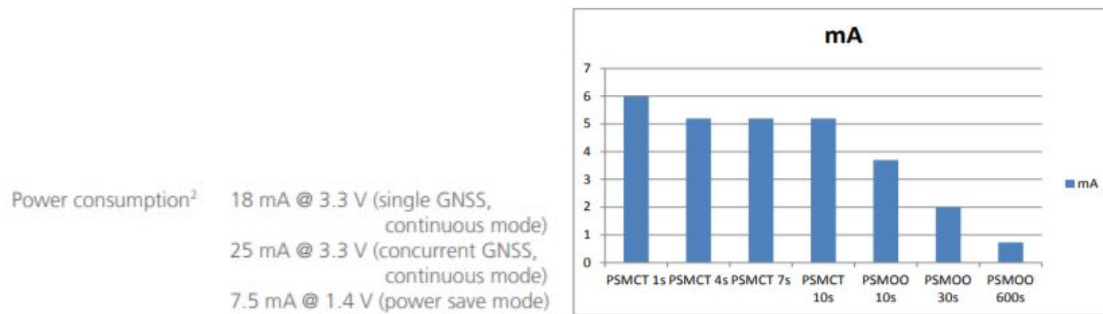


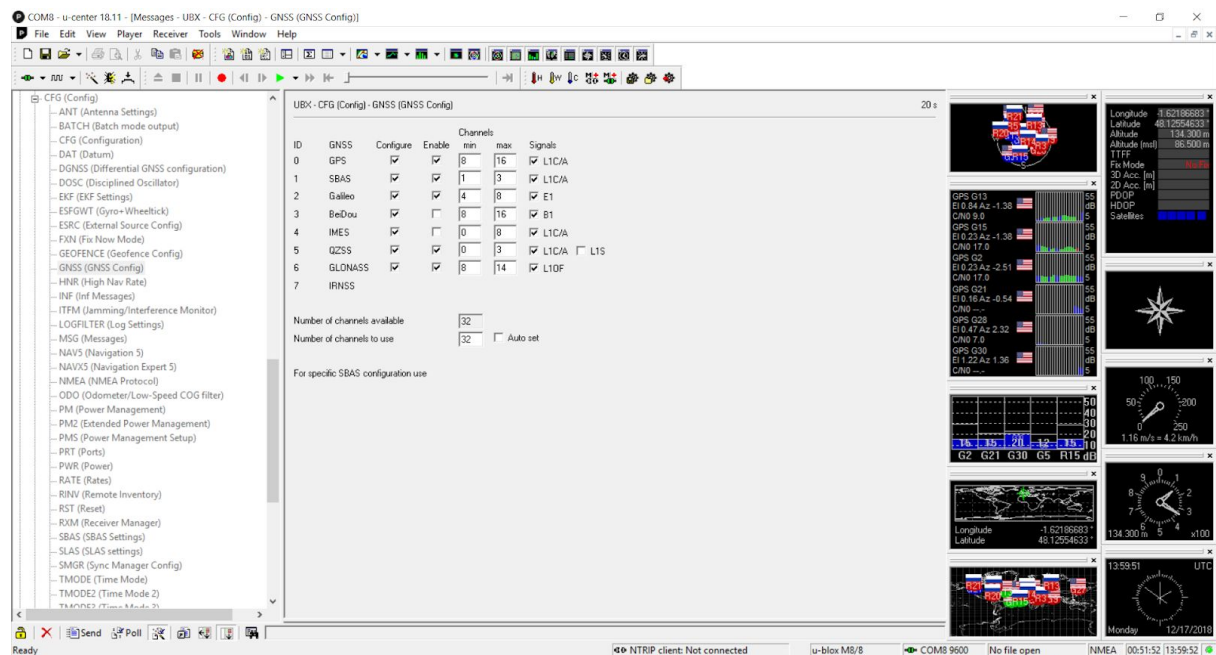
Diagramme d'état des différents modes de fonctionnement du module GPS

Nous utilisons donc ces deux modes selon la durée de veille demandée par la station de base. Mais la vérification du bon fonctionnement de ces modes n'a pas pu être vérifié pour les mêmes raisons que le module LoRa. On note cependant une diminution de la consommation globale.



*Informations sur la consommation du module GPS issues des datasheet
(PSMCT = Power Save Mode Cyclic Tracking, PSMOO = Power Save Mode ON/OFF)*

La structure des trames à envoyer au GPS via la liaison série a été déterminée à partir de la documentation sur le protocole UBX et du logiciel u-center qui permet de communiquer et de paramétrer le module GPS directement connecté en UART à un PC via un adaptateur USB. Ce logiciel est d'ailleurs très complet et permet une configuration avancée.



Logiciel u-center permettant de configurer les paramètres du module GPS

D'autres améliorations minimes ont été apportées sur le fonctionnement de l'ESP32 telles que la possibilité d'enlever tous les messages de debug via la liaison série ou de minimiser le temps d'initialisation.

Cependant, des optimisations plus efficaces ne peuvent pas être réalisées sous l'environnement Arduino qui fait fonctionner un seul coeur de l'ESP32 à 240MHz. Par exemple, la modification de la fréquence de fonctionnement du CPU ou l'utilisation du coprocesseur ULP n'est possible qu'en recompilant les bibliothèques fournies ou en utilisant le SDK d'Espressif. Cette dernière solution semble la meilleure en termes d'optimisations.

Power mode	Description			Power consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 14 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered on.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP co-processor is powered on.			150 μ A
	ULP sensor-monitored pattern			100 μ A @1% duty
	RTC timer + RTC memory			10 μ A
Hibernation	RTC timer only			5 μ A
Power off	CHIP_PU is set to low level, the chip is powered off.			0.1 μ A

Consommation de l'ESP32 dans ses différents modes de fonctionnement

2) Autonomie estimée

L'estimation de l'autonomie s'est faite via la mesure du courant du port USB alimentant la balise grâce au dispositif prêté par notre professeur, M. Weiss. Nous avons effectué des mesures sur 5 minutes afin d'avoir une idée représentative de l'énergie consommée durant cette période. Cependant, cette mesure ne prend pas en compte le système d'alimentation à partir de batteries et est biaisé par la consommation de l'adaptateur USB/UART servant à la programmation et à l'alimentation.

Configuration testée	Énergie consommée en 5 minutes (mWh)	Autonomie sur une batterie de 2000 mAh
Sans optimisation	92	6h
Veille minimale (5s)	25	22h
Veille moyenne (1min)	17	33h (1j + 9h)
Veille maximale (18min)	10	55h (2j + 7h)

Récapitulatif de la consommation de la balise mesurée

A titre indicatif, le courant en veille est stable et vaut 18mA. Le maximum de consommation est enregistré lors de l'émission et atteint 230mA. En dehors de ces deux périodes, l'activité du système est fluctuante et la consommation est de l'ordre de 100 mA. L'ensemble des composants fonctionne sous 3,3V.

Ainsi, la stratégie de veille mise en place a permis de décupler (dans le meilleur des cas) l'autonomie du dispositif. Cependant, il semble nécessaire de recharger quotidiennement la balise.

L'optimisation de la consommation est encore possible et peut permettre d'atteindre l'objectif d'une semaine fixé en début de partie. Pour cela, il sera important de changer d'environnement de développement, de confirmer l'état de veille des différents composants et de mesurer précisément la consommation de la balise dans sa configuration autonome finale afin de prendre en compte le rendement du circuit d'alimentation.

Il peut être également intéressant d'utiliser un système d'Energy Harvesting afin de prolonger l'autonomie de la balise si des dispositifs sont adaptés à un animal de compagnie.

Conclusion

Au cours de ce TL, nous avons pu améliorer notre dispositif de géolocalisation. La consommation a en effet été réduite et de nouvelles fonctionnalités sont implémentées.

Cependant, la portée n'a pas été significativement améliorée ce qui limite l'intérêt du dispositif ou nécessité de placer stratégiquement la station de base, sur un toit par exemple. Il pourrait être pertinent d'opter pour une autre technologie de communication afin d'obtenir une portée plus importante. Le module RF utilisé est capable de d'émettre et de recevoir via des bandes passantes beaucoup plus étroites ou d'utiliser d'autres codages sources, ce qui pourrait suffir pour obtenir une portée intéressante.

La consommation est réduite, mais encore trop élevée pour une utilisation sans rechargement quotidien. L'energy harvesting et l'optimisation du code via un changement d'environnement de développement pourraient être la solution.

Le système d'alimentation, le boîtier et le circuit imprimé n'ayant pas pu être conçus, ni réalisés dans le délai imparti, il est difficile d'estimer l'encombrement final du produit. Le choix des composants semble néanmoins pertinent.

