INSA

SRC 09

BUREAU D'ÉTUDE
RAPPORT

2020 • 2021

AIR'O X AIR BREIZH

SYSTÈME DE SURVEILLANCE DE LA QUALITÉ DE L'AIR

LE BOUHART Glenn, GUELLAEN Quentin, AGARINI Romain,
BUNOUF Thibaut, MEILHAT Marie

REMERCIEMENTS

Nous tenons à remercier l'ensemble des enseignants de SRC, plus particulièrement Mr MERIC et Mme UZEL pour leurs précieux conseils, et leur soutien technique. Merci également à Gilles Picoult, qui est toujours d'une grande aide dans nos différents projets depuis notre arrivée en SRC.

Un grand merci également à Air Breizh pour leur confiance, leurs conseils, ainsi que leur soutien matériel. Cela aura été un réel plaisir d'avancer ensemble durant ce semestre.

Aussi, nous remercions le pôle numérique de Rennes Métropole et sa DSI pour l'accès à leur réseau, plus particulièrement Mr Hervé. Un merci également à Marion Floury pour son aide avec la plateforme Wiotys de Wi6labs.

Nous tenions aussi à remercier Etienne Le Corre pour son aide lors de la modélisation de notre boitier, et le prêt de son imprimante 3D.

TABLE DES MATIERES

GLOSSAIRE

Microcontrôleur : circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

LoRa : technologie de modulation par étalement de spectre qui permet aux objets connectés d’échanger des données de faible taille en bas débit.

LoRaWAN : protocole de télécommunication permettant la communication à bas débit, par radio, d'objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à l'Internet via des passerelles, participant ainsi à l'Internet des objets.

Framework : En programmation informatique, un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie d'un logiciel.

Back-end : En informatique, back-end est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au **front-end** qui lui est la partie visible de l'iceberg

INTRODUCTION

"Il est reconnu à chacun le droit de respirer un air qui ne nuise pas à sa santé et d'être informé de la qualité de l'air qu'il respire."

Loi sur l'air et l'utilisation rationnelle de l'énergie du 30 décembre 1996

Mots-clés : objets connectés, interface, cartographie, internet des objets, qualité de l'air, réseau basse-consommation

Ce rapport présente le travail fourni lors du premier semestre de notre 5ème année en SRC. Il présente le fruit du partenariat avec l'association Air Breizh dans le cadre du module "Projet industriel" pour la réalisation d'un prototype de dispositif de surveillance de la qualité de l'air

Ce projet, démarré l'an dernier et axé sur la thématique de la pollution de l'air, nous tient énormément à cœur, et nous sommes ravis d'avoir pu l'emmener un peu plus loin cette année.

Après avoir fait une présentation globale du projet et de ses acteurs, nous nous pencherons sur le côté technique du projet. Ensuite, certaines phases de tests et leur résultats seront présentés, avant de faire un point sur la gestion de projet, sa suite, et de conclure.

PARTIE I
PRÉSENTATION DU PROJET

Nous allons ici faire une présentation globale de l'historique et des différents acteurs de ce projet industriel. Nous présenterons d'abord le projet Air'o, qui date de 2018, puis l'association Air Breizh qui nous a suivi durant ce semestre. Enfin, nous préciserons les objectifs du projet, définis ensemble.

PARTIE I
PRÉSENTATION DU PROJET

I | PRÉSENTATION DE AIR'O

Air'o est un projet initié lors de notre 4e année dans le cadre du module "projet électronique" du département Systèmes et Réseaux de Communication de l'INSA Rennes. En septembre 2019, nous devions choisir le sujet sur lequel nous voulions travailler. Nous sommes partis du constat que nous ne pouvions plus nier l'urgence climatique mondiale et l'augmentation de la pollution, et nous voulions intégrer cette notion au module. Il a alors été assez évident pour notre groupe de porter un projet à dimension environnementale. La pollution étant la troisième cause de mortalité en France, nous avons décidé de créer un dispositif de surveillance de la qualité de l'air.

Son fonctionnement est le suivant :
Un utilisateur possède un capteur de pollution fixé à son vélo. Ce capteur communique des données (1) à un serveur (3) régulièrement via le réseau LoRaWAN* (2). Pendant la réalisation d'un trajet, le vélo est en mouvement. Les données de qualité de l'air sont associées à des données de localisation, pour finalement les traiter et les afficher sur une carte via une interface (4). Le schéma suivant représente le fonctionnement global du projet :

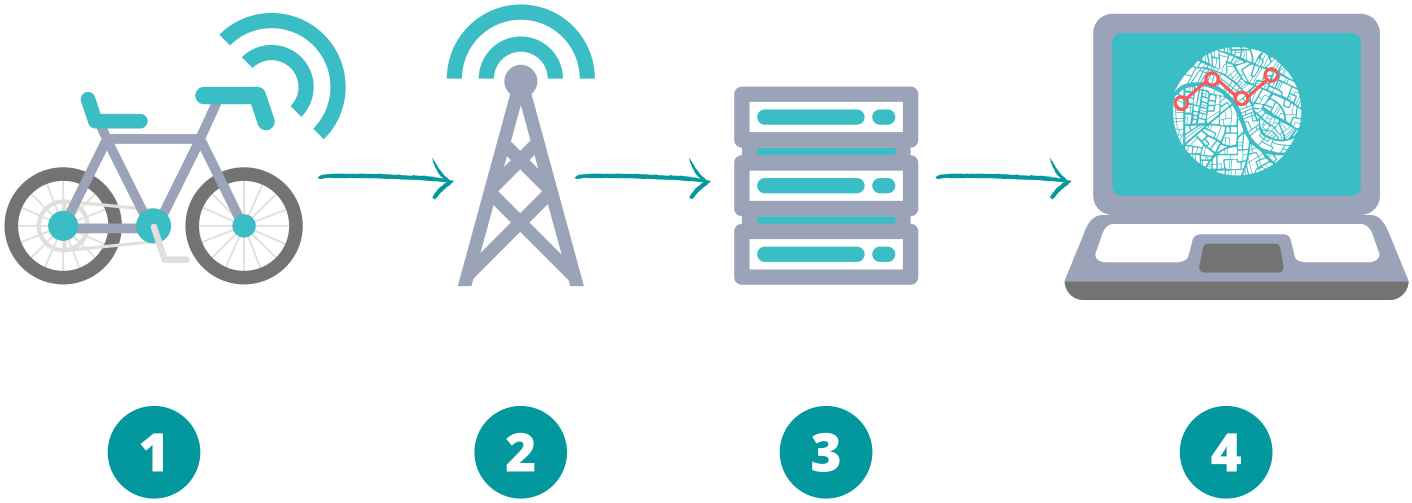


Figure 1 - Schéma global de fonctionnement du dispositif

*voir partie II.2.1

PARTIE I
PRÉSENTATION DU PROJET

II | PRÉSENTATION DE L'ASSOCIATION AIR BREIZH

Air Breizh est une des 19 associations de surveillance de la qualité de l'air en France, agréées par le Ministère de l'Environnement. Ensemble, elles constituent le dispositif national appelé ATMO, qui est la fédération des associations agréées de surveillance de la qualité de l'air.

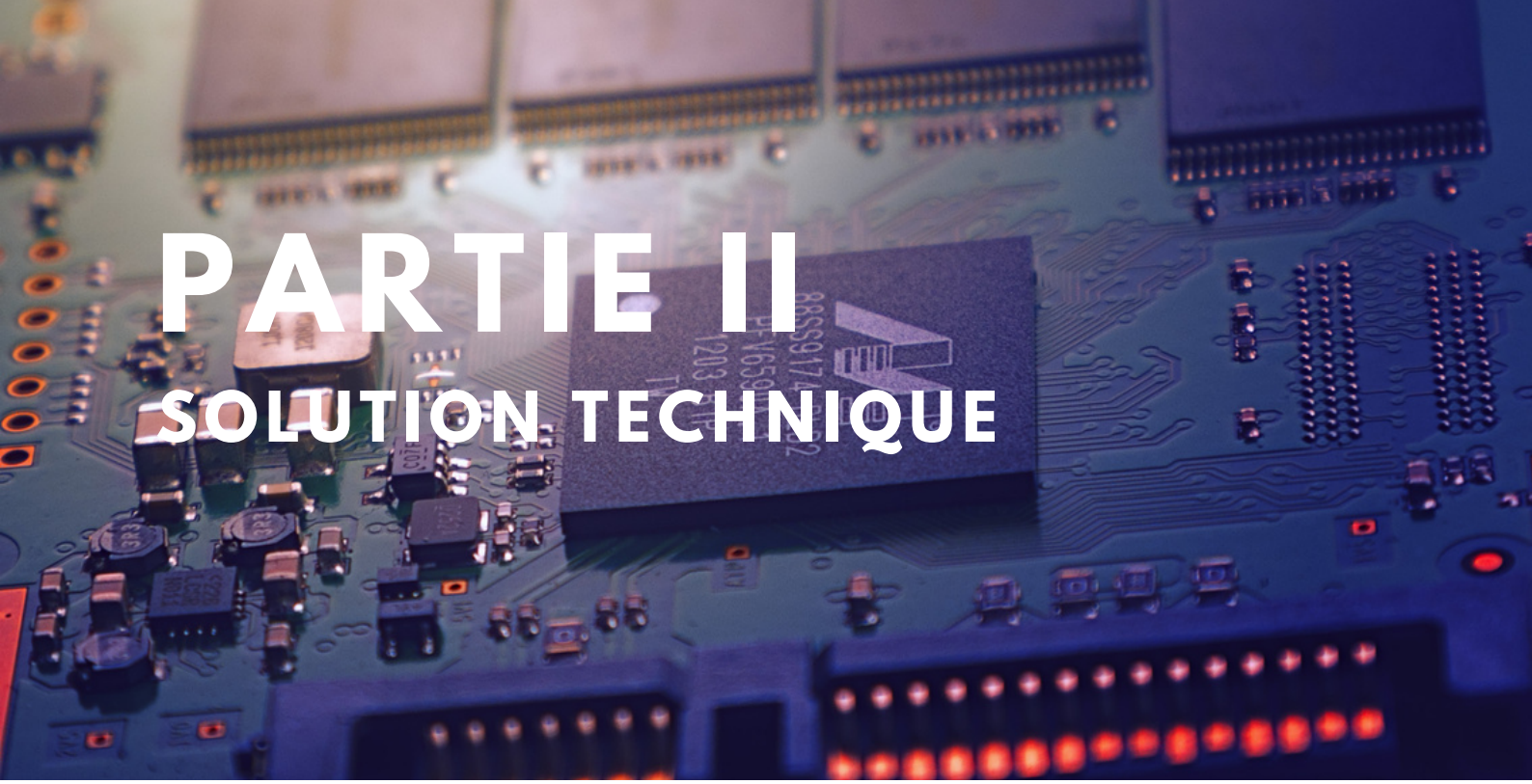
Les quatre missions de Air Breizh sont les suivantes :
- Mesurer une dizaine de polluants différents
- Informer, grâce aux indices de la qualité de l'air
- Etudier, grâce à différents équipements et méthodes d'analyse
- Sensibiliser via leur site web, des formations ou encore des publications

Sur leur site (Airbreizh, qualité de l'air en Bretagne), il est possible de visualiser la qualité de l'air sur différentes locations en Bretagne, grâce à un indice de pollution, l'indice ATMO.

III | OBJECTIFS DU PROJET

Nous sommes entrés en contact avec AirBreizh en octobre 2020, afin qu'ils nous accompagnent pour ce nouveau module "Bureau d'étude". Cette prise de contact a été l'occasion de redéfinir le projet pour ce nouveau semestre. Le fonctionnement global reste le même, mais les outils utilisés sont différents : nouveaux microcontrôleurs, nouveau système de base de donnée...

L'objectif pour la fin du semestre était donc de réaliser un prototype de dispositif de surveillance de la qualité de l'air fonctionnel, sur le même principe que l'an dernier. Ce prototype doit comprendre un device (Objet connecté mobile) qui prend les mesures, une transmission de données vers un serveur dédié avec une chaîne de communication complète, et l'affichage des données recueillies sur une interface.
Un point intéressant pour AirBreizh était également que l'on teste différents microcontrôleurs, et que notre rendu leur permette de reproduire/faire évoluer le projet de leur côté facilement.



PARTIE II
SOLUTION TECHNIQUE

Dans cette partie, nous aborderons la face technique du projet. Dans un premier temps, nous nous concentrerons sur les devices et leurs composants : micro-contrôleurs, capteurs, alimentation... Ensuite, nous présenterons le réseau LoRaWAN, son fonctionnement et notre utilisation. Enfin, nous nous concentrerons sur l'interface utilisateur.

PARTIE II
SOLUTION TECHNIQUE

I | DEVICES

Nos devices sont chacun composés des éléments suivants : Des capteurs (pollution et GPS), un module de communication LoRa, un micro-contrôleur, une alimentation, le tout inséré dans un boîtier. Chaque composant est détaillé ci-dessous.

1.1 | Microcontrôleurs

Pour ce semestre, nous avons pu avoir accès à différentes cartes de différents fournisseurs : Pycom, STMicroElectronics, Espressif Systems...

- PYCOM

L'INSA nous a mis à disposition 2 "cartes pycom", comme nous allons les appeler dans ce rapport. Il s'agit en fait d'une carte LoPy 4 avec l'expansion board 2.0 de chez Pycom :

Le Lopy 4 est une carte de développement miniature dédiée aux objets connectés, basée sur le langage Python 3. Ses caractéristiques sont spécifiées en Annexe B.

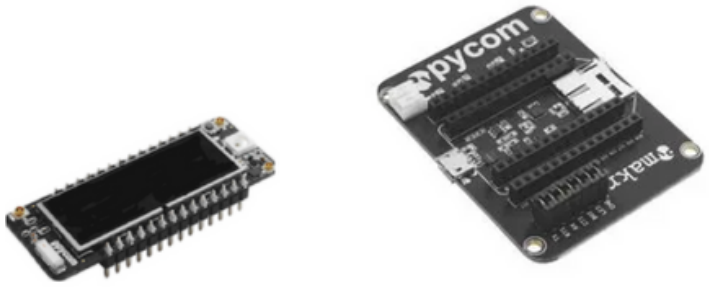


Figure 2 - Carte LoPy 4 et Expansion board 2.0 de chez Pycom

Cette carte est celle que nous avons le plus utilisé, du fait notamment de sa facilité d'utilisation. La documentation est plutôt facile d'accès, et la prise en main du langage python est assez instinctive et aisée. En cas de souci, les forums sont complets et actifs. Il y a quelques détails et particularités lors de l'installation des extensions/logiciels qu'il faut appréhender et des méthodes de résolutions d'erreurs à connaitre. Nous avons précisé ces points dans un document qui a déjà été remis à l'association Air Breizh.

Pour cette carte, le code d'émissions de données en LoRa est disponible dans l'archive numérique du projet.

- ESP32

ESP32 est une série de microcontrôleurs d'Espressif Systems. Il embarque un module wifi, un module bluetooth ainsi qu'un module LoRa. Le modèle mis à notre disposition est le TTGO LORA32. Il embarque aussi un petit écran Oled qui permet d'afficher quelques informations.

Figure 3 - ESP32

Les caractéristiques de cette carte sont disponibles en Annexe D.

ESP32 semblait être une bonne solution pour notre device. L'environnement de travail sous Arduino est agréable et nous permet d'utiliser des bibliothèques existantes simplement. La première connexion à la carte était rapide et facile à mettre en place. Cependant, nous avons rencontré plusieurs problèmes : La documentation de ce microcontrôleur est introuvable et peu d'informations sont disponibles. L'environnement de travail sous Arduino est agréable et nous permet d'utiliser des bibliothèques existantes simplement. l'ESP32 dispose d'un module LoRa 433MHz et n'est donc pas adapté aux bibliothèques que nous utilisons. Pour émettre à cette fréquence, nous aurions dû reprendre intégralement une bibliothèque en langage C. Nous avons donc mis de côté cette solution.

- STM32

Nous avions une carte Nucleo L073RZ avec le module LoRa Mbed SX1272.
La carte STM32 nous permet de rentrer très en profondeur du point de vue hardware mais cela nous complique la tâche. Nous avons réussi à faire fonctionner un test d'envoi de trames LoRa à partir d'un projet existant.

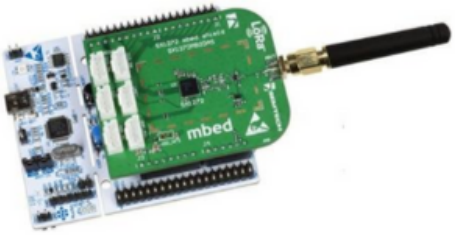


Figure 4 - STM32

Cependant, la complexité de l'architecture du programme et le logiciel utilisé pour le développement étant sous licence (disponible uniquement sur les pc de l'INSA) nous ont rapidement découragés. Néanmoins des solutions libres de droit restent disponibles pour utiliser cette carte, notamment ArduinoIDE ou CubeIDE. STM32 reste une piste à explorer.

1.2 | Capteurs

- Pollution : SDS011

Le capteur SDS011 mesure le taux de particules fines dans l'air. Ces particules proviennent de plusieurs sources (fumées, pollen, poussières, ...) et leur concentration est calculée selon deux tailles distinctes : 10um et 2.5um. Un ventilateur permet de faire circuler l'air dans un système qui compte et mesure les particules grâce à un laser.
La datasheet de ce composant est disponible dans la sitographie de ce rapport.

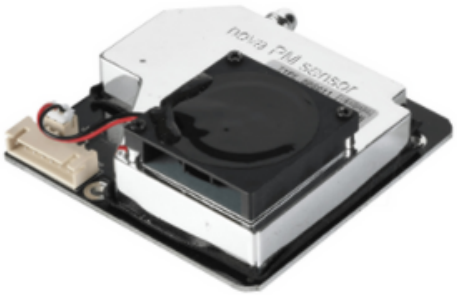


Figure 5 - Capteur SDS011

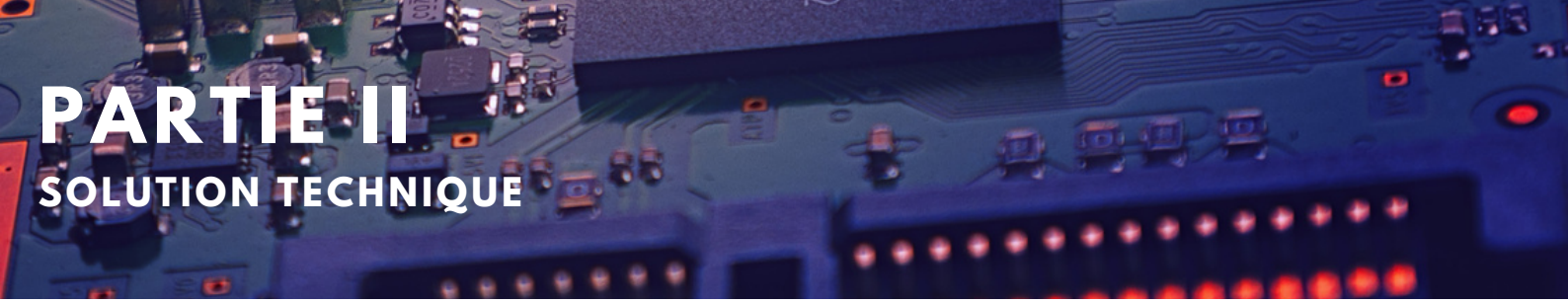
Interface de communication	Tension d'alimentation
UART	5V
Pins à connecter (SDS011)	**Pins à connecter (Pycom)**
5V	Vin
GND	GND
TXD	P21
RXD	P22

Au niveau software, nous utilisons la bibliothèque "sds011.py", un code d'exemple d'utilisation est disponible dans l'archive numérique du projet.

- Pollution : SENSIRION SPS30

Le capteur Sensirion fonctionne sur le même principe. En théorie il atteint une meilleure précision et permet de mieux identifier différentes tailles de particules (entre 1um et 10um). Il est également plus compact.Cependant les connecteurs que nous avions à notre disposition n'étaient pas adaptés pour expérimenter son fonctionnement.

Figure 6 - Capteur SPS30



PARTIE II
SOLUTION TECHNIQUE

Interface de communication	Tension d'alimentation
UART ou I2C (au choix)	5V

La datasheet du SENSIRON SPS30 est disponible en lien dans la sitographie.

- Module GPS

Le module GPS Grove permet la récupération d'informations sur la position géographique, l'altitude ou encore la vitesse de notre objet connecté. Il communique via une liaison série des trames qui peuvent être décodées pour en extraire les données souhaitées. Sa datasheet est disponible dans la sitographie.

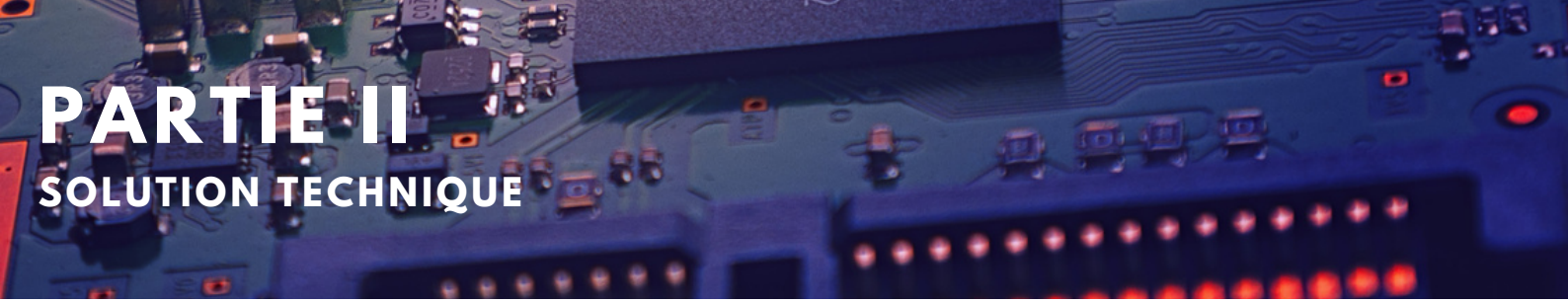
Figure 7- Module GPS

Interface de communication	Tension d'alimentation
UART	3.3V

Son utilisation est simplifiée par la bibliothèque MicropyGPS. Un exemple de code minimal est également disponible dans l'archive numérique du projet.

1.3 | Alimentation

L'alimentation de nos devices est un point essentiel au vu de nos objectifs en termes de consommation énergétique et de mobilité. Nous avons dans un premier temps réalisé une étude sur la consommation de nos différents composants :



PARTIE II
SOLUTION TECHNIQUE

Composants	Tension d'alimentation	Consommation
STM32 L073RZ	3,3V ou 5V ou 7-12 V	Mode dynamique :200 mA Mode Low Power : 100 mA
PYCOM LOPY 4	3,3V ou 3,5-5,5V	En transmission LoRa : 28 mA
ESP 32	2,3-3,6V	Environ 200 mA
Shield LoRa Sx2172 (couplé au STM32)	3,3V	En émission : 10 mA Au repos : 200 nA
Capteur SPS 30	4,5V - 5,5V	En mesure : 55±10 mA Au repos : 0,3 mA
Capteur SDS011	5V	En mesure : 70 mA ±10mA Au repos : < 4 mA
GPS neo 6M	3.3V	En mesure : 35 mA

La carte PYCOM Lopy 4 s'étant rapidement démarquée comme étant la solution la plus pertinente, nous avons décidé de chercher une alimentation délivrant 5V et avec une capacité approchant les 10.000 mAh.

Dès lors plusieurs solutions sont possibles pour alimenter nos devices:

- **Piles classiques :** solution non optimale pour l'utilisateur et par rapport à nos engagements écologiques sur ce projet

- **Batterie rechargeable :** Solution idéale en apparence, car la grande majorité de ces batteries suspendent l'alimentation du device si la consommation énergétique est très faible. Cela n'est donc pas en phase avec l'émission en LoRa.

- **Système chargeur + piles rechargeables :**
  Le système d'alimentation retenu:
  - 4 accus R6 2600 mAh
  - sortie : 5 V
  - charge totale nécessaire : 6h
  - Dimensions: 81 x 64 x 25 mm

Figure 8- Piles rechargeables

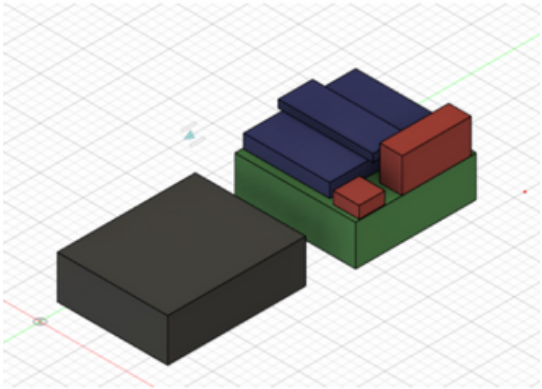
Autonomie mesurée : 9h (pour un débit important et pas de mise en veille)
On peut s'attendre à une amélioration significative de cette autonomie lorsque le code sera optimisé pour respecter le duty cycle (émission 1% du temps) en LoRa.

Initialement nous souhaitions rendre notre système autonome en énergie en utilisant une dynamo de moyeu et une batterie. Le système complet était trop coûteux par rapport à notre budget. La consommation énergétique de notre système étant très faible, on peut s'attendre à ce que l'utilisation de la dynamo de moyeu soit suffisante.

1.4 | Boitier et device final

Nous avons réalisé un boîtier en utilisant une impression 3D plastique. Ce boîtier a pour objectif d'être suffisamment compact pour améliorer la mobilité du système.

Dans un premier temps, nous nous sommes penchés sur l'optimisation de l'agencement des composants. C'est l'agencement suivant qui a été retenu:



Pycom Lopy 4 + Expansion Board
Capteur SDS011
Capteur GPS
Chargeur + piles

Figure 9- Modélisation 3D - Agencement des composants

Puis, nous avons modélisé le boîtier suivant sur le logiciel Fusion 360 en respectant les dimensions de nos composants, en intégrant les différents orifices nécessaires et une couche de mousse (1 cm) pour absorber les vibrations:

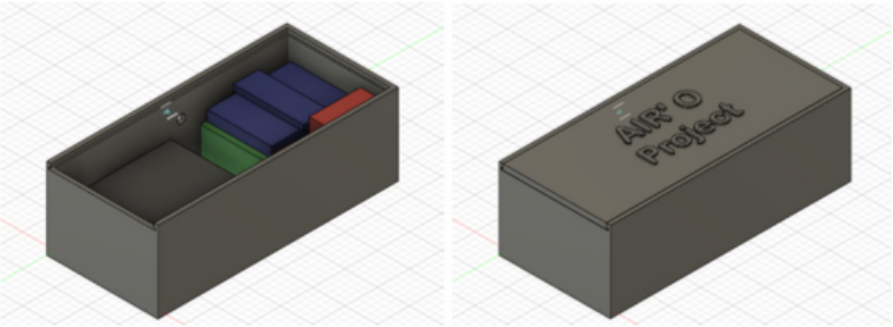


Figure 10- Modélisation 3D - Agencement des composants et boitier

Le rendu final de notre système est le suivant :

Figure 11- Boitier imprimé avec composants

II | RÉSEAU LORAWAN

2.1 | Fonctionnement

LoRaWAN (Long Range Wide Area Network) est un protocole de communication qui appartient à la catégorie des LPWAN (Low Power Wide-Area Network), réseaux basse consommation d'énergie et longue portée. Ces réseaux sont adaptés aux objets connectés dont l'application requiert une autonomie importante, ce qui est notre cas pour ce projet.

PARTIE II
SOLUTION TECHNIQUE

Sur le graphique suivant est représenté le positionnement des réseaux LPWAN par rapport aux autres réseaux sans fils.

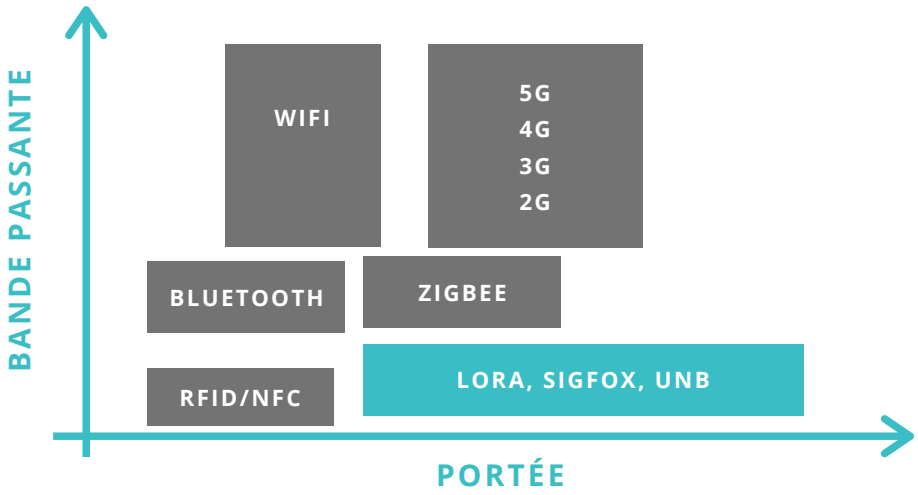


Figure 12 - Comparaison des réseaux sans fils

L'architecture d'un réseau LoRaWAN se décompose en 4 parties : End Devices, Gateway LoRaWAN (antenne), Serveurs, Application.
On les retrouve dans le schéma suivant :

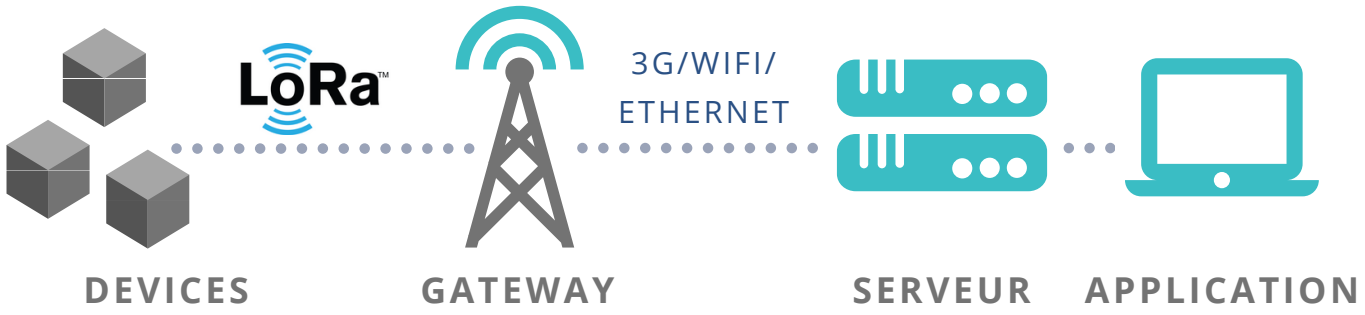
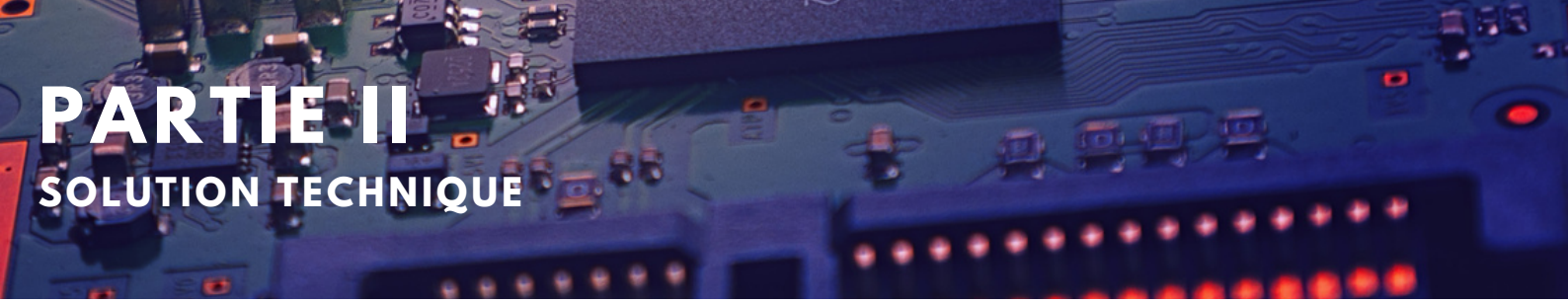


Figure 13 - Architecture d'un réseau LoRaWAN

Les devices envoient des données à l'antenne grâce à la modulation LoRa. L'antenne est connectée à internet (via une connexion wifi ou un câble ethernet par exemple) et envoie ces données sur les serveurs via internet. Elle sert de relais. Ces données peuvent ensuite être récupérées sur les serveurs, et utilisées pour des applications diverses et variées.



PARTIE II
SOLUTION TECHNIQUE

Nous n'allons pas nous étendre sur le fonctionnement de la modulation LoRa ici, mais donnerons tout de même quelques spécifications à propos de la modulation LoRa :

- **Bande de fréquence en Europe**
  L'interface radio LoRa utilise la bande de fréquence ISM (Industrielle, Scientifique et Médicale) dite Sub-GHz des 868 MHz (alors qu'aux Etats-Unis on utilise la bande des 915 MHz).
- **Largeur de bande BW ou encombrement spectral du signal modulé**
  La norme prévoit 3 valeurs de BW possibles : 125 kHz, 250 kHz et 500 kHz. En Europe, seules les largeurs de bande 125 kHz et 250 kHz sont utilisées.

Différence entre LoRa et LoRaWAN
La technologie LoRa désigne l'interface radio du réseau de communications sans fil (couche physique), alors que LoRaWAN désigne le protocole de communication.

Une transmission utilisant la modulation LoRa se fait suivant le schéma ci-dessous :

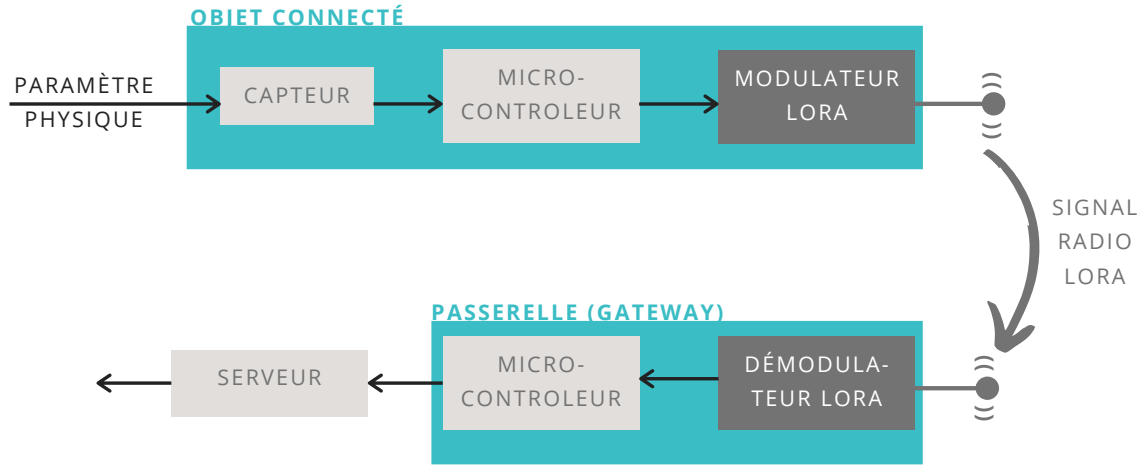
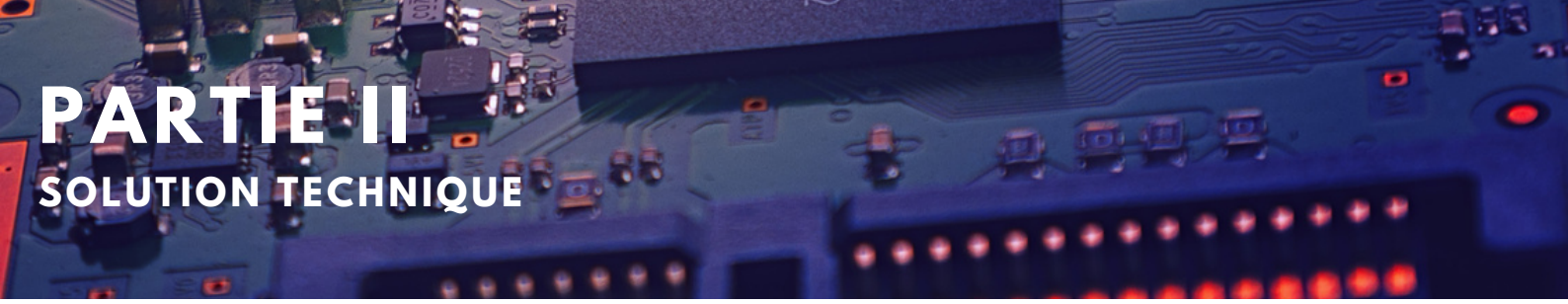


Figure 14 - Chaîne de communication LoRa

Le capteur récupère le signal d'un paramètre physique (Taux de particules fines et localisation dans notre cas). Ce signal est ensuite traité par un microcontrôleur, modulé, puis envoyé via un signal radio LoRa.



PARTIE II
SOLUTION TECHNIQUE

2.2 | Wi6labs

Wi6labs est une solution de réseau et d'infrastructure pour l'IoT. Elle opère l'infrastructure réseau du network serveur jusqu'au cloud sécurisé. Sur leur interface on peut notamment :

- enregistrer la/les gateway(s)
- enregistrer les capteurs
- récupérer les différentes clés nécessaires pour la chaîne de communication
- Superviser les flux de données

Après s'être identifiés, nous avons accès au trafic de notre/nos gateway(s). Nous avons également le détail des données échangées par chaque capteur enregistré.

Le serveur de Wi6labs est notre Network Server, nous pouvons alors transmettre les données via une requête http vers notre Data Server, traiter ces données, et les utiliser/afficher dans une interface.

2.3 | Système embarqué

A l'extrémité du système global se trouve l'objet connecté. Il constitue l'élément embarqué et nous allons ici décrire son fonctionnement.

Lorsqu'il est alimenté, il cherche à joindre le réseau LoRa. Une fois cette connexion établie, un cycle infini se lance. Ce cycle réalise la mesure des données GPS et de pollution de l'air à l'aide des instruments auxquels il est relié. Ensuite, vient L'émission de ces données vers la Gateway la plus proche, elle même connectée au Network Server.
A la fin de chaque cycle, l'objet passe en mode sommeil pendant une durée paramétrable.

La technologie LoRa est par définition bas débit. Ainsi pour limiter le temps d'émission et aussi pour ne pas dépasser la taille limite du payload (charge utile contenant le message à transmettre) fixée par le Network Server, nous avons compressé les données à transmettre dans une trame.

Elle se présente comme suit :

$AIRO,6.9,8.7,[49, 33.3739, 'N'],[1, 50.8075, 'W'],(10, 5, 29.0)

Eléments de la trame	Signification
$AIRO	Tag de début de trame
6.9,8.7	Taux de PM2.5, Taux de PM10
[49, 33.3739, 'N']	Latitude [degrés, minutes, orientation]
[1, 50.8075, 'W']	Longitude [degrés, minutes, orientation]
(10, 5, 29.0)	Horaire (heure, minute, secondes)

Ainsi, chaque trame émise permet de situer la mesure des taux de particules fines dans un contexte spatiotemporel.

Toute l'information est contenue dans une chaine de caractère. Cette chaîne est traduite en hexadécimal lors de la transmission et elle reste sous cette forme sur les serveurs Wi6labs et Airo (Network Server et Data Server) ainsi que sur la base de données utilisée par l'application. Elle est destinée à être séparée et analysée par l'application présentée dans la partie suivante.

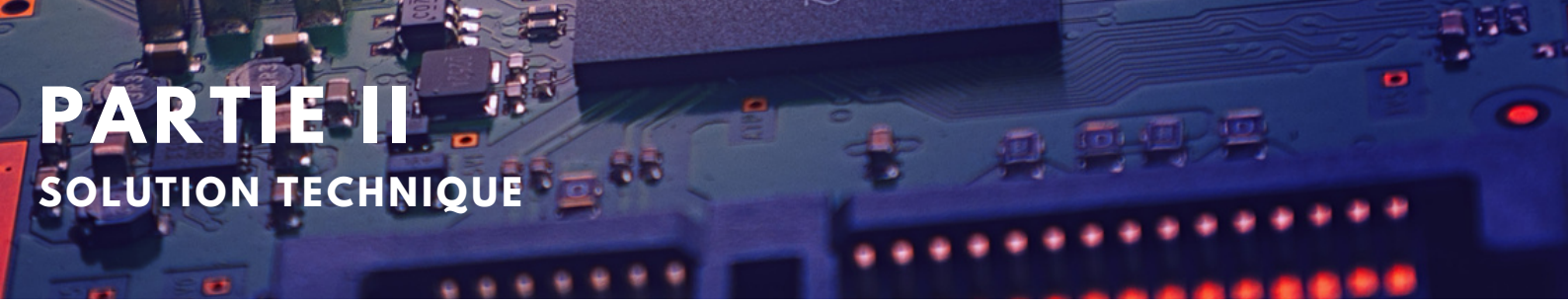
III | APPLICATION

3.1 | Outils

Le projet AirO nous a amené à développer une application web utilisant les technologies Django et VueJS d'après le cahier des charges donné par AirBreizh. Le site donne accès à différents onglets pour consulter les données mesurées par les devices, les trajets effectués, mais aussi une cartographie des données de pollution suivant une échelle de couleur. Pour réaliser cette interface, nous nous sommes formés sur les technologies suivantes :

- DJANGO

Django est un framework python haut niveau qui propose une solution de développement web à vocation simple et rapide.



PARTIE II
SOLUTION TECHNIQUE

Django met à la disposition du développeur :
- Une architecture MVC (Model Vue Contrôleur)
- Une API d'accès aux données (Pymongo)
- Un serveur web de développement pour faire les tests sans déploiement

Ses atouts sont sa rapidité de mise en œuvre, sa documentation fournie, son système d'authentification qui garantit une bonne sécurité et son adaptabilité. Il est largement utilisé par de nombreux géants du web comme Google, Youtube, Dropbox...

- VUEJS

Vue JS est un framework javascript utilisé pour construire des interfaces utilisateurs. La librairie Vue se consacre à la modélisation de la couche vue client exclusivement et est facilement intégrable dans différents projets et s'insert parfaitement dans notre interface Django. Ses atouts sont sa grande adaptabilité, sa simplicité de mise en place et ses performances aussi bien sur des applications simples que des sites web plus élaborés.

- GITLAB

GitLab est un logiciel libre de forge basé sur git, proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue. L'insa dispose d'un espace gitlab dédié aux élèves. Nous avons pu exploiter les fonctionnalités de cet outil afin de développer l'interface utilisateur. Chaque fonctionnalité a pu être ajoutée au fur et à mesure et le travail pouvait être réalisé en parallèle.

Le livrable de l'interface est composé d'un serveur django lancé depuis le fichier manage.py. Tous les fichiers de configuration du serveur sont regroupés dans le dossier Dash.

Deux applications principales sont lancés à l'instanciation du serveur :
- airoapp -> Application qui fait office de data server.
- apps/dashboard -> Application interface web.

Ces deux applications ont été développées de manière totalement indépendantes, puis nous les avons assemblées afin de pouvoir faire l'affichage des données en direct.

3.2 | Data server

La première application est celle qui fait office de Data Server dans notre architecture. Elle fait le lien entre le Network Server et la base de données.

On a configuré sur Wi6labs (le Network Server), l'envoi d'une requête http automatique à chaque réception d'un message provenant des objets connectés. La requête envoyée contient la charge utile avec les données de pollution, de localisation et le timestamp mais aussi des informations sur l'identité de l'objet et sur la qualité de la transmission LoRa.

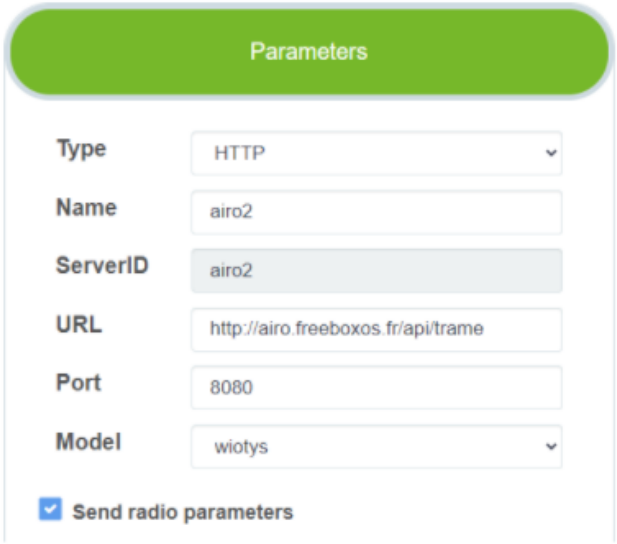
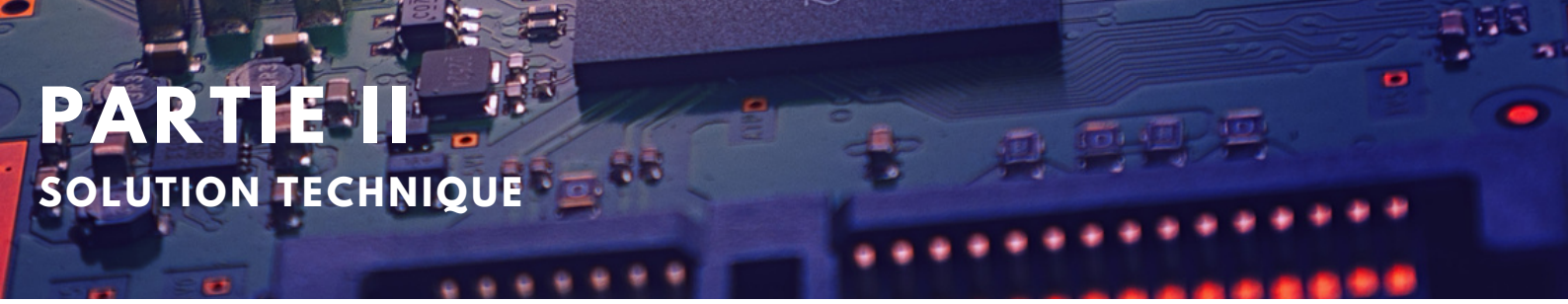


Figure 15 - Configuration de la direction des requêtes lors de la réception d'un message

Notre application intercepte ces requêtes et les analyse afin d'insérer un nouveau document dans la base de données.

Figure 16 -Fonctionnement du Data Server



PARTIE II
SOLUTION TECHNIQUE

Le contenu de la requête est divisé en champs qui viennent instancier un objet associé à un modèle. Pour ne pas surcharger notre base de données, nous ne gardons que les champs suivants : *deveui, sensor_id, timestamp* et *payload_deciphered* (charge utile non cryptée contenant la trame mise en forme lors de l'envoi du message).

La base de données est implémentée via MongoDB. Nous avons choisi ce système de gestion de base de données car il nous était demandé de travailler en NoSQL. MongoDB permet de créer des collections (équivalentes aux tableaux en SQL) de documents (équivalents aux lignes d'un tableau en SQL). Le principal avantage du NoSQL est la possibilité de faire évoluer les champs des documents de la collection.

Ainsi AirBreizh pourra adapter le modèle de création de document suivant les informations qui doivent intégrer la base de données.

Après émission et traitement de plusieurs trames, on les retrouve dans la base de données qui se présente comme suit :

airoapp_trame

	_id ObjectId	deveui String	sensor_id String	timestamp String	payload_deciphered Stri
41	600fef4c68a99b51fb55a0af	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:30:35+01:00"	"244149524f2c392e362c31
42	600fef6368a99b51fb55a0b0	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:30:57+01:00"	"244149524f2c392e362c31
43	600fef7768a99b51fb55a0b1	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:31:17+01:00"	"244149524f2c31302e312c
44	600fef8b68a99b51fb55a0b2	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:31:37+01:00"	"244149524f2c31302e332c
45	600fefa168a99b51fb55a0b3	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:31:59+01:00"	"244149524f2c31302e332c
46	600fefb568a99b51fb55a0b4	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:32:19+01:00"	"244149524f2c392e372c31
47	600fefc968a99b51fb55a0b5	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:32:39+01:00"	"244149524f2c31302e322c
48	600fefdd68a99b51fb55a0b6	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:32:59+01:00"	"244149524f2c31302e322c
49	600feff268a99b51fb55a0b7	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:33:20+01:00"	"244149524f2c392e382c31
50	600ff00668a99b51fb55a0b8	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:33:40+01:00"	"244149524f2c392e372c31
51	600ff01a68a99b51fb55a0b9	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:34:00+01:00"	"244149524f2c392e392c31
52	600ff03068a99b51fb55a0ba	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:34:22+01:00"	"244149524f2c392e392c31
53	600ff04668a99b51fb55a0bb	"70b3d5499055aed6"	"70b3d5499055aed6"	"2021-01-26T11:34:44+01:00"	"244149524f2c392e392c31

Figure 17 - Collection contenant les trames dans la base de donnée

La base de données sera hébergée dans le même environnement que le serveur et que l'interface utilisateur. Les données à traiter et afficher dans cette dernière seront prélevées via pymongo, une bibliothèque python permettant de réaliser des requêtes dans MongoDB.

Note : Le Data Server permet aussi d'obtenir et de modifier les données enregistrées dans la base de données mais ces fonctionnalités n'ont finalement pas été utilisées dans l'application livrée.

3.3 | Interface Utilisateur

L'architecture de l'interface se décompose en trois parties :

- Le back end (partie logique et traitement des données construit avec le framework Django qui communique avec la base de données MongoDB.)
- Le front end (affichage gestion des pages)
- La couche Vue.JS qui lève des exceptions et lance des composants (notifications, messages d'erreur)

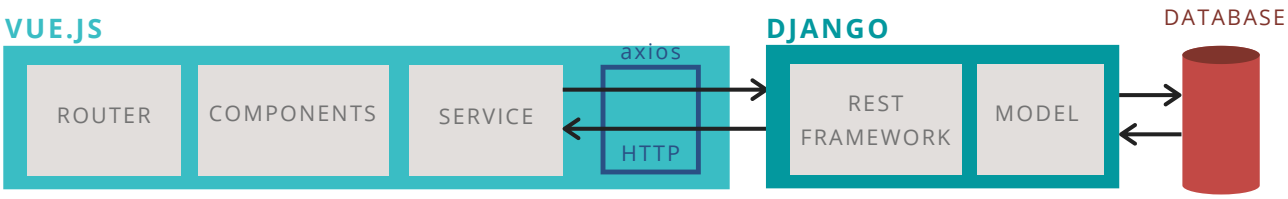


Figure 18 - Exemple d'architecture site Django/VueJS

L'application dashboard joue le rôle de routeur entre les différents composants du site. Les modèles de requêtes sont enregistrés dans le fichier urls.py et permettent d'exécuter les fonctions correspondantes et de retourner les pages à afficher en instanciant les composants.

Voici le principe de fonctionnement de l'interface lors de requêtes concernant l'application de gestion des relevés : (voir figure 19 ci-dessous)

- *Relevé* permet d'afficher un relevé en utilisant son identificateur (ID)
- Relevés permet d'afficher la liste complète des relevés créés par l'utilisateur
- Releve_Add permet d'ajouter un relevé
- Relevé_Edit permet de modifier les champs d'un relevé
- Api_Delete_Releve permet d'effacer un relevé, avec validation VueJS : affichage d'une boîte de dialogue pour le cas où l'utilisateur clique par mégarde sur supprimer le relevé

PARTIE II
SOLUTION TECHNIQUE

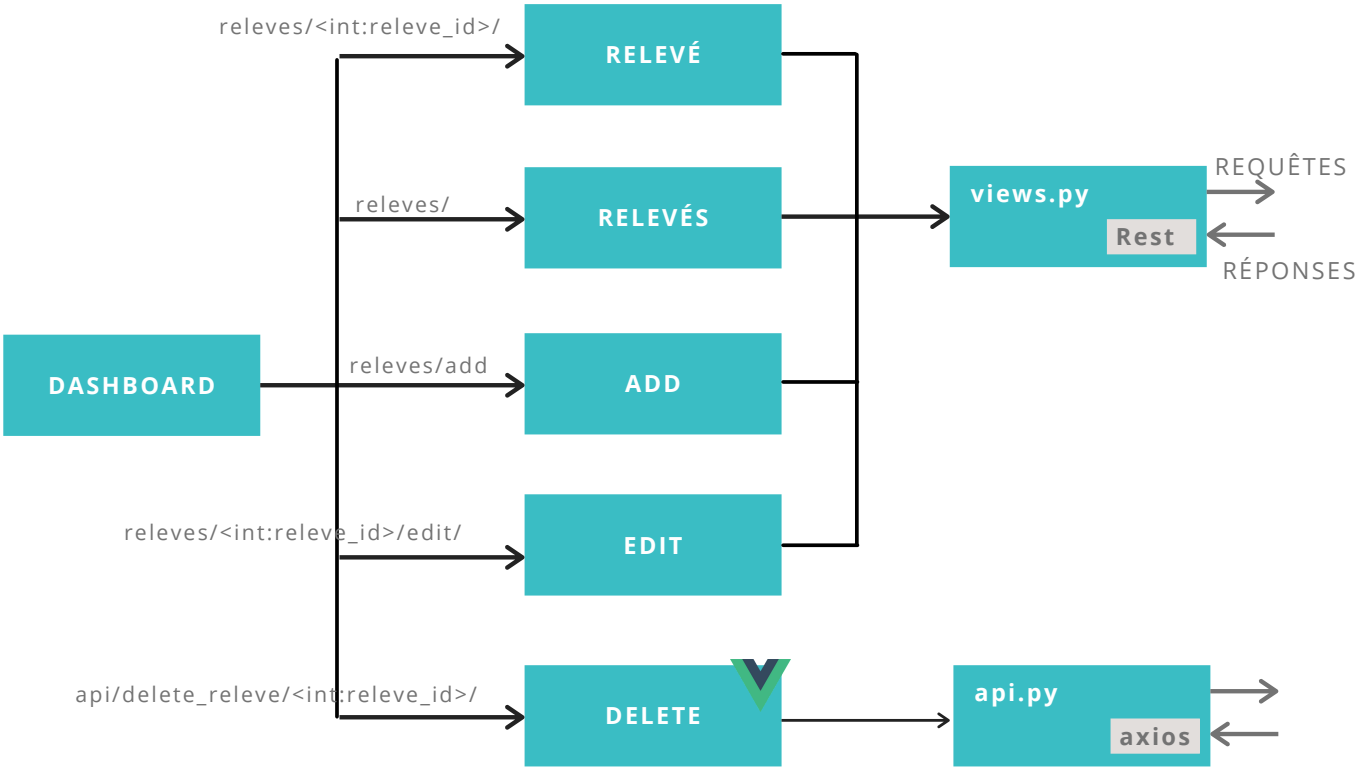


Figure 19 - Architecture du site Air'o pour la partie "relevés"

views.py et api.py contiennent la définition des méthodes qui effectuent les différentes actions définies précédemment (accès à la base de données locale, gestion des requêtes...), et utilisent respectivement les frameworks Django Rest et Axios pour envoyer les requêtes.

Les mêmes techniques sont utilisées pour coder les autres composants : dashboard, trajets et cartographie.

Au final, l'interface est constituée d'une page d'accueil, qui permet aux utilisateurs de se créer un compte où bien de s'y connecter. Une fois connecté, l'utilisateur dispose d'un dashboard sur lequel s'affiche le dernier relevé effectué avec les détails qui y sont associés : date, heure, position et valeurs relevées par les capteurs de particules fines.

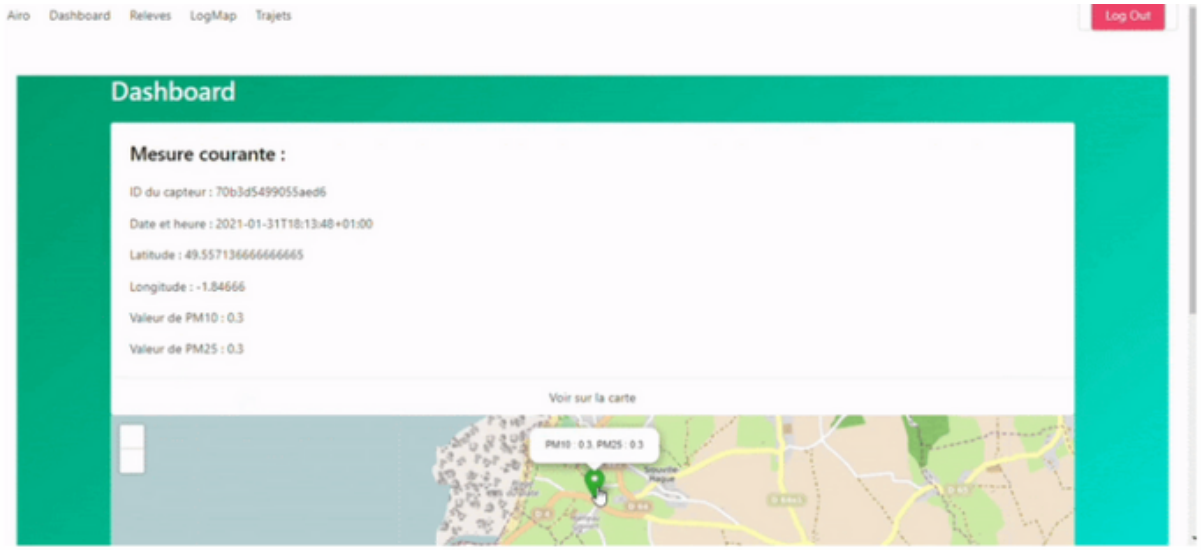
Ensuite, l'onglet relevé permet d'afficher toutes les données disponibles sur la BDD. et d'enregistrer une donnée dans un relevé.

PARTIE II
SOLUTION TECHNIQUE

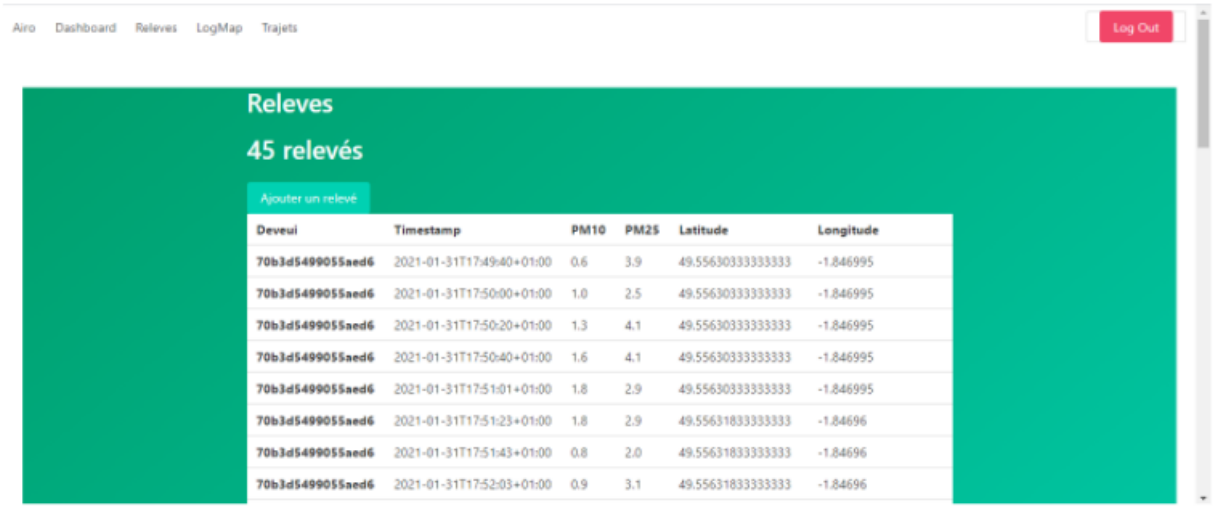
L'onglet LogMap propose une cartographie des données disponibles utilisant l'API open map. L'objectif était de créer un filtre permettant d'avoir une idée globale des zones les plus polluées à un temps donné. Malheureusement nous avons manqué de temps pour coder cette fonctionnalité.

L'onglet trajet, pas fonctionnel, était supposé permettre aux utilisateurs de choisir une période de mesure afin d'enregistrer un potentiel trajet effectué et de pouvoir l'afficher sur une carte.

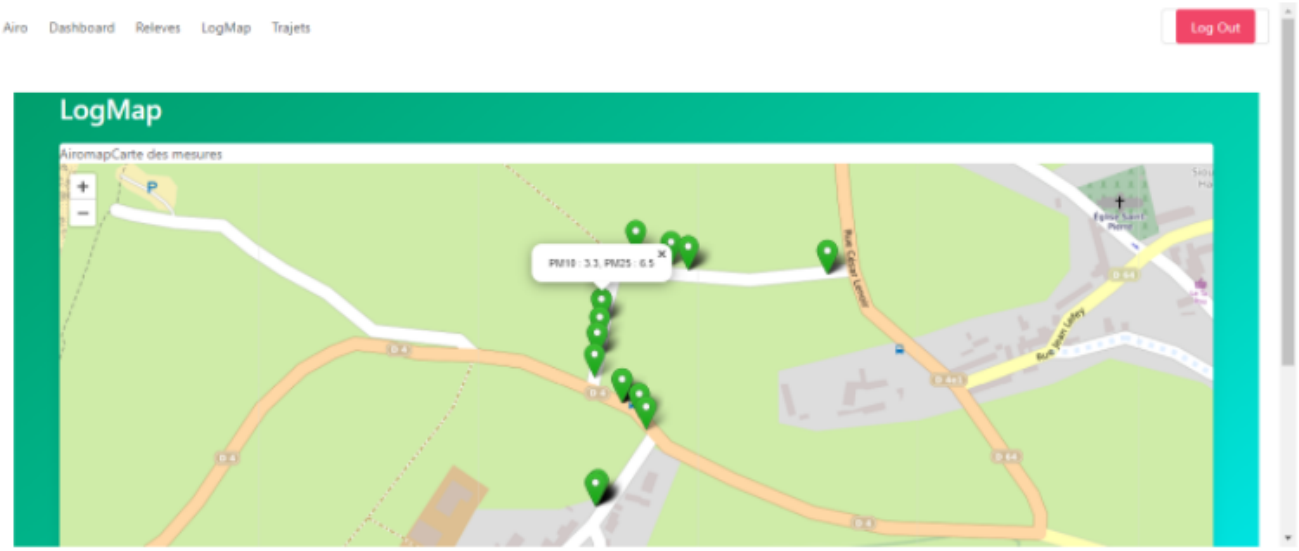
- Onglet **dashboard**



- Onglet **relevés**



- Onglet **LogMap**



PARTIE III

TESTS

Une fois les devices assemblés et la chaîne de communication complétée, nous avons pu réaliser quelques tests, dans la limite du possible avec la situation liée au distanciel. Nous nous sommes penchés sur l'autonomie des devices, avons réalisé des tests en mobilité autour de notre lieu de vie. Ce dernier test a permis de récupérer des informations sur la portée des devices, mais aussi de vérifier le fonctionnement des capteurs GPS et pollution. Nous avons également réalisé des tests avec plusieurs devices simultanément.

PARTIE III
TESTS

I | TESTS EN MOBILITÉ

Nous avons réalisé des tests avec un device, pour notamment étudier la portée de l'antenne, mais aussi vérifier le bon fonctionnement des capteurs et de l'objet en général.

Nous avons donc emmené le prototype avec nous autour de la maison où était branchée l'antenne. Nous avons parcouru quelques centaines de mètres. Après avoir récupéré les données sur Wi6labs, nous avons codé un script python (disponible en annexe B) pour afficher les différents PSNR récupérés en fonction de la distance :

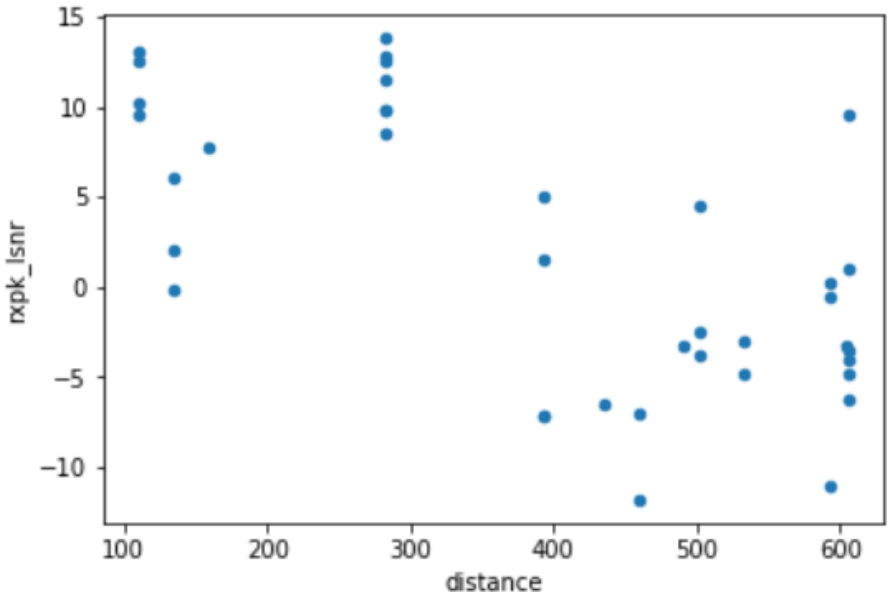


Figure 20 - PSNR en fonction de la distance

Entre 100m et 300m, nous étions pratiquement en vision directe de la gateway, ce qui explique les valeurs hautes de PSNR récupérées. Entre 300 et 550 m, nous étions dans un creux, raison pour laquelle les transmissions étaient moins bonnes, voire nulles. A 600m, nous étions presque encore en vision directe de l'antenne, raison pour laquelle le PSNR remonte.
Nous remarquons que la portée de l'antenne dépend énormément du relief et des obstacles. En effet, en théorie, la portée de notre module LoRa (qui d'ailleurs signifie Long-Range, donc longue portée) peut aller jusqu'à 40km. Mais ceci se vérifie dans des situations idéales : beau temps et vision directe entre le device et la gateway.

PARTIE III
TESTS

Dans notre cas, le relief et l'humidité de l'air influent beaucoup sur la qualité de la transmission. En ville, les immeubles vont sûrement compliquer les communications.
Si on se penche maintenant sur les données GPS et de pollution récupérées, nous avons cet affichage sur notre interface :

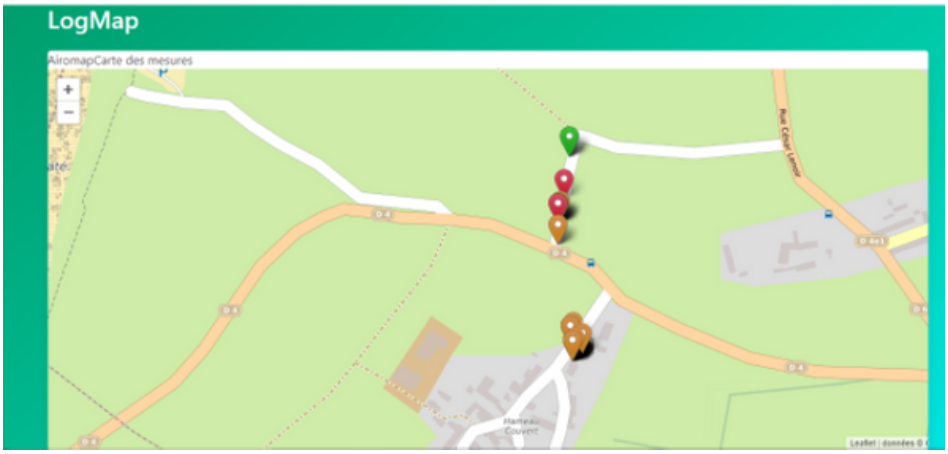


Figure 21 - Affichage des données sur l'interface

Les points oranges en bas de la carte correspondent aux données récupérées dans la maison. Elles sont assez hautes, et c'est normal : la qualité de l'air intérieur est généralement plus mauvaise que pour un espace ouvert. Les points rouges correspondent à un moment où nous avons croisé une voiture, le taux de particules fines a alors beaucoup augmenté, ce qui est assez cohérent. Le point vert, quant à lui, correspond à l'arrivée dans un chemin non emprunté par les automobiles, et aucune voiture n'était proche de nous à ce moment-là.

Pour cette session de test, nos résultats sont assez cohérents, et nous pouvons justifier nos observations. Les phénomènes observés, les données récupérées n'ont globalement rien de surprenant. Cependant, nous avons récupéré moins de mesures que ce que nous voulions. En effet, le capteur GPS n'a pas toujours réussi à récupérer les données de localisation.

PARTIE III
TESTS

II| TESTS EN SIMULTANÉ

Nous n'avons malheureusement pas pu réaliser des tests en simultané. En effet, une des 2 cartes pycom a été corrompue, ce qui a été un élément bloquant pour cette session de test. L'objectif de ce test était de :

- vérifier la récupération des données quand des trames arrivent de 2 devices différents.
- comparer les données de pollution récupérées
- gérer l'affichage des données pour différents devices sur l'interface

III | AVEC PLUSIEURS GATEWAYS

A notre retour sur Rennes, nous avons pu observer la gestion des connexions aux différentes gateways. En effet, nous avions une seule gateway avec nous hors de Rennes pendant la majorité du semestre, mais il y en a également une branchée à l'INSA en permanence. Les 2 sont enregistrées sur la plateforme wiotys de Wi6labs où nous pouvons nous connecter avec les identifiants de l'INSA. Les devices envoient les trames en broadcast, les gateways assez proches envoient alors les JOIN REQUEST aux serveurs (Network server, Join Server, App server...). Ensuite, le Join server répond ("JOIN ACCEPT") à la gateway qui a le meilleur SNR (Rapport Signal à Bruit).

Timestamp ▲	DevAddr	GatewayID	Freq	SF	MsgType
01/02/2021 : 13:48:17:963	0045e8be	0080000000016037	868.300	7	Join Accept
01/02/2021 : 13:48:16:815	0045e8be	0080000000016037	868.300	7	Join Request
29/01/2021 : 16:44:42:917	0045e8be	AA555A00004A30BB	868.500	7	Join Accept
29/01/2021 : 16:44:41:773	0045e8be	AA555A00004A30BB	868.500	7	Join Request

figure 22 - Récapitulatif des trames pour un device

Sur la figure ci-dessus, on peut voir que le 29 janvier (lignes 3 et 4), nous nous connections à la gateway qui a l'identifiant AA55------- Mais à notre arrivée sur le campus, nous avons bien reçu le "Join Accept" de la gateway située sur les toits de l'INSA (identifiant 008-------)

PARTIE IV
GESTION DE PROJET

Cette partie aborde un aspect plus logistique du projet, avec notamment un point sur le Budget. Nous y évoquerons aussi l'organisation et les blocages liés au distanciel. Enfin, nous indiquerons les améliorations et suites que nous pensons possibles.

PARTIE IV
GESTION DE PROJET

I | BUDGET

Nous avons récupéré des composants grâce à AirBreizh, au département SRC, et notre projet de l'an dernier. Afin de pouvoir réaliser plusieurs devices, nous avons réalisé la commande ci-dessous :

Composant	Quantité	Fournisseur	Prix unitaire TTC
Capteur de qualité de l'air SDS011	1	GoTronic	34,95€
Module GPS Grove 11302003	2	GoTronic	24,65€
Chargeur 4 accus NiMh R3/R6 X411	1	GoTronic	25,95€

Au total, cette commande aura coûté 110,20€ TTC.

II | GESTION DU DISTANCIEL : CE QU'ON AURAIT VOULU FAIRE

Ce semestre s'est déroulé entièrement en distanciel, ce qui a induit quelques adaptations en termes de communication et quelques révisions de nos objectifs.

Depuis bientôt un an, nous commençons à maîtriser la communication et la gestion de projet à distance. Les multiples réunions en visio se sont bien déroulées, et nous avions la chance d'avoir une partie du groupe réunie durant une partie du semestre. Cependant, c'est côté matériel que la situation a été plus difficile à gérer. En effet, le principal travail de ce projet se concentrait sur les devices, et notamment les tests d'émissions de données avec la gateway. Ce travail nécessitait donc : l'antenne, les cartes, les capteurs réunis... ce qui n'était évidemment pas possible à plusieurs endroits en même temps, et a été un élément bloquant et frustrant pour certains d'entre nous. Cependant, bien qu'ayant été freinés plusieurs fois, nous avons pu tout de même obtenir des devices fonctionnels.

L'impossibilité de nous réunir à Rennes durant l'entièreté du semestre a également empêché certains de nos plans de se concrétiser :

- La calibration des capteurs à côté d'une station AASQA était un point qui nous semblait intéressant pour le projet. En effet, afin de préciser et affiner les données de taux de particules fines récupérées par nos capteurs, nous voulions comparer ces

données pendant un certain laps de temps à celle d'une station agréée. Cette étape de calibration n'a malheureusement pas pu avoir lieu, par soucis d'emploi du temps, et de localisation des membres de l'équipe.
- La phase de test en mobilité dans la ville de Rennes n'a pas pu avoir lieu non plus. En effet, les membres de l'association Air Breizh nous avaient fait part de leurs doutes concernant la bonne couverture du réseau LoRaWAN sur Rennes. Nous avions prévu de parcourir la ville avec nos devices. Ensuite, une analyse des données récupérées nous aurait permis de visualiser les endroits de la ville ou nos émissions de données avaient le moins bon/meilleur rapport signal à bruit.

Les conséquences du distanciel auront certes été frustrantes, car nous aurions vraiment aimé réaliser certaines étapes prévues initialement, mais elles n'auront pas été désastreuses car nous pouvons quand même présenter un prototype fonctionnel.

III | AMÉLIORATIONS ET SUITES POSSIBLES

En réfléchissant à notre vision de la suite du projet, nous avons trouvé 2 axes distincts d'améliorations : l'interface et le device.

3.1 Interface

L'interface est une partie du projet qui nous semble clairement à améliorer. En effet, nous avons actuellement un affichage des données rudimentaire. Certaines fonctionnalités comme par exemple la gestion des trajets, de plusieurs devices ou l'enregistrement des relevés de pollution, sont notamment à travailler.
Actuellement, la cartographie affiche les données récupérées et les codes couleurs associés à la pollution (vert= taux de pollution faible, rouge= taux élevé). L'accès à une cartographie générale de la ville, avec potentiellement des zones à éviter serait d'après nous intéressant à coder. Il serait possible de coder un algorithme qui trace les zones de haut taux de pollution, quand il y a une concentration de "données rouges" à un endroit.
Une idée que nous avions également eue (malheureusement hors de notre domaine de compétence actuel) est le calcul d'itinéraires améliorés pour les cyclistes. Par exemple, l'utilisateur entrerait le trajet qu'il s'apprête à faire, et l'application pourrait lui donner des itinéraires de substitutions, avec des petits détours, pour qu'il respire moins de particules fines pendant son déplacement.

Côté graphisme, il serait sympathique de rendre l'interface agréable à utiliser. L'adaptation aux navigateurs mobiles, voire le développement d'une application nous semble également pertinente.

3.2 Devices

Côté device, il y a également beaucoup de possibilités d'améliorations. Certes, nos devices fonctionnent mais certains points méritent vérification et réflexion :

Niveau consommation, une optimisation est nécessaire. En effet, la gestion des différents états du device est à vérifier (veille, actif, émission...). Normalement, les objets connectés LoRa ont de longues durées de vie, jusqu'à 10 ans parfois. Nos devices méritent donc clairement une étude approfondie de leur alimentation et consommation pour atteindre ce genre de durée de vie.

Également, nos trames sont actuellement plutôt longues, et nous pensons qu'une optimisation pourrait être intéressante à faire : pertinence des données transmises, format... Cette étude des trames pourrait réduire le temps d'émission pour respecter la règle des 1%, et ainsi améliorer la consommation, mais aussi permettre d'émettre des données plus régulièrement.

Notre prototype est actuellement assez volumineux, et c'est assez normal pour un prototype. Cependant, il faut penser à l'utilisation finale qui est le placement de l'objet connecté sur un vélo. Une miniaturisation est donc nécessaire.

CONCLUSION

Ce rendu de projet marque la fin de notre cursus à l'INSA. Nous l'aurons porté et accompagné pendant 1 an et demi avec fierté et plaisir. Nous pouvons dire que nous sommes satisfaits du travail réalisé durant ce semestre malgré des conditions de travail parfois difficiles liées au confinement et au distanciel.

Nous avons atteint l'objectif de réaliser un prototype fonctionnel, allant du device à l'interface, avec les contraintes fixées par Air Breizh. Aussi, nous avons pu améliorer nos connaissances sur divers points comme par exemple la programmation sur Python, ou encore la technologie LoRa, mais aussi la gestion de projet en distanciel.

Bien sûr, nous restons disponibles auprès d'Air Breizh, voire d'éventuels repreneurs du projet si un besoin de précisions, conseils ou avis se présente. Nous exprimons notre envie que ce projet avance encore, et notre frustration de ne pas avoir pu l'emmener plus loin durant ce semestre.

TABLE DES FIGURES

SITOGRAPHIE

Airbreizh, qualité de l'air en Bretagne
www.airbreizh.asso.fr

Pycom - Next Generation iot Platform
pycom.io

Wi6labs - Solutions de réseau et d'infrastructure pour l'IoT
www.wi6labs.com

LoRa Alliance®
lora-alliance.org

L'indice de la qualité de l'air ATMO - Atmo France
atmo-france.org

Datasheet SDS011 (gotronic)
www.gotronic.fr/pj2-35853-1843.pdf

Datasheet SENSIRION
www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9.6_Particulate_Matter/Datasheets/Sensirion_PM_Sensors_Datasheet_SPS30.pdf

Datasheet module GPS (SeeedStudio)
files.seeedstudio.com/wiki/Grove-GPS/res/E-1612-UB_Datasheets_Sheet.pdf

Django
https://www.djangoproject.com

VueJS
https://vuejs.org/Tutoriel

Dashboard Django/VueJS:
https://github.com/SteinOveHelset/sknil

Exemple d'architecture site Django/VueJS
https://bezkoder.com/django-vue-js-rest-framework/

ANNEXES

ANNEXE A | CONTACTS

LE BOUHART Glenn *glebouha@insa-rennes.fr*
GUELLAEN Quentin *qguellae@insa-rennes.fr*
MEILHAT Marie *mmeilhat@insa-rennes.fr*
AGARINI Romain *ragarini@insa-rennes.fr*
BUNOUF Thibaut *tbunouf@insa-rennes.fr*

ANNEXE B | CARACTÉRISTIQUES MODULE LOPY 4

- alimentation à prévoir: 3,3 à 5,5 Vcc
- consommation:
  - WiFi actif: 12 mA
  - LoRa actif: 3 mA
  - Sigfox actif (réception/transmission): 12/42 mA
- microprocesseur: ESP32 à 160 MHz
- 8 MB de mémoire flash
- 4 MB de mémoire ram
- module horloge temps réel 32 KHz
- 8 E/S analogiques 12 bits
- 2 x bus série, I2C, 2 x SPI et I2S
- interface carte SD (sans le lecteur ni la carte)
- connexion WiFi 802.11 b/g/n à 16 Mbps (protocole de sécurité: WEP, WPA/WP2 PSK)
- module Bluetooth standard et compatible BLE
- module LoRa et LoRaWan basé sur un SX1272
- module Sigfox
- support cryptage SHA, MD5, DES, AES
- 3 connecteurs UFL pour antennes WiFi, LoRa et Sigfox
- 1 led RGB
- 1 bouton-poussoir reset
- régulateur de tension 3,3 Vcc/400 mA
- Sécurité et certifications: SSL/TLS, WPA-Enterprise, FCC-2AJMTLOPY4R et CE0700
- logiciel Pymakr compatible Windows, Mac et Linux
- dimensions: 55 x 20 x 3,5 mm (sans les connecteurs)
- poids: 7 g
- version: 4.0

ANNEXE C | NOTEBOOK PYTHON : PSNR/DISTANCE

```
#%%
import pandas as pd
from datetime import datetime
from math import sin, cos, acos, pi
import matplotlib.pyplot as pl
import plotly.express as px
filename = "WIOTYS01.csv"

def latlongen (param1, param2, orient, type):
    res = -1
    if type == 'lat':
        if 'N' in orient:
            res=float(param1) + 1/60*float(param2)
    if type == 'long':
        if 'W' in orient:
            res=-float(param1) - 1/60*float(param2)
    return res

def deg2rad(dd):
    """Convertit un angle "degrés décimaux" en "radians"
    """
    return dd/180*pi

def distanceGPS(latA, longA):
    """Retourne la distance en mètres entre les 2 points A et B connus grâce à
       leurs coordonnées GPS (en radians).
    """
    # Rayon de la terre en mètres (sphère IAG-GRS80)
    RT = 6378137
    latB=deg2rad(49.556285)
    longB=deg2rad(-1.846788)
    latA=deg2rad(latA)
    longA=deg2rad(longA)
    # angle en radians entre les 2 points
    S = acos(sin(latA)*sin(latB) + cos(latA)*cos(latB)*cos(abs(longB-longA)))
    # distance entre les 2 points, comptée sur un arc de grand cercle
    return S*RT
#%% Import & cleaning
df = pd.read_csv(filename)
df.dropna(axis=0, subset=['PayloadDeciphered'], inplace=True)
df['Date'] = pd.to_datetime(df['timestamp'], unit = 'ms')
start_date = datetime.strptime('2021-01-29 14:50:50.103','%Y-%m-%d %H:%M:%S.%f')
end_date = datetime.strptime('2021-01-29 16:50:50.103','%Y-%m-%d %H:%M:%S.%f')
newdf = df[(df['Date'] > start_date) & (df['Date'] <= end_date)]
# %%
```

```
newdf['pay']=newdf['PayloadDeciphered'].map(lambda x : bytes.fromhex(x).decode('utf-8'))
newdf[['type','pm10','pm25','lat1','lat2','lato','long1','long2','longo','h','m','s']]=newdf['pay'].str.split(',',exp
and=True)
newdf.lat1=newdf.lat1.str.strip('[')
newdf.long1=newdf.long1.str.strip('[')
newdf.lato=newdf.lato.str.strip(']')
newdf.longo=newdf.longo.str.strip(']')
newdf.h=newdf.h.str.strip('[')
newdf.s=newdf.s.str.strip(']')

newdf['fulllat'] = newdf.apply(lambda row : latlongen(row.lat1, row.lat2, row.lato, 'lat'), axis=1)
newdf['fulllong'] = newdf.apply(lambda row : latlongen(row.long1, row.long2, row.longo, 'long'), axis=1)
newdf['distance'] = newdf.apply(lambda row : distanceGPS(row.fulllat, row.fulllong), axis=1)


finaldf=newdf[(newdf['distance'] < 1000)]
finaldf
# %%
print(finaldf.keys())
finaldf.plot.scatter(x="distance", y="rxpk_lsnr")

# %%
fig = px.scatter_geo(lat=finaldf.fulllat, lon=finaldf.fulllong, color=finaldf.pm10)
fig.update_layout(geo_scope='europe')
# %%
```

ANNEXE D | CARACTÉRISTIQUES ESP32

- alimentation à prévoir: 3,3 à 5,5 Vcc
- consommation: 10 à 14 mA
- microprocesseur: ESP32 à 240 MHz
- 4 MB de mémoire flash
- 320 KB de mémoire ram
- Modules : bluetooth, Wifi, LoRa, écran Oled
- Environnement de développement ESP32 pour arduino
- dimensions: 53 x 28 x 3,5 mm (sans les connecteurs)

ANNEXE E | CARACTÉRISTIQUES NUCLEO

- alimentation à prévoir: 3,3 à 5 Vcc
- 100mA Max.
- microprocesseur: 192 KB de mémoire flash
- 4 MB de mémoire ram
- module horloge temps réel 32 KHz
- Module Lora Mbed SX1272
- logiciel Keil-uVision5
- dimensions: 134 x 70 x 7 mm (sans les connecteurs ni le module LoRa)

20 AVENUE DES BUTTES DE COESMES
CS70839
35708 RENNES CEDEX 7

WWW.INSA-RENNES.FR

LE BOUHART Glenn, GUELLAEN Quentin, AGARINI Romain,
BUNOUF Thibaut, MEILHAT Marie