Rapport final

PROJET DE REALISATION TECHNIQUE

GROSS Romain KILANI Amine
4GE2 – ANNEE 2017-2018

Plan

I. Introduction

Dans le cadre de notre projet de réalisation technique, nous nous sommes penchés sur le sujet de l'Internet des objets (IoT en anglais). Par définition, l'Internet des objets est une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution.

En d'autres termes, elle caractérise des objets physiques connectés ayant leur propre identité numérique et capables de communiquer les uns avec les autres.

Se basant sur cette idée, le sujet de notre Projet de Réalisation Technique concerne la communication sans fil du niveau de remplissage des conteneurs dits « secs » (déchets secs tels que verre, carton,…) par l'utilisation de la technique de modulation LoRa. Son faible coût d'installation et sa basse consommation en font une méthode de transmission d'information très captivant dans le contexte d'une mise en place à grande échelle.

Grâce à une mesure régulière du niveau des conteneurs, le service de ramassage de déchets peut accroître son efficacité lors de ses tournées.

L'optimisation de la collecte des déchets n'est qu'un exemple d'utilisation de notre solution. Il est évident que celle-ci peut être utilisée dans de nombreux systèmes nécessitant une mesure de remplissage et sa transmission à distance.

Ce PRT fût encadré par M.Mogniotte qui nous a bien renseignés sur le sujet avant de nous lancer dans le développement d'une expérience.

Le rapport final présenté est le support de toutes les informations recueillies durant ce semestre. Il permet principalement de renseigner sur le déroulement de notre PRT, c'est-à-dire la présentation du LoRaWAN, les choix technologiques, les problèmes rencontrés, etc.

II. Présentation du LoRaWAN

Afin de connecter des milliards de capteurs qui seront déployés dans le monde et qui participeront à la construction de ce que l'on appelle des « Smart City » ou « Ville intelligente » en français, nous pouvons utiliser ce que l'on appelle des LPWAN.

Les LPWAN sont des réseaux sans fils basse consommation, bas débit et longue portée qui sont utilisés pour des équipements dont les ressources énergétiques sont limitées et nécessitant une autonomie de plusieurs années. Une des caractéristiques des IoT est de ne pas exiger un débit élevé, les LPWAN sont donc au cœur de l'émergence de l'IoT.

Un des réseaux LPWAN est le réseau LoRaWAN (*Long Range Radio Wide Area Network*) qui constitue un protocole réseau bas débit et longue portée basé sur la technologie radio LoRa. La Société Semtech est le seul producteur de composants LoRa qui a fait l'acquisition en 2012 de la start-up française Cycléo et possède les droits d'exploitation de la technologie. La license appartient à Semtech mais d'autres fabricants de LoRa produisent également sous cette license.

La technologie LoRa opère dans les bandes de fréquences ISM (environ 868 MHz pour l'Europe et 915MHz pour le reste du monde) et utilise une modulation à étalement de spectre. Les avantages de la technologie LoRa sont majeurs, l'utilisation de cette modulation particulière permet d'optimiser les performances de portée tout en augmentant la robustesse du signal et la sensibilité du récepteur et tout cela avec une faible consommation d'énergie.

Utilisée auparavant pour les communications spatiales et militaires, cette modulation est adoptée par Bouygues Telecom et Orange depuis 2016 pour ses performances de communications longue portée et bas débit.

Logiquement, une portée plus élevée implique une consommation d'énergie plus importante puisque la portée est déterminée par la bande passante, la puissance de sortie du signal ainsi que le Spreading Factor (Facteur d'étalement). L'étalement du signal augmente sa portée au détriment du débit car il est transmis sur une plus longue période et donc au détriment de l'autonomie de l'équipement.

La Figure 1 propose un schéma illustrant l'impact de l'étalement du signal sur la portée et le débit d'une modulation LoRa sur une bande européenne 868MHz.

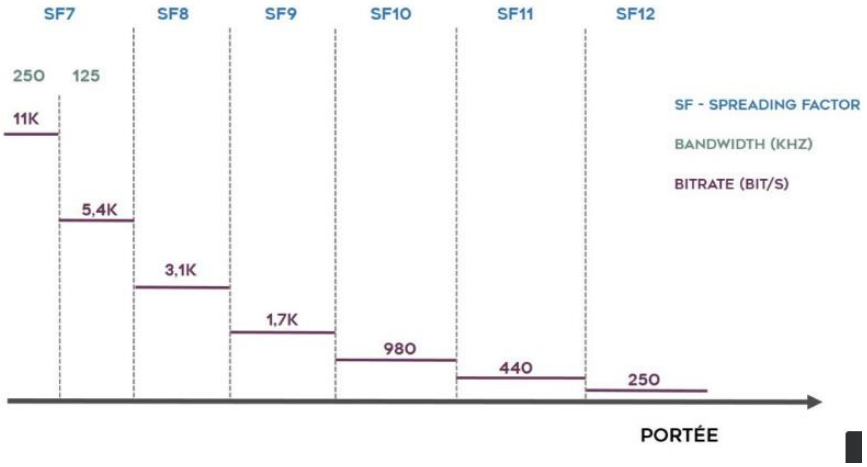


Figure 1 : Impact du SF sur la portée et le débit

Un réseau LoRaWAN qui s'appuie sur une modulation LoRa dans la bande 868 MHz présente 6 facteurs d'étalement (SF7, SF8, SF9, SF10, SF11, SF12). Le réseau doit supporter au minimum les trois canaux suivants (de 125 kHz de bande passante chacun) : 869.10 ; 868.30 et 868.50 MHz.

Semtech propose un outil intéressant qui permet de calculer les performances estimées d'une communication LoRa en fonction de plusieurs paramètres (SF, bande passante, fréquence, etc.).

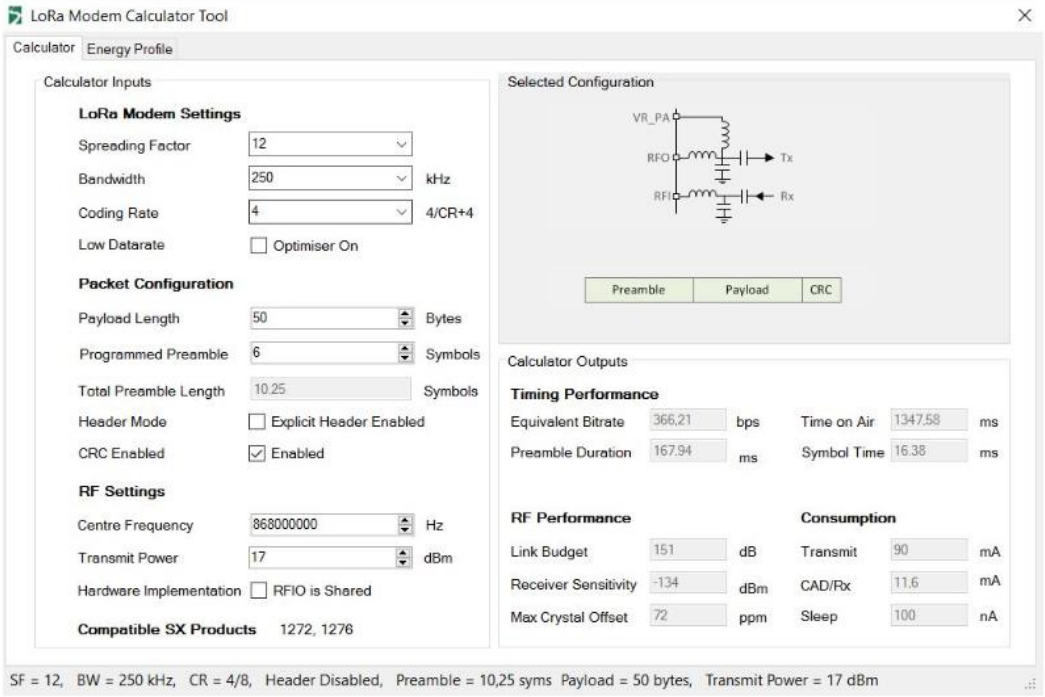


Figure 2 : Outil Semtech

Dans le bien connu modèle OSI, la technologie LoRa représente donc la couche physique d'un réseau LoRaWAN, définissant le lien radio entre les équipements et les concentrateurs appelés respectivement les *end-devices* et les *gateways* dans les spécifications LoRaWAN.

Afin que deux équipements communiquent à travers un réseau, il est nécessaire de définir un protocole réseau. Ainsi, LoRaWAN définit le protocole réseau LoRa MAC. Les différentes couches du réseau LoRaWAN sont représentées à la Figure 3.

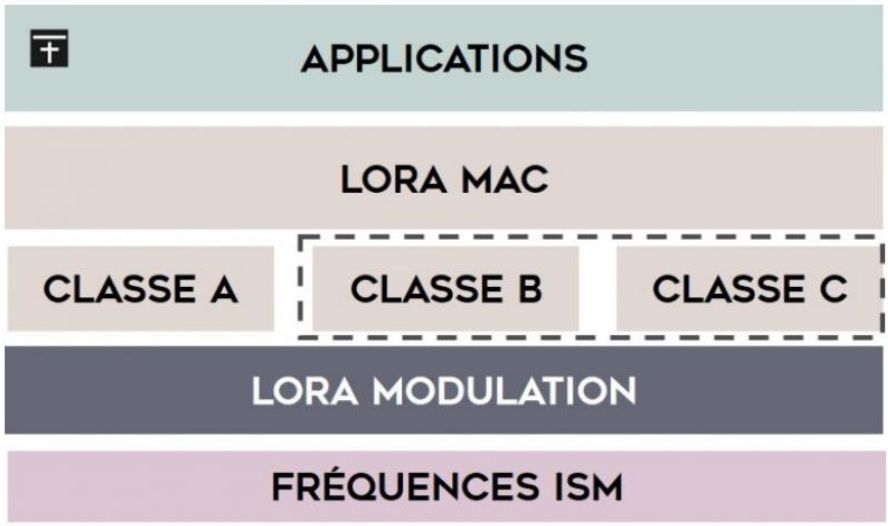


Figure 3 : Couches d'un réseau LoRaWAN

En s'appuyant sur les performances de la modulation LoRa, le protocole LoRaWAN assure des communications bidirectionnelles et définit ainsi trois classes d'équipements : Classe A, B et C. La différence entre ces classes réside essentiellement dans le nombre de fenêtres alloués par l'équipement pour la réception des messages envoyés par le réseau.

Le protocole LoRaWAN prévoit un débit adaptatif compris entre 0.3 et 50 kbits par seconde.

Architecture d'un réseau LoRaWAN

Le réseau LoRaWAN présente une topologie en étoile puisque les équipements (*end-devices*) communiquent en LoRa avec des concentrateurs (*gateways*). Les concentrateurs centralisent les messages reçus par les équipements pour les transmettre au serveur de gestion du réseau. La liaison entre les concentrateurs et le serveur se fait par des technologies très haut débit (Ethernet, 4G..).

L'architecture du réseau LoRaWAN est résumée à la Figure 4.

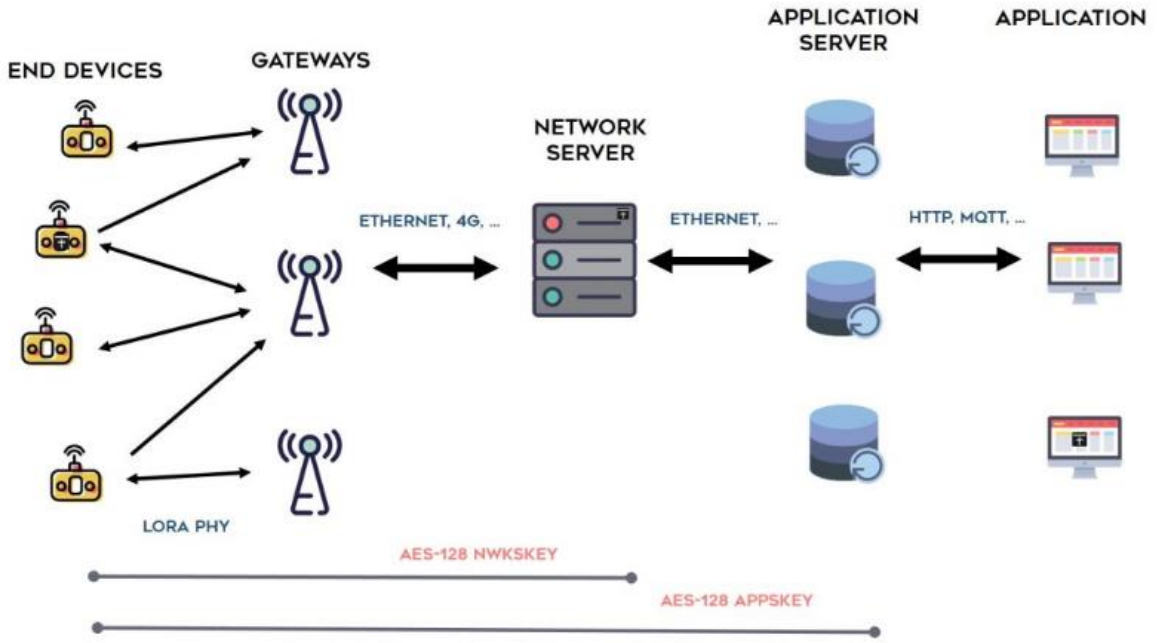


Figure 4 : Architecture du réseau LoRaWAN

Applications du réseau LoRaWAN

Compte tenu de ses performances en termes de portée et d'autonomie des équipements, le réseau LoRaWAN permet le déploiement et la remontée de données de capteurs en tout genre (qualité de l'air, remplissage de poubelles, détecteur de présence, …). L'exploitation de ces données ouvre des perspectives considérables dans de nombreux domaines que ce soit pour la gestion des ressources, pour l'optimisation des déplacements ou encore pour l'augmentation de la productivité.

Notre expérience ainsi que notre premier rapport (Plan Directeur de Projet) se sont uniquement basés sur l'utilisation pour les remplissages des poubelles.

III. Présentation du système et de la chaîne d'information

Cette partie présente la problématique de la collecte des tournées en proposant une solution basée sur la technologie LoRa.

Lors du PDP, nous n'avons pas abordé les architectures du réseau LoRaWAN mais simplifié le système. Cela donnait donc le schéma Figure 5.

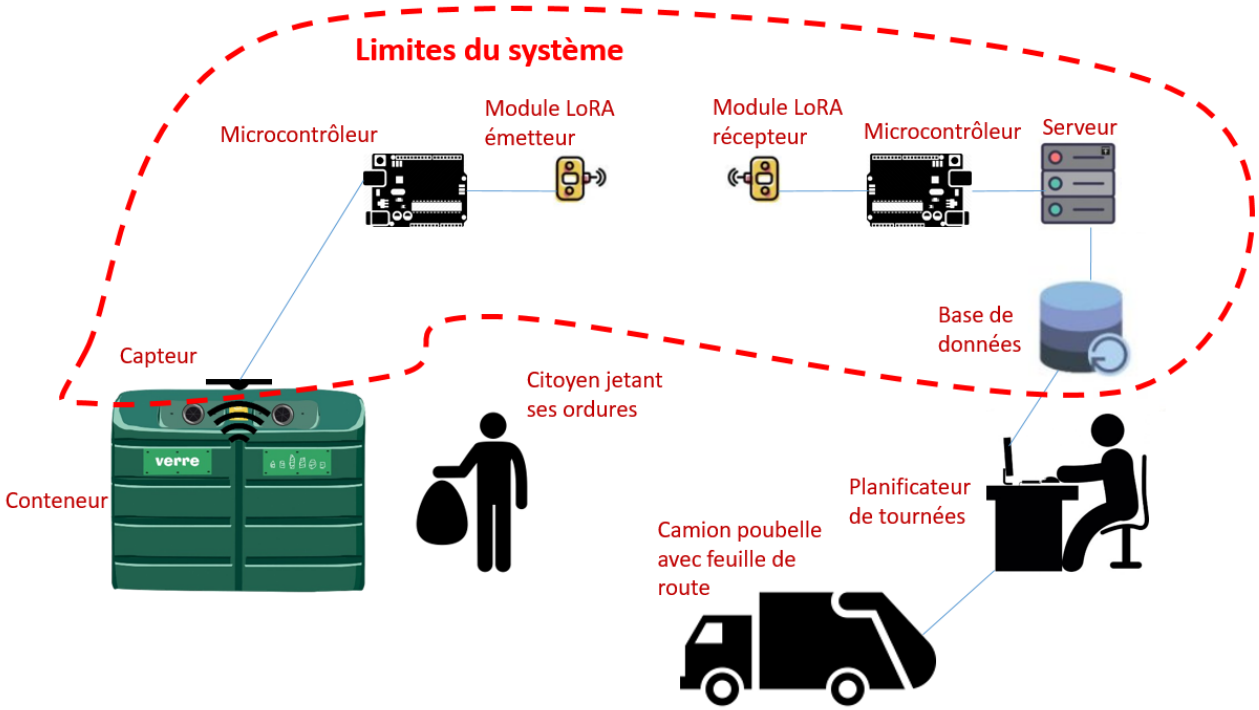


Figure 5 : Système présenté sur le PDP

La problématique d'aujourd'hui est que les camions poubelles font souvent des tournées « inutiles » où le conteneur est encore vide. La situation d'un conteneur trop plein car le camion n'est pas passé au bon moment est aussi un scénario fréquent. C'est ainsi qu'intervient notre système qui permet de créer ce que l'on appelle le conteneur intelligent répondant aux attentes d'une Smart City.

Le système rend donc service à deux utilisateurs qui sont les sociétés de ramassage des déchets ainsi que les citoyens désirant jeter leurs déchets (Voir PDP pour plus de détails).

Comme nous pouvons le constater, notre système représente le conteneur à déchets où l'on vient placer un capteur à ultrasons qui mesurera le taux de remplissage, enverra l'information au microcontrôleur qui le transmettra au module LoRa afin que celui-ci puisse émettre ces données à distance. Les données seront captées par le second module LoRa et transmis à un serveur par l'intermédiaire d'un second microcontrôleur pour pouvoir être stocké dans une base de données.

Notre système ainsi défini, nous souhaitons réaliser un prototype à petite échelle sans concentrateur pour notre PRT. La liste du matériel se compose donc de :

-1 Capteur à ultrasons (ou autre capteur de distance)

-2 Microcontrôleurs

-2 Modules LoRa

-1 Logiciel de programmation

-1 Serveur et base de données

IV. Choix technologiques

Dans notre projet, nous aurons besoin de deux unités “intelligentes” pour quatre fonctions principales distinctes:

- acquérir l’information: en l'occurrence, connaître le taux de remplissage d’un conteneur.
- transmettre cette information: on choisit la modulation LoRa pour ce faire.
- réceptionner cette information
- retranscrire cette information pour qu’elle soit interprétable par un opérateur humain: nous avons fait le choix d’un affichage au travers d’un graphique qui évolue en fonction des différents niveaux reçus dans le temps.
-

D’autres fonctions en plus seront adjointes à ces dernières. Elles seront explicitées plus tard dans leurs parties respectives.

1. Acquisition des données

Pour obtenir le niveau de remplissage d’un conteneur, nous avons fait le choix technologique de l’ultrason: cette méthode marche pour connaitre la distance entre le capteur et un obstacle placé face à lui. Ici, nous n’aurons qu’à positionner ce capteur sous le couvercle du conteneur et le mettre face au fond de ce dernier afin de connaître la distance entre le haut de la poubelle et le haut de la pile de déchets.

Nous avons fait le choix d’un capteur à ultrasons bas de gamme très utilisé : le HC-SR04.

Figure 6 : Capteur à ultrason HC-SR04

En bref, pour déclencher une mesure, il faut présenter une impulsion « high » (5V) d’au moins 10 uS sur l’entrée « Trig ». Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz puis attend le signal réfléchi. Lorsque celui-ci est détecté, il envoie un signal « High » sur la sortie « Echo » dont la durée est proportionnelle à la distance mesurée. La plage de mesure du HC-SR04 est de 2cm à 400cm avec une résolution de la mesure à 0.3cm ce qui serait largement suffisant pour notre utilisation.

Le capteur donne une information interprétable par un microcontrôleur. Nous avons choisi d’utiliser un Arduino UNO pour sa simplicité de programmation et du fait

que nous en disposions de deux avant le début du projet. Nous attribuons ainsi deux pins du premier Arduino pour la gestion du capteur.

Figure 7 : Carte électronique Arduino UNO

2. Transmission des données

Une fois la mesure sauvegardée sur l'Arduino, il faut transmettre cette information à la partie "serveur". Pour ce faire, nous avons fait le choix technologique de la modulation de signal LoRa. Pour la réaliser, il nous faut un module LoRa qui réalise cette tâche ainsi que son antenne adaptée.

Après recherches, le meilleur compromis qui nous a paru fût de commander directement depuis la Chine via Aliexpress. Le choix de la puce a été le SX1276 développé depuis peu par Semtech, qui nous permet la modulation et démodulation du signal pour une dizaine d'euros par composant. Ce chip est relativement abordable pour sa capacité: il annonce une portée de 3000 mètres pour 100mW théorique. Les antennes associées ont un gain de 3dBi.

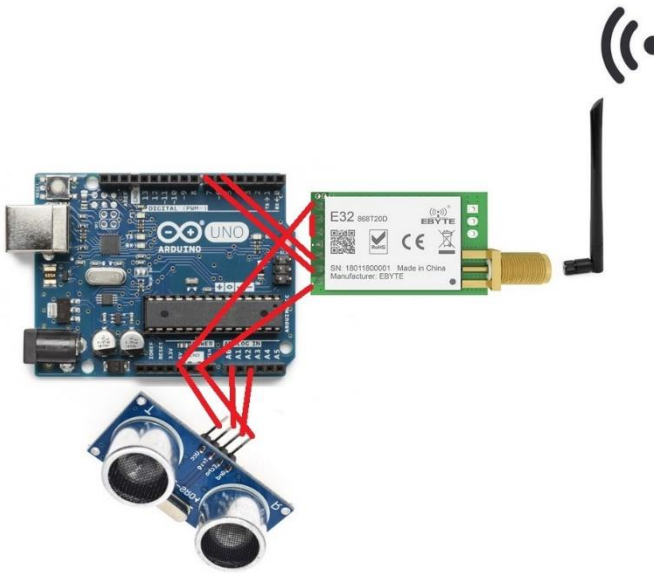


Figure 8 : Partie transmission des données

Dans cette partie de la chaîne d'information, nous récupèrons la distance entre le niveau haut des déchets et le couvercle du conteneur et nous la transmettons au module LoRa pour l'émission. Pour ce faire, nous avons recours à deux fonctions:

- *void mesure(float *mesure)*

Envoie une impulsion de 10µs sur la broche allouée au pin TRIGGER du capteur.

Mesure le temps parcouru de l'onde ultrasonique revenue sur le capteur ECHO qui donne une impulsion en retour pour être détectée.

Nous avons fait cela en appelant les fonctions

digitalWrite(TRIGGER_PIN, HIGH);

delayMicroseconds(10);

et

duration = pulseIn(ECHO_PIN, HIGH);

En prenant en considération la vitesse du son dans l'air de 340 m/s, on en déduit la distance:

d= v *duration

On applique à cette variable une fonction floor pour arrondir le résultat donné en millimètres.

Nous avons pris le soin de passer en paramètre un pointeur dans la fonction afin de pouvoir changer la valeur envoyée sur le module LoRa dans la boucle loop.

- *mySerial.write(percent)*

Dans la boucle loop, nous décidons d'écrire sur le port série toute les secondes (dans le cadre du prototype, tout en sachant que la réalité n'est pas aussi exigeante) la nouvelle valeur de la distance mesurée par la fonction mesure. Ceci est effectué en mettant un delay(1000).

3. Réception des données

Nous utilisons le même dispositif (excepté le capteur ultrason) sur un autre Arduino pour recevoir le message envoyé. Cet Arduino sera relié à un ordinateur en USB qui pourra accéder à ses valeurs reçues.

Cet ensemble de valeurs sera stocké sur un fichier .txt qui servira de base de données pour un programme sur le PC qui affichera les valeurs stockées chronologiquement.

Figure 9 : Partie réception des données

Nous avons créé une liaison série RX/TX sur deux pins de l'Arduino via un objet ***SoftwareSerial***. Dès que le module est disponible (état "available" de la liaison), on stocke la valeur donnée par le module dans une variable XXX et on envoie la commande *"#S|SEND|[**XXX**;]#"* sur la liaison série USB faite avec l'ordinateur.

Création de la database

Pour la création d'une base de données, nous nous servons d'un programme existant: Gobetwino. Ce logiciel permet à l'ordinateur relié à l'Arduino récepteur du message LoRa de lui envoyer des commandes et d'en recevoir sur son port USB.

Ici, on récupère la valeur dans le SEND et on la stocke dans un fichier .txt dont on aura précisé le chemin sur le disque dur du PC sur le logiciel Gobetwino.

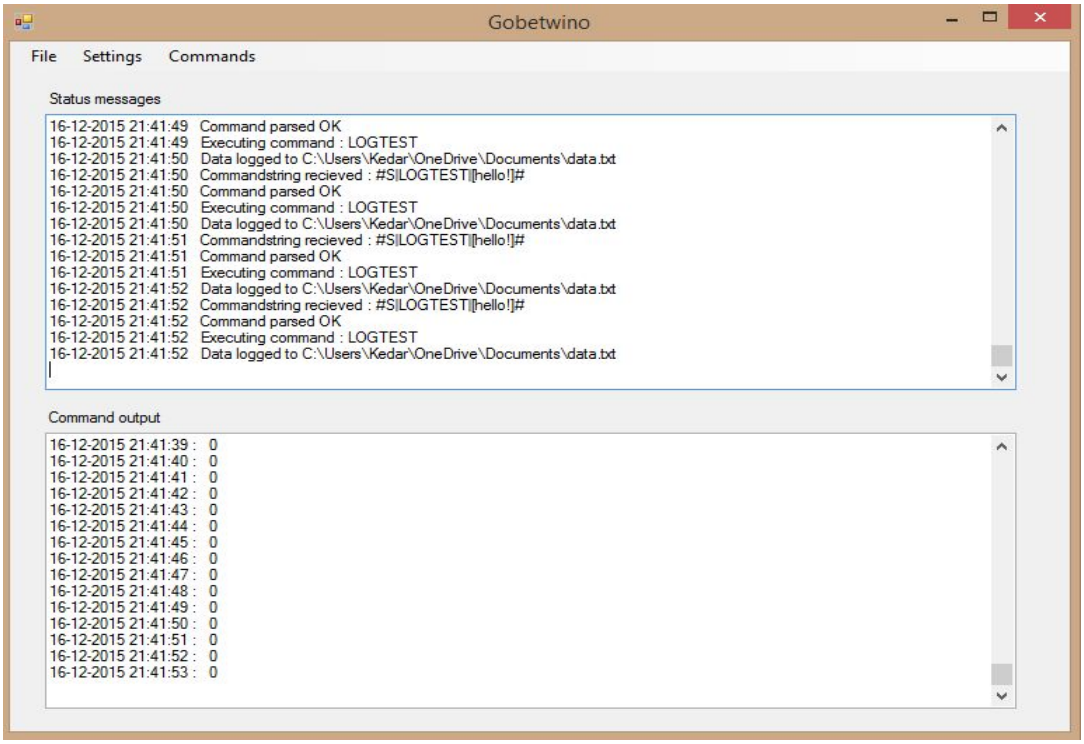
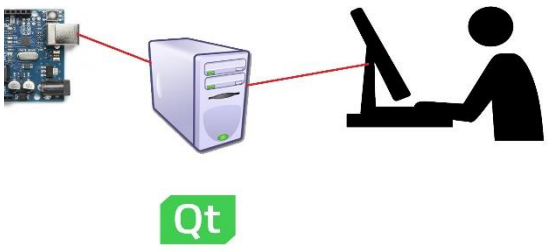


Figure 10 : Visualisation du logiciel Gobetwino

Serveur et Interface graphique



Pour rendre la lecture possible des différentes valeurs obtenues par un utilisateur, nous avons développé une interface graphique propre à notre application.

Pour ce faire, nous nous sommes penché sur le langage Qt qui permet l'utilisation de classes prédéfinies dans ses bibliothèques afin d'afficher ce que nous voulions. Le résultat final est illustré ci-dessous.

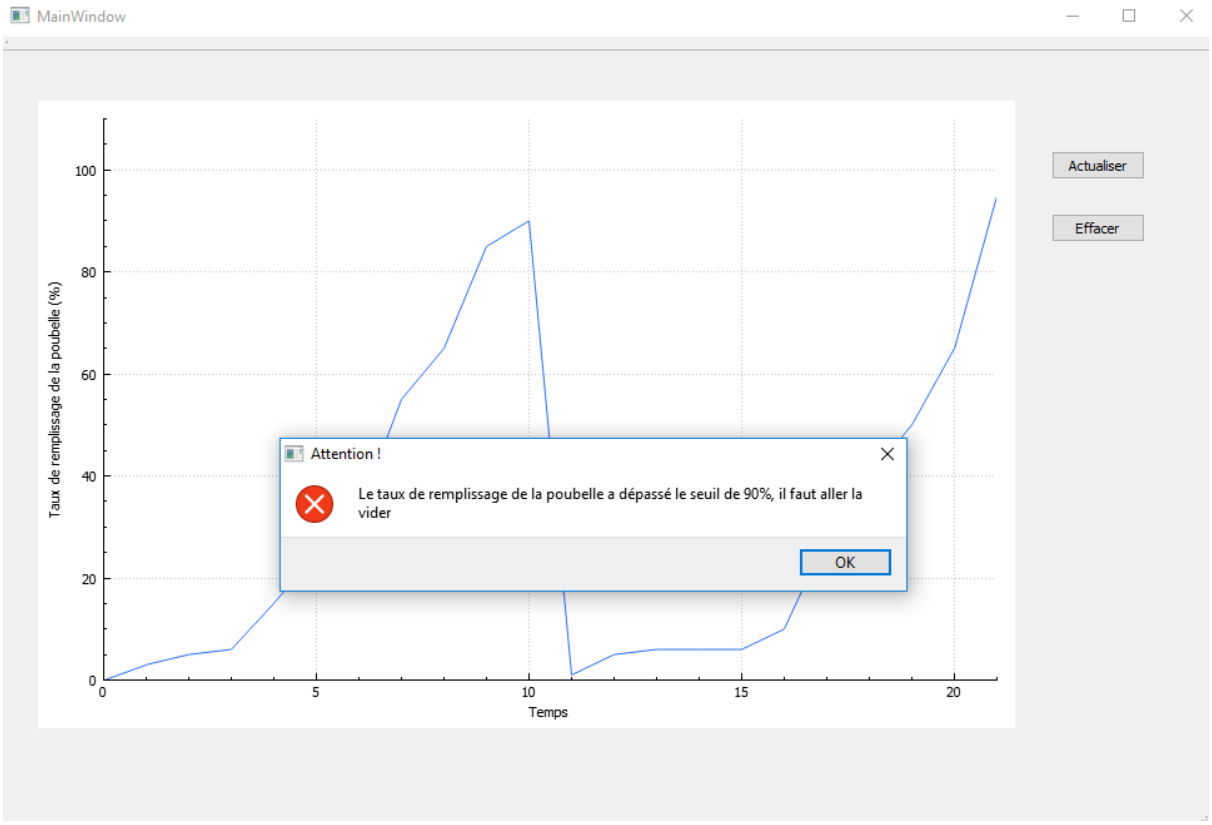


Figure 11 : Interface graphique avec affichage d'erreur

Nous avons eu recours à la classe *QCustomPlot* afin d'avoir le graphique de suivi, et aux différents QWidget déjà présent dans la bibliothèque standard de Qt pour faire la fenêtre principale ainsi que les boutons "Actualiser" et "Effacer".

"Actualiser" permet de mettre à jour manuellement les dernières mesures reçues sur la base de données.

"Effacer" permet de supprimer sur la base de données toutes les mesures faites.

Lorsque le conteneur a atteint 90% de sa limite de stockage, un message apparaît sur l'écran de l'utilisateur l'informant qu'il est bientôt plein.

Programmation

C'est depuis l'interface Qt de la Figure 12 que l'on met en forme la fenêtre que nous souhaitons obtenir à la fin. Il est possible d'y définir la taille de la fenêtre, les différents boutons et zone d'affichage par un système de drag and drop intuitif. Cette interface génère un fichier XML à l'extension ".ui" pour user interface appelé dans le constructeur de la classe mainwindow.

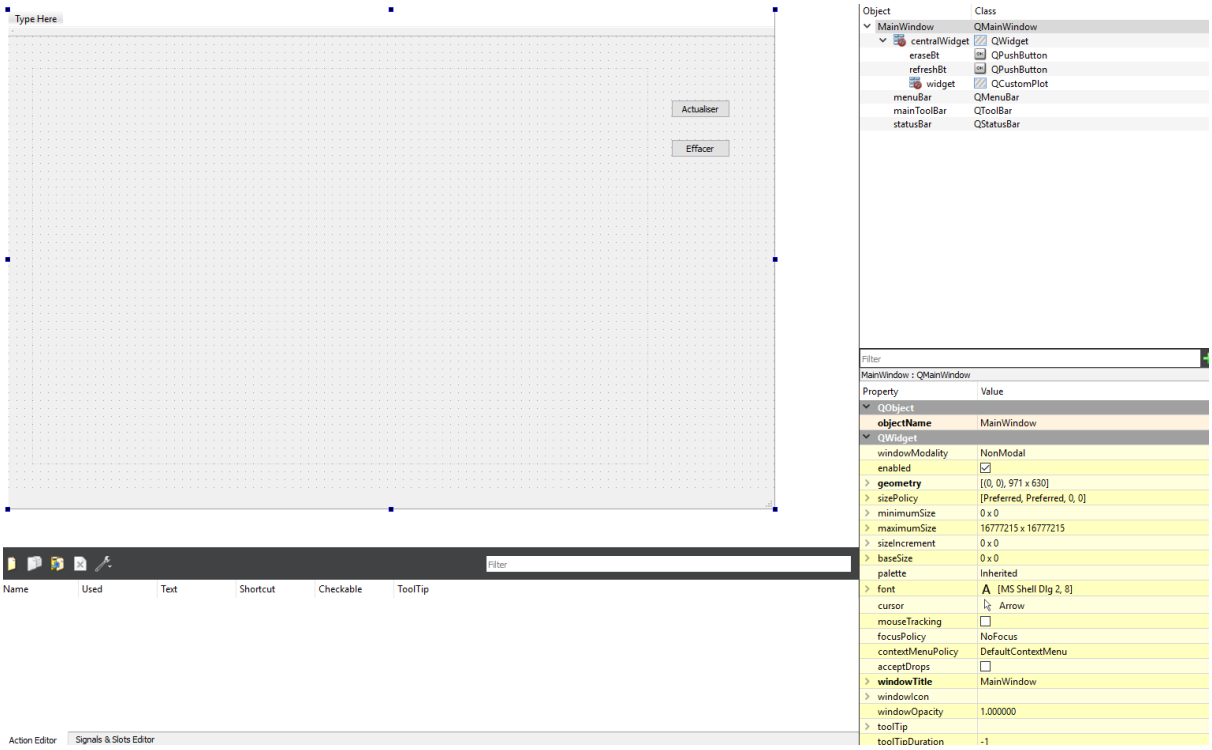


Figure 12 : Interface Qt Creator

Le système Signal/Slot

Il faut savoir que nous utilisons des objets issus de classes prédéfinies avec des propriétés particulières pour certaines. Nous nous pencherons sur le cas des QPushButton. Nous aimerions lors d'un clic actualiser l'affichage fait par QCustomPlot. Le langage Qt prend tout son sens dans ce type d'application: il sait gérer la notion "Signal-Slot".

L'objet QPushButton affiché sous la forme d'un bouton "Actualiser" par exemple. Lors d'un clic dessus, un des attributs de cette classe passe de false à true. Cette variable est appelée ***signal***. Si dans le constructeur de la classe mainwindow, on a connecté ce signal à une fonction particulière (appelée ***slot***) d'une autre classe comme l'actualisation de l'affichage de QCustomPlot, alors l'appui du bouton provoque la mise à jour du graphique.

Et ceux indépendamment du temps t auquel on effectue cette opération. Elle peut être répétée plusieurs fois sans encombrer le déroulement du programme.

Les fonctions implémentées

Pour y arriver, nous avons ajouté plusieurs fonctions à la classe "mainwindow" prédéfini lors de la création d'une application.

- MainWindow::addPoint()

Elle se charge de stocker dans un tableau tous les points sauvegardés sur la database sur l'objet de type QCustomPlot.

Cette fonction est une fonction slot. Elle est connectée au signal "clicked(bool)" du bouton "Actualiser".

Elle fait un accès au fichier database grâce à la classe QFile qui permet de lire ou d'écrire dessus. Ici, on n'utilise que la fonction "readAll()" de cet objet QFile afin d'en retirer la chaine de caractère sauvegardée.

Chaque mesure est séparée d'un ";". On remplit un tableau QStringList de string de type QString de ces valeurs par une fonction "split()" de la classe QStringList. Cette fonction se charge de séparer toutes les données et de les allouer dans une case spécifique de l'objet QStringList. Nous convertirons en double ces QString obtenues par la fonction "toDouble()" de la classe QString.

Une fois les distances enregistrées, nous faisons un bref calcul pour connaître le taux de remplissage de la cuve pour chaque distance mesurée. Il suffit de connaître la profondeur initiale du conteneur et de faire le calcul suivant:

$$Taux\ de\ remplissage = \frac{profondeur\ initiale - distance\ mesurée}{profondeur\ initiale} * 100$$

Une fois ces opérations effectuées, elle appelle la fonction "MainWindow::plot()" définie plus bas.

Elle vérifie si le dernier point enregistré a une ordonnée supérieur 90. Cela signifierait que le conteneur est bientôt saturé. Dans ce cas, elle appelle la fonction "warning()" définie plus bas.

- MainWindow::erase()

Elle fait un accès au fichier de base de donnée par la classe QFile et supprime son contenu par la fonction "flush()".

Cette fonction est une fonction slot. Elle est connectée au signal "clicked(bool)" du bouton "Effacer".

- MainWindow::plot()

Elle affiche les points sauvegardés par la fonction addPoint().

Elle place ses points grâce à la méthode setData(QVector<double>,QVector<double>) de la classe QCustomPlot, qui donne les valeurs des abscisses et ordonnées des différents points enregistrés.

Elle réajuste l'échelle des axes pour une meilleur lecture grâce la fonction "rescaleKeyAxis()" de la classe QCustomPlot.

"replot()" et "update()" définies dans QCustomPlot se chargent d'effacer les anciennes valeurs et d'afficher les nouvelles sur le graphique.

- MainWindow::warning()

Elle affiche une nouvelle fenêtre avec le message

"Le taux de remplissage de la poubelle a dépassé le seuil de 90%, il faut aller la vider"

Pour cela, elle crée un objet de type QMessageBox et l'instancie en "critical".

Après avoir présenté notre prototype nous avons donc dû le tester.

V. Présentation de l'expérience

Nous avons effectué deux petites expériences avec notre système.

1. Première expérience

Pour notre première expérience, nous ne voulions pas tester la communication longue distance mais plutôt le bon fonctionnement du code, des microcontrôleurs, des modules LoRa et du capteur. Ainsi, les deux modules LoRa étaient à proximité, le capteur placé sur une corbeille dont nous faisions varier le remplissage avec des objets quelconques.

Nous pouvons observer une photographie de l'expérience à la Figure 13.

Plusieurs problèmes se sont présentés lors de l'élaboration de cette expérience et seront mentionnés dans une partie ultérieure. Une fois les problèmes résolus, nous étions surpris d'observer à quel point les mesures étaient justes.

Les 5 mesures effectuées sont reportées dans le tableau de la Figure 14. Nous pouvons en conclure que les résultats sont plus que satisfaisants (considérant que pour notre application, même 3cm d'erreur de mesure serait largement acceptable).

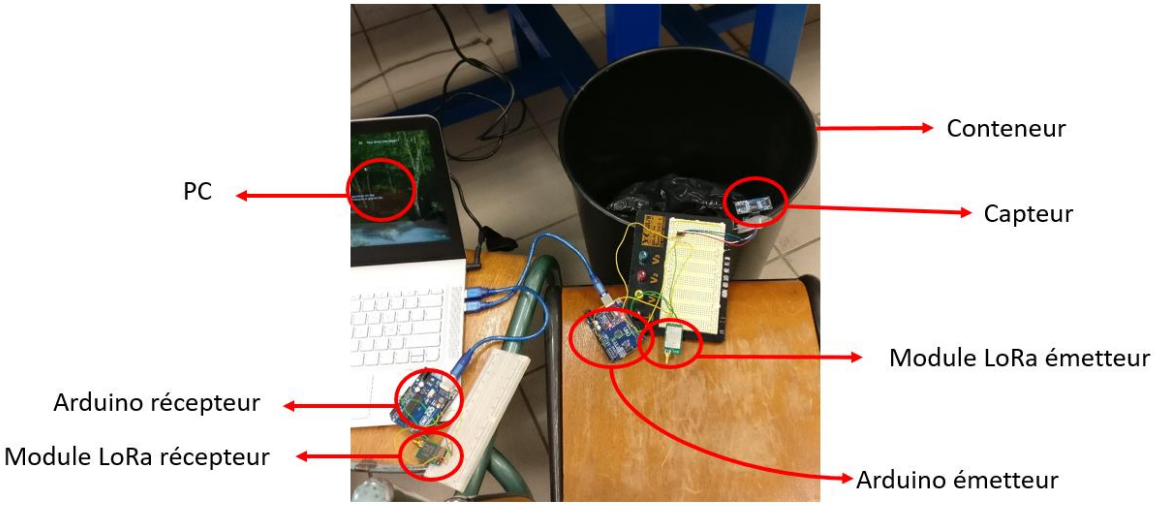


Figure 13 : Photographie légendée de l'expérience 1

Mesure réelle distance (cm)	Mesure capteur (cm)	Erreur de mesure (cm)
55	53,5	1,5
35	36	1
20	21	1
12	53,5	1,5
7	7	0

Figure 14 : Résultat des mesures

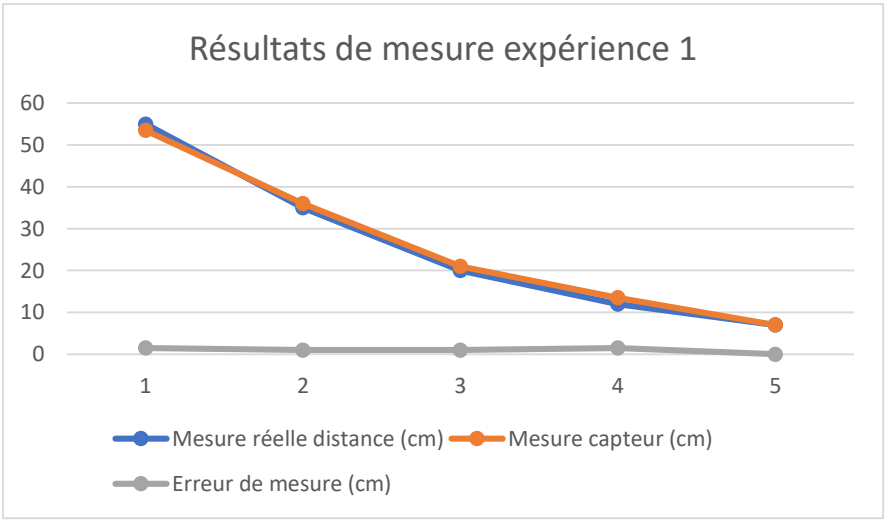


Figure 15 : Représentation graphique des résultats

2. Seconde expérience

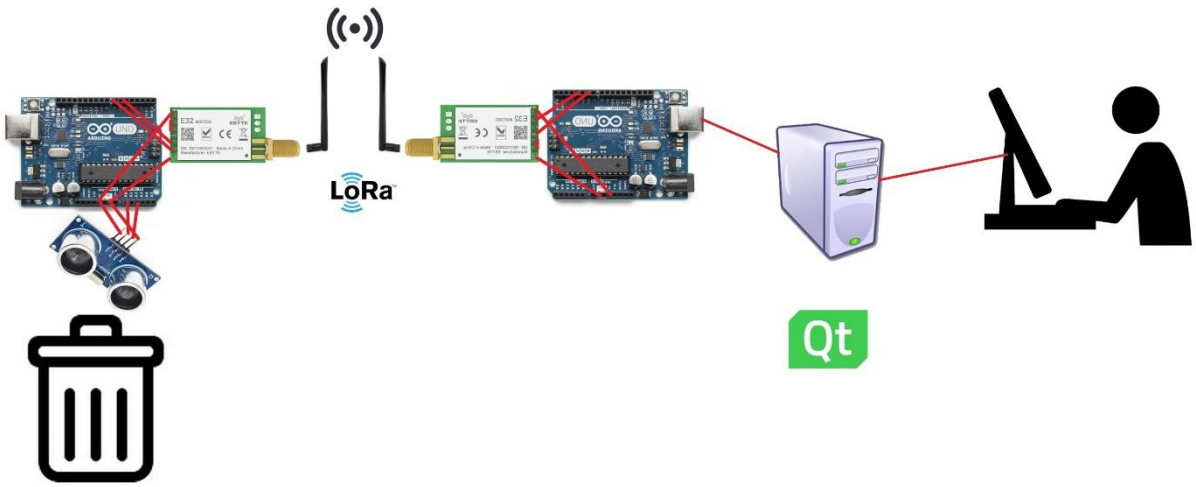


Figure 16 : Schéma global de l'expérience

La seconde expérience consistait simplement à tester la portée de notre communication avec les deux antennes. Pour cela, nous avons utilisé un PC portable pour le côté récepteur et une batterie externe pour alimenter le côté émetteur.

Ensuite, il suffisait que l'un d'entre nous marche le plus loin possible jusqu'à ce que nous puissions plus recevoir les données. Grâce à un calcul de la distance à vol d'oiseau sur Maps, nous avons trouvé une portée maximale de 879m ce qui est très satisfaisant pour des dispositifs si peu chers.

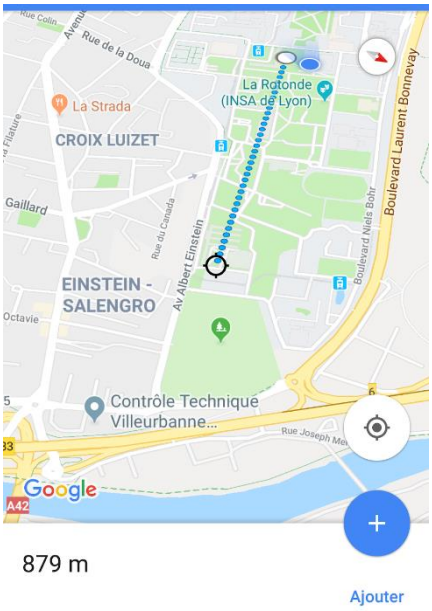


Figure 17 : Calcul de la portée maximale

VI. Problèmes rencontrés

Ecran LCD

Nous avons élaboré les différents programmes de chaque Arduino avant l'arrivée des modules LoRa. Notamment, nous avions à disposition un shield LCD qui s'emboîtait sur les pins de l'Arduino. Nous avions programmé ainsi un menu déroulant sur cet écran, sur lequel était affiché les informations suivantes : distance mesurée - taux de remplissage - profondeur du conteneur. Nous aurions vérifié directement ces valeurs depuis l'Arduino émetteur, sans passer par la transmission LoRa.

Cependant, nous n'avions plus de pin disponible pour une liaison série RX/TX.

Nous avons donc abandonné l'idée d'utiliser cet écran.

Antenne adaptée manquante

Lors de la commande faite sur Aliexpress, il était écrit sur l'article qu'une antenne adaptée était fournie avec le module. Or après 3 semaines, à la réception les antennes étaient manquantes. Il aura fallu attendre 3 semaines de plus pour les recevoir.

Ces temps de livraison longs ont bien changé notre calendrier lors des phases d'essai de notre système.

Capteur à ultrason défectueux

Lors de la première expérience, nous ne comprenions pas pourquoi nous avions des valeurs totalement fausses. Nous avons revérifié le code entièrement. Par chance, Romain s'est rappelé que lors de la commande des capteurs, deux d'entre eux ont été livrés. Nous avons donc pu remplacer le capteur et l'expérience fonctionnait à merveille.

C'est une chance d'avoir pensé à commander deux capteurs au lieu d'un car si ce n'était pas le cas, nous aurions dû en recommander un autre et au vu des temps de livraison, nous n'aurions pas pu effectuer notre expérience à temps.

Délai de livraison

Ce problème étant surement le problème de tous les PRT, lorsque nous commandons des composants en Chine, la livraison met plusieurs semaines. Il fallait donc tout prévoir pour que nous puissions avoir tous les composants en une commande, et non défectueux ! Sachant que le risque est très élevé puisque le matériel provient de Chine (ça n'a pas raté avec le capteur HC-SR04).

VII. Conclusion et perspectives

Ce Projet de Réalisation Technologique est un projet très enrichissant pour la 4ème année de GE. Il permet d'acquérir une certaine autonomie et d'agir en ingénieur face à un cahier des charges. De plus, une organisation optimale ainsi qu'un travail d'équipe sont primordiaux pour le bon déroulement du projet ce qui se rapproche du futur métier que nous allons incarner.

Concernant notre sujet, nous avons eu la satisfaction d'être arrivé à nos fins en ayant fait une grande partie de ce que nous voulions réaliser au départ. Les résultats de nos expériences ont confirmés la faisabilité de cette solution qui semble primordiale à considérer et améliorer pour le futur.

Quelques solutions d'optimisation ont été pensées pour notre projet mais non réalisé par choix techniques, financiers ou de temps.

Lors de la conception de l'affichage, nous nous sommes arrêtés au fait d'actualiser manuellement le graphique: un opérateur doit cliquer sur le bouton "Actualiser" pour rafraîchir l'affichage de la base de données. On aurait pu éventuellement faire évoluer le graphique juste en fonction de la base de données qui s'actualise. Cependant, après plusieurs recherches sur les signaux des objets de type QFile, nous n'avons pas trouvé de solution adapté à cette question. Il aurait fallu se pencher sur une autre classe Qt.

Lors de la conception de notre prototype, nous avons fait une communication appelée de "point à point", ne nous souciant pas de la conception d'un réseau LoRaWAN. Il aurait fallu :

- soit développer nous même un concentrateur LoRa, afin d'adresser des adresses à chaque device LoRa reconnues par ce dernier. Nous serions passés par le site https://www.thethingsnetwork.org/ pour partager l'adresse à diffuser de ce concentrateur.
- soit souscrire à un abonnement Orange-Business qui propose des solutions LoRa pour professionnel et obtenir les données cloudées sur son site directement

Pour une miniaturisation du système plus poussée, nous aurions pu remplacer l'Arduino côté réception par un Raspberry Pi qui en plus de recevoir les données de son module LoRa, aurait enregistré dans un fichier base de données directement sur sa mémoire.

Raspberry Pi dispose de la bibliothèque Qt par défaut. Il aurait été très facile d'avoir l'affichage directement en sortie vidéo depuis le Raspberry.

Pour pousser la réflexion plus loin, on aurait pu se servir également de ce Raspberry comme serveur pour héberger un site web. En louant un nom de domaine, on aurait pu rendre accessible les données reçues depuis Internet.

Bibliographie

Sites internet :

[1] https://www.frugalprototype.com/technologie-lora-reseau-lorawan/

[2] https://www.carcnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/

[3] https://letmeknow.fr/blog/2015/10/27/tutomodulelora/