

DOSSIER TECHNIQUE

PARTIE COMMUNE

2^{ème} année BTS système et numérique
Option Informatique et Réseau

Projet Kolantr

Etudiant : Xavier HERVIER, Thibaut JOUANDOUDET, Hugo CARRICART, Tony TESTA, Arthur LABARRE, Théo SERNOT, Armand HOUPIN

Professeurs référents : Patrice SPEZIALI – Eric PERRUCHE



I. Table des matières

Contexte	4
Objectifs du projet	4
Matériel imposé pour solutions	5
Contraintes techniques	5
Spécifications Générales	5
Sous Système de Consultation :	10
Scénarios	10
Modèle de classe du domaine :	11
.....	11
Description de l'architecture du système :	12
Architecture matérielle	12
Diagramme de séquence :	13
Architecture Logicielle :	14
Recette sous-système simulation (local) :	14
RECETTE XAVIER COLLECTE	15
RECETTE ARMAND CONSULTATION	15
Répartition du travail par étudiant	17
DÉPLOIEMENT ou RECETTES / VALIDATION	18
1. Site de consultation (Armand)	18
CONCLUSION Partie Commune	19
Bilan technique Site de consultation (Armand)	19
Bilan technique IHM sur E-paper (Xavier)	19
PARTIE PERSONNELLE ...	
I. Etudiant Xavier HERVIER	22
A. Description partie personnelle	22
B. Conception détaillée	23
C. Implémentation.....	30
D. Test	38
E. Bilan technique.....	40
F. Partie Physique	40
II. Etudiant Thibaut Jouandouet	43
A. Description partie personnelle	43
B. Conception détaillée	44
C. Implémentation.....	48
D. Test	49
E. Bilan Technique.....	55
F. Partie Physique	55
III. Etudiant Hugo CARRICART	60
A. Description partie personnelle	60

B.	Conception détaillée	61
C.	Implémentation.....	67
D.	Test.....	77
E.	Partie physique	80
F.	Bilan technique :.....	81
IV.	Etudiant Testa Tony	82
A.	Description personnelle.....	82
B.	Conception détaillée	83
C.	Implémentation.....	88
D.	Tests	96
E.	Etude Physique	100
V.	Etudiant Arthur LABARRE	102
A.	Description partie personnelle.....	102
B.	Conception détaillée	102
C.	Implémentation.....	106
D.	Test	121
E.	Bilan personnel technique	124
F.	Partie physique	125
VI.	Etudiant Théo SERNOT	127
A.	Description partie personnelle	127
B.	Conception détaillé.....	128
C.	Implémentation.....	131
D.	Tests	147
E.	Partie physique	148
F.	Bilan personnel technique	151
VII.	Étudiant Armand HOUPIN	152
A.	Description partie personnelle	152
B.	Conception détaillée	152
C.	Implémentation.....	159
D.	Tests	176
E.	Partie physique	183
F.	Bilan personnel technique	186

Contexte

Objectifs du projet

Ce projet a pour objectif de permettre à une entreprise de proposer à un client d'effectuer des campagnes de mesures de trafic routiers, ceci consiste à mesurer le nombre de véhicules passant sur une portion routière ainsi que déterminer leur type (Véhicules Légers VL ou Poids Lourds PL), leur vitesse ainsi que la vitesse et enfin la fréquence de passage des véhicules.

Le tout pouvant être supervisé à distance via une page web et le système est testable via un sous-système de simulation afin de vérifier que le système de collecte et de supervision affiche et collecte des données cohérentes.

Matériel imposé pour solutions

Ce projet est une reprise du projet SNET (Système d'analyse de trafic) réalisé en 2020, ce projet était basé sur une architecture Raspberry pi, ce projet a donc été porté sur une architectureMBED avec des cartes Nucléo F746ZG dans un souci d'économie d'énergie et pour les performances de mesure de vitesse des véhicules.

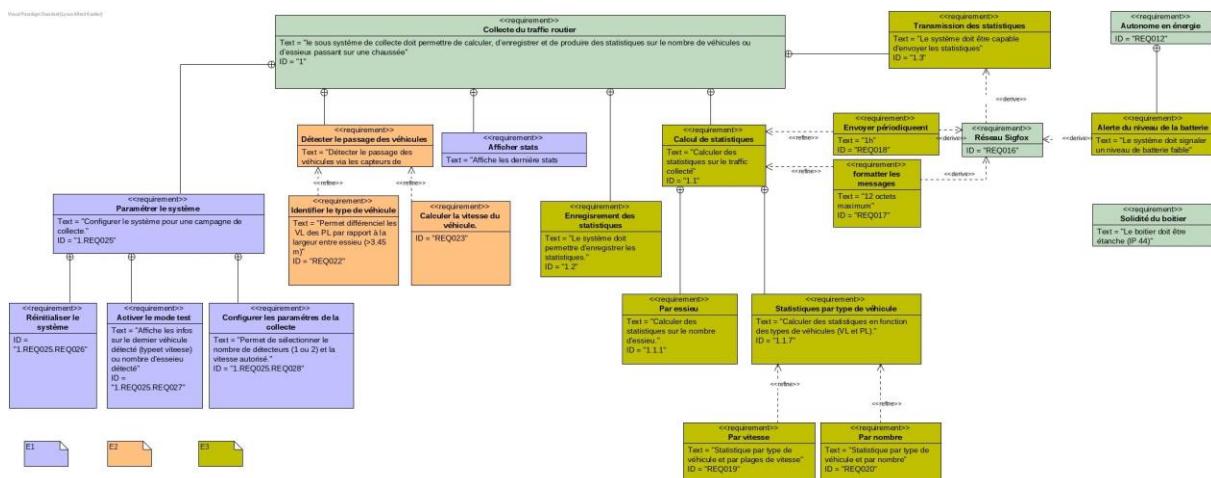
Contraintes techniques

Plusieurs contraintes techniques ont été formulées pour ce projet notamment l'utilisation de platines Nucléo F746ZG pour la majeure partie de l'infrastructure matérielle du système qui est disposée de la manière suivante :

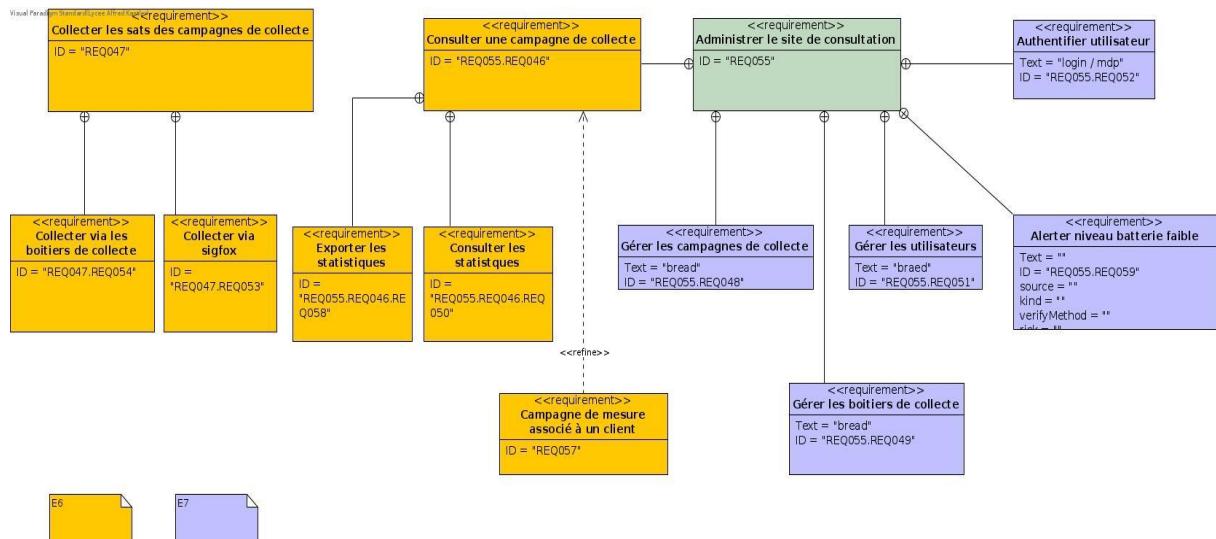
Capteurs	Collecte	Transmission	Visualisation	Gestion	Logiciel	Simulation
2 pressostats	F746ZG	Sigfox	PC/Mobile	Carte PiJuice	PC	F746ZG

Spécifications Générales

Les exigences pour ce système sont représentées pour chaque sous-systèmes du projet, ici, les exigences du sous-système de collecte :



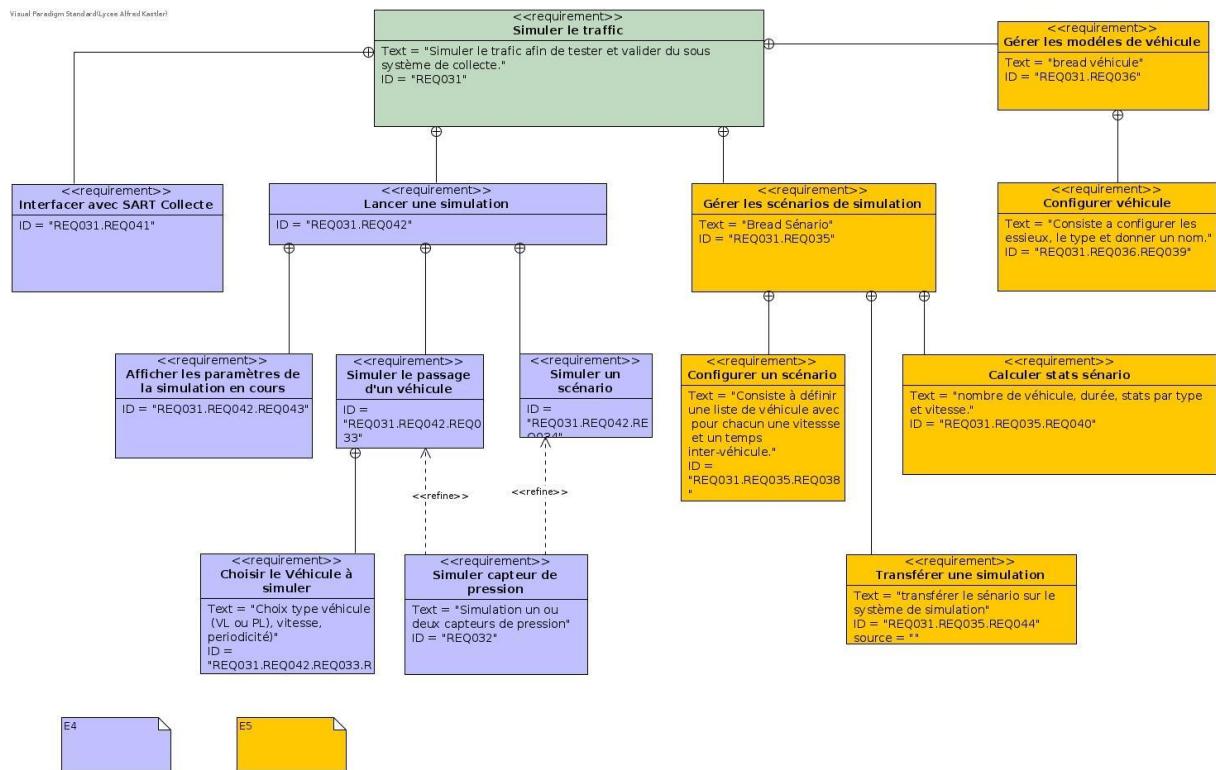
Les exigences du sous-système de consultation :



E6

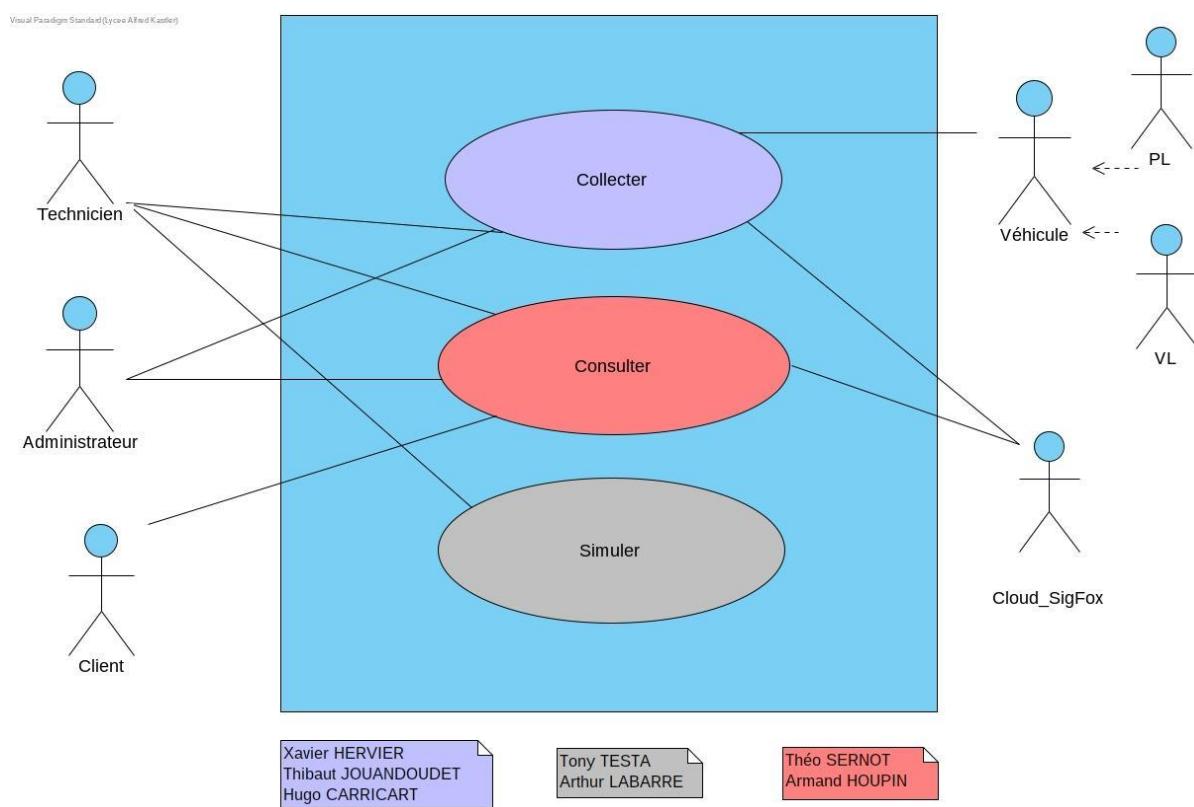
E7

Les exigences du sous-système de Simulation :

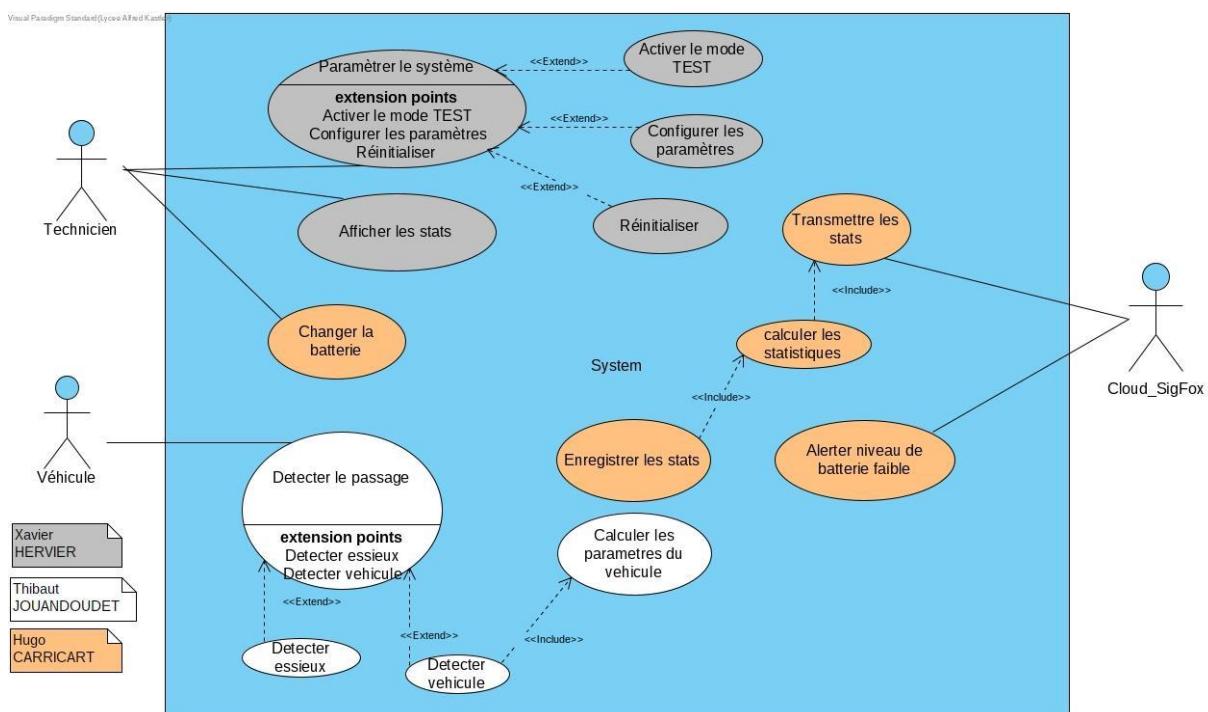


Cas D'utilisations

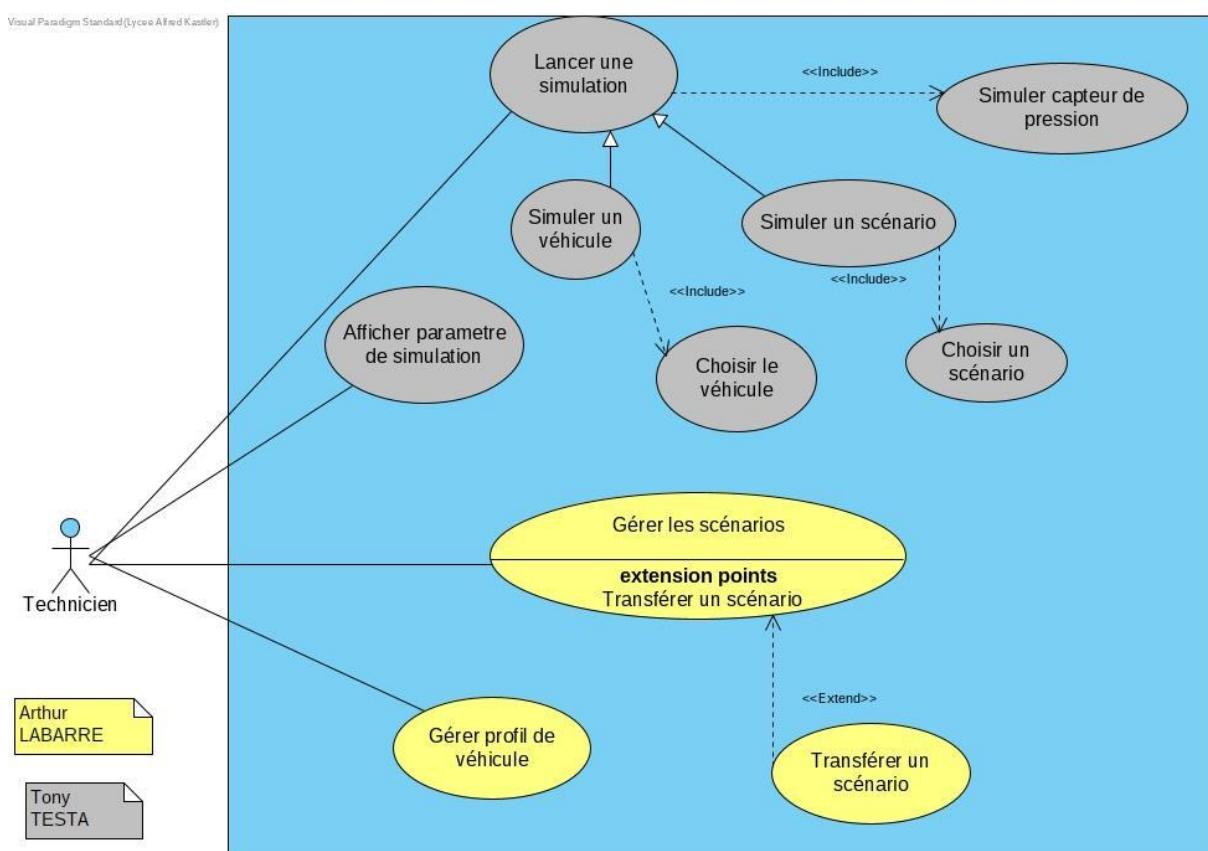
Cas d'utilisation du système :



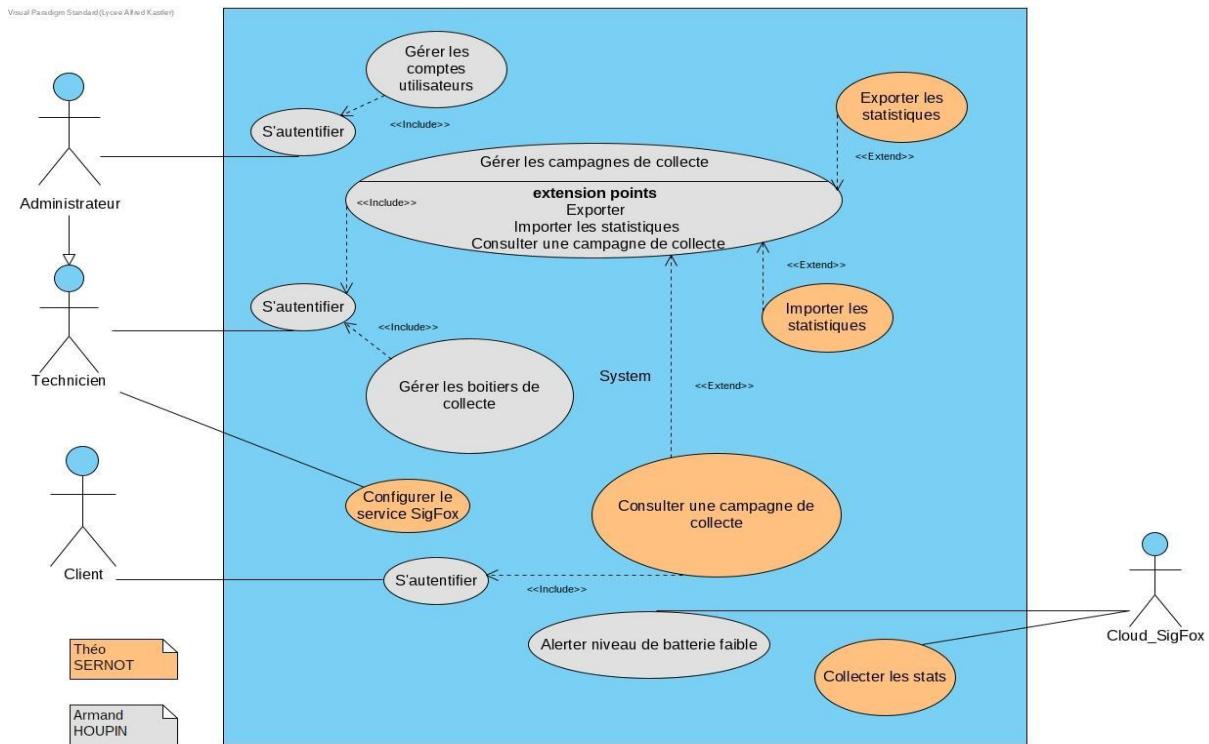
Sous Système de Collecte :



Sous Système de Simulation :



Sous Système de Consultation :

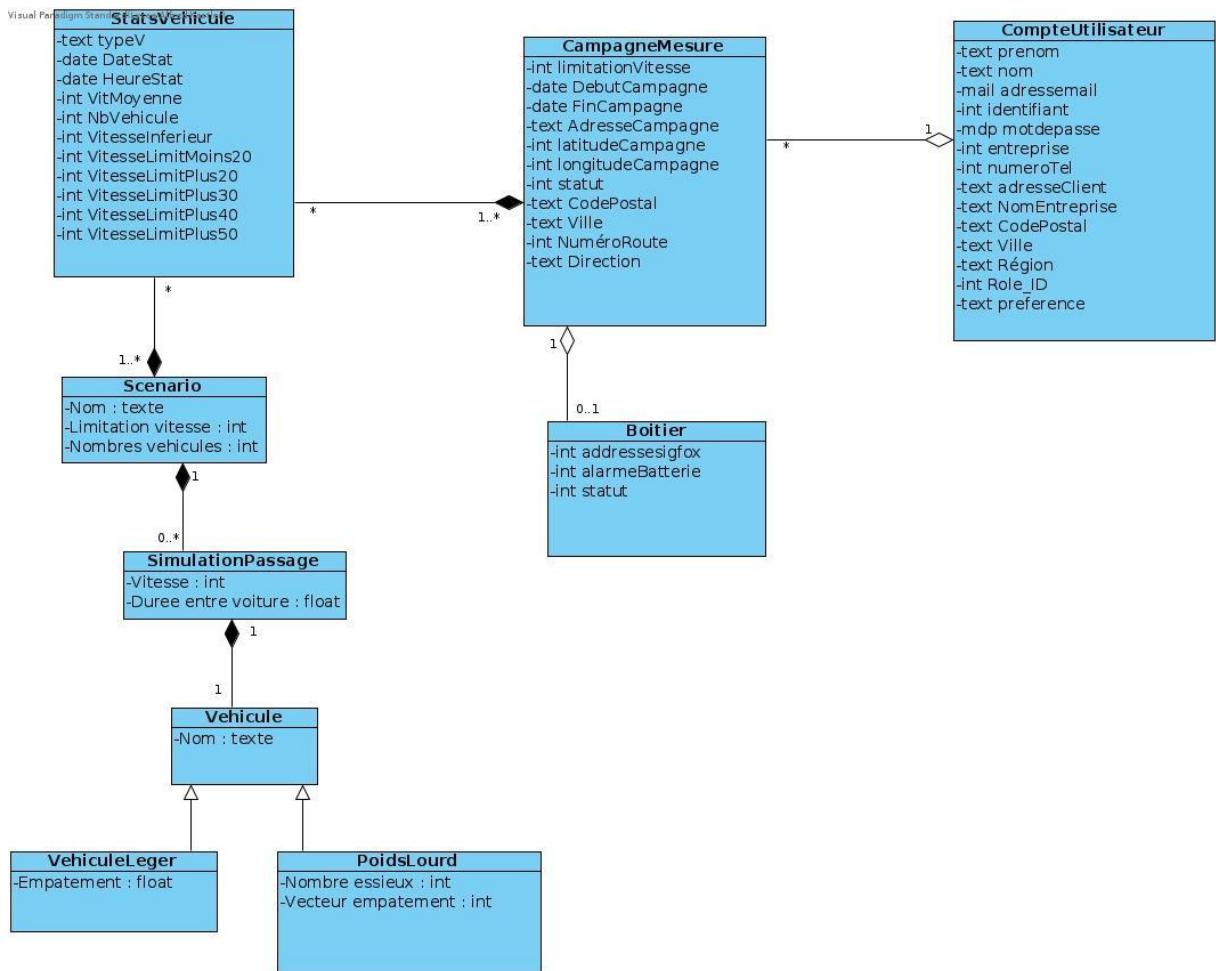


Scénarios

Un client souhaite mesurer les différentes données du trafic sur une portion routière, cette campagne peut donc être mise en place par le projet, le client peut superviser les données depuis le site web (depuis son compte).

Lors de l'installation ou d'une panne/données incohérentes, les techniciens peuvent intervenir sur le système en le testant via un sous-système de simulation.

Modèle de classe du domaine :



Description de l'architecture du système :

Architecture matérielle

L'architecture matérielle, peut être représentée par le diagramme de déploiement suivant :

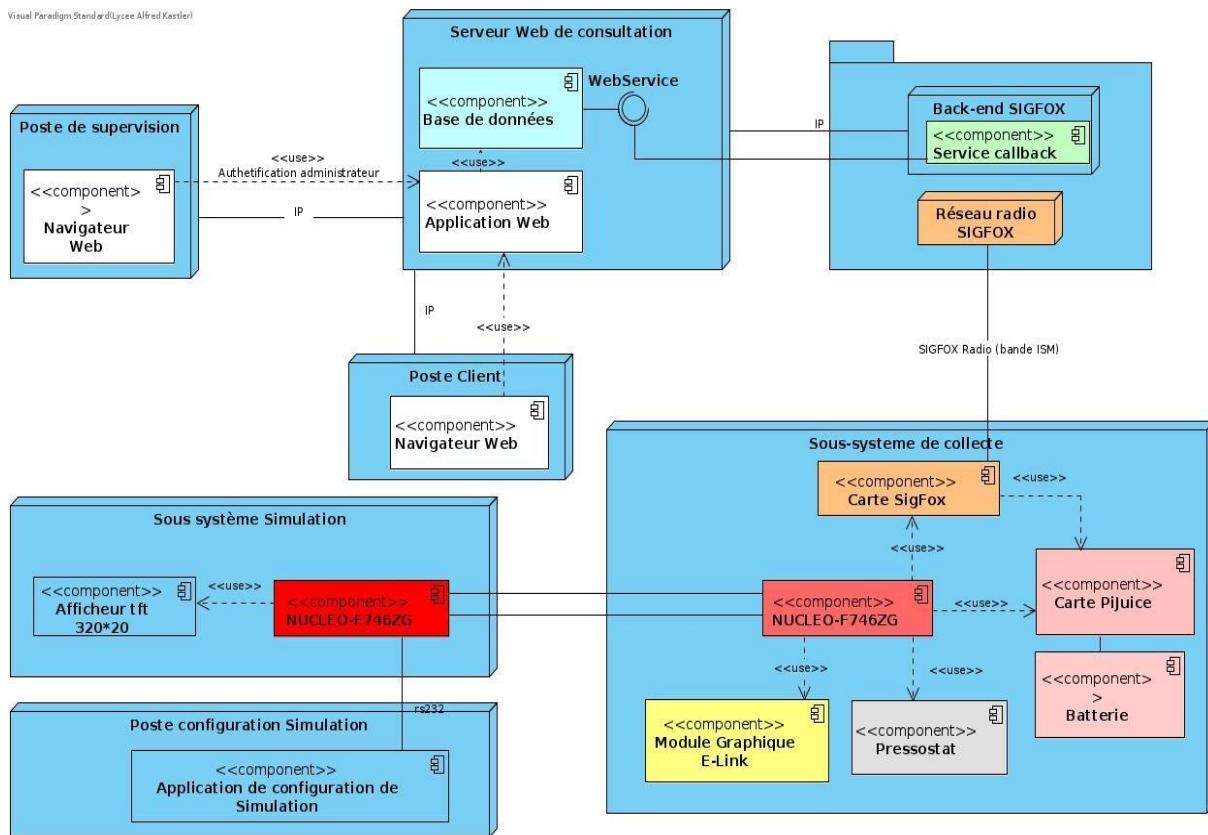
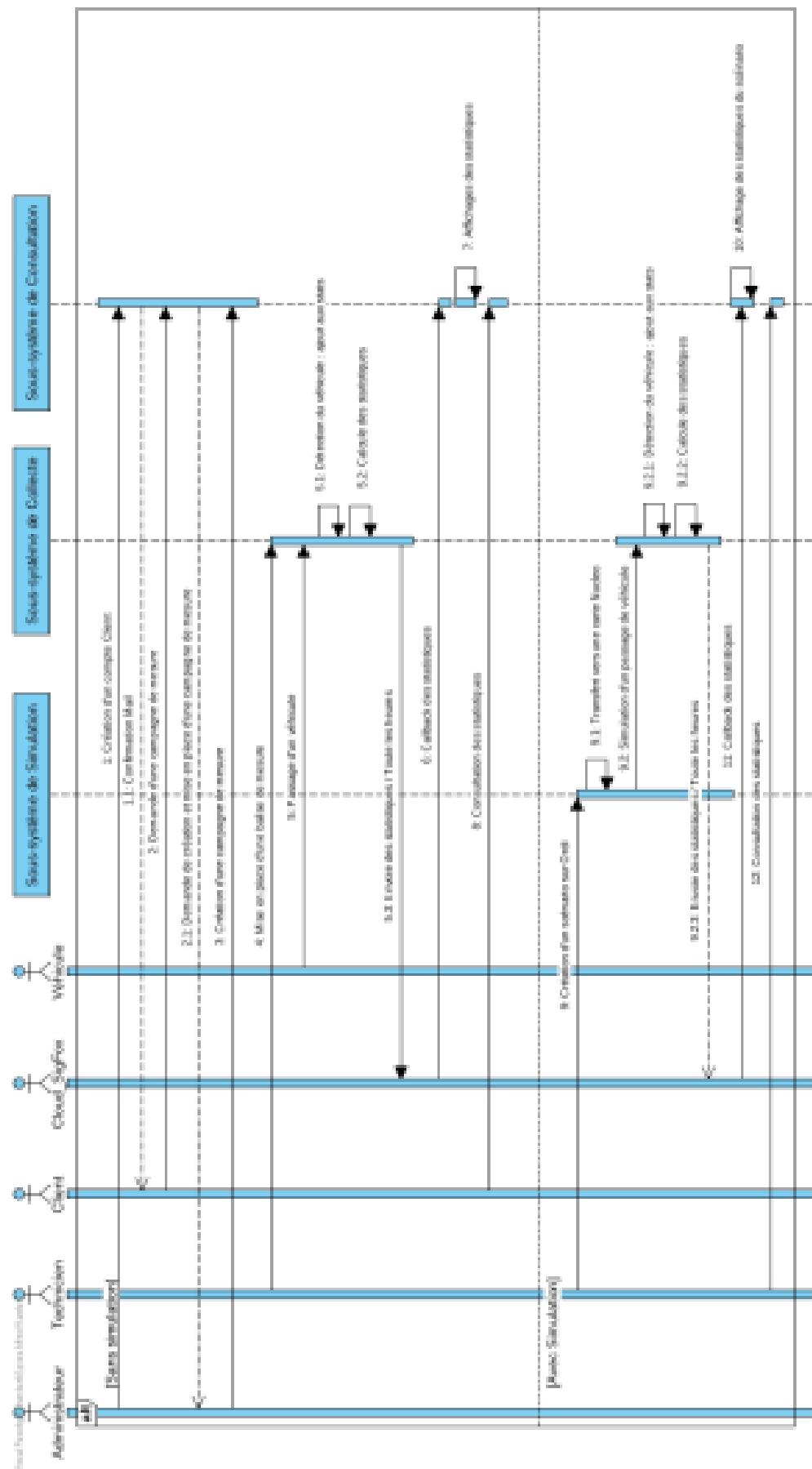


Diagramme de séquence :



Architecture Logicielle :

Désignation :	Caractéristiques techniques :
Sous système de collecte	langage de programmation C++ / mbed-os
Sous système de simulation	langage de programmation C++ / mbed-os Qt / C++
Sous système de Consultation	php / framework laravel Bibliothèque BOOTSTRAP pour interface responsive Bibliothèque graphique HIGHCHARTS Back-End SIGFOX

Recette sous-système simulation (local) :

Elément à tester	Test	Comportement attendu	Validation
Signal créé et cohérent	Acquérir le signal (analog discovery) Puis vérifier les valeurs de temps	Signal créé et Valeurs de temps cohérentes	ok
IHM	Vérifier que les valeurs données par l'IHM sont bonnes et correctement affichées	Valeurs correctes et affichées comme il le faut	ok
Scénario reçu	Vérifier la valeur de la variable trame reçue dans le programme	La valeur de la trame reçue correspond pour chaque caractère à la trame envoyée	ok
Interprétation du JSON	Vérifier que le programme interprète le JSON (récupère un champ et affiche les caractères de ce champ)	Les caractères des champs JSON sélectionnés sont correctement affichés	ok

RECETTE XAVIER COLLECTE

Élément à tester	Test	Comportement attendu	Validation
Menu sélectif	Utiliser les boutons PLUS ou MOINS	Navigation entre les items	Ok
Accès au différent sous menu	Utiliser le bouton valide sur un item	Affichage du nouvelle écran actif	Ok
Afficher Menu	Vérifier si l'écran sélectionner s'affiche	Affichage de l'écran	Ok
Mode Test	Activer le mode test	Afficher des valeurs valides	Nok
Modifier les valeurs	Utiliser le bouton valide sur un item appartenant au Paramètre	Mode Edition activer possibilité d'incrémenter la valeur	Nok
Réinitialisation	Sélectionner item Réinitialiser sur l'écran Paramètre	Valeurs des paramètres de collecte remise par défaut	Nok
Affichage Stats	Sélectionner item Stats sur l'écran d'Accueil	Stats valide par rapport à la collecte	Ok

RECETTE ARMAND CONSULTATION

Élément a testé	Test	Comportement attendu	Validation
Créer un compte	Ne pas remplir le formulaire	Dis à l'utilisateur de remplir le formulaire	OK
Créer un compte	Remplir le formulaire	Créer un compte et renvoie sur la liste des comptes avec le nouveau compte qui apparaît	OK
Créer un compte	Mettre deux fois la même adresse e-mail	Indique à l'utilisateur que le client a déjà été créer	OK
Modifier un compte	Modifier le formulaire prérempli	Modifier les informations du compte et retourne sur la liste des comptes	OK
Supprimer un compte	Cliquer sur la poubelle pour supprimer un compte	Une boîte de dialogue apparaît pour confirmer la suppression	OK
Créer une campagne de mesure	Remplir le formulaire	Créer une campagne de mesure et renvoie sur la liste des campagnes de mesures	OK

Créer une campagne de mesure	Ne pas remplir le formulaire	Dis à l'utilisateur de remplir le formulaire	OK
Créer une campagne de mesure	Associer dans le formulaire la campagne à un client depuis une liste déroulante	Associe le client à la campagne de mesure	OK
Créer une campagne de mesure	Associer dans le formulaire la campagne à un boîtier depuis une liste déroulante	Associe le boîtier à la campagne de mesure	OK
Modifier une campagne de mesure	Modifier le formulaire prérempli	Modifier les informations de la campagne de mesure et retourne sur la liste des campagnes de mesures	OK
Supprimer une campagne de mesure	Cliquer sur la poubelle pour supprimer une campagne de mesure	Une boîte de dialogue apparaît pour confirmer la suppression	OK
Créer un boîtier	Remplir le formulaire	Créer un boîtier et renvoie sur la liste des campagnes de mesures	OK
Créer un boîtier	Ne pas remplir le formulaire	Dis à l'utilisateur de remplir le formulaire	OK
Modifier un boîtier	Modifier le formulaire prérempli	Modifier les informations du boîtier et retourne sur la liste des boîtiers	OK
Supprimer un boîtier	Cliquer sur la poubelle pour supprimer un boîtier	Une boîte de dialogue apparaît pour confirmer la suppression	OK
Visualisation d'une campagne de mesure	Depuis un compte client, voir une campagne de mesure	Voir les informations de la campagne de mesures et les statistiques liée à la campagne de mesures	OK
Visualisation d'une campagne de mesure	Cliquer sur le bouton pour télécharger les statistiques en CSV	Créer un fichier sur l'ordinateur en CSV avec les statistiques	OK
Visualisation d'une campagne de mesure	Aller sur la page de la campagne de mesure	Avoir des graphiques sur les statistiques	OK
Alerter une batterie faible	Avoir une batterie faible	Envoie un e-mail à tous les utilisateurs qui ont un rôle admin et technicien qui spécifie une batterie faible	OK

Répartition du travail par étudiant

PRENOM NOM	TACHE A REALISER	CAS D'UTILISATION	SOUS SYSTEMES
Xavier HERVIER	Configurer Mode test	Activer le mode test	Collecte
Xavier HERVIER	Affichage des statistiques de collecte	Afficher Stats	Collecte
Xavier HERVIER	Configuration de la réinitialisation des paramètre	Réinitialiser	Collecte
Xavier HERVIER	Configuration des paramètres de collecte	Paramétrier le système	Collecte
Tony Testa	Partie locale/matérielle simulation	Gérer véhicule, recevoir scénario et simuler interfaçer avec utilisateur	Simulation
Théo Sernot	développement web service configuration de callback	Configurer le service Sigfox	Consultation
Théo Sernot	consulter les données d'une campagne de collecte issues de la télémesure ou d'une importation des données	Consulter une campagne de collecte	Consultation
Théo Sernot	Exporter les données d'une campagne	Exporter les statistiques	Consultation
Théo Sernot	importer les données issues d'un sous système de collecte ;	Importer les statistiques	Consultation
Théo Sernot	collecter les stats des campagnes de collecte. Collecter via les boîtiers de collecte Collecter via Sigfox	Collecter les statistiques	Consultation
Arthur LABARRE	Configurer les essieux, le type et donner un nom.	Gérer profil de véhicule	Simulation
Arthur LABARRE	Définir une liste de véhicule avec pour	Gérer les scénarios	Simulation

	chacun une vitesse et un temps inter-véhicule.		
Arthur LABARRE	Transférer le scénario sur le système de simulation	Transférer un scénario	Simulation

Armand HOUPIN	Créer des comptes utilisateurs	Gérer compte d'utilisateur	Consultation
Armand HOUPIN	Modifier Comptes Utilisateur	Gérer compte d'utilisateur	Consultation
Armand HOUPIN	Supprimer Comptes Utilisateurs	Gérer compte d'utilisateur	Consultation
Armand HOUPIN	Créer une campagne de mesure	Gérer les campagnes de mesures	Consultation
Armand HOUPIN	Modifier une campagne de mesure	Gérer les campagnes de mesures	Consultation
Armand HOUPIN	Supprimer Une campagne de mesure	Gérer les campagnes de mesures	Consultation
Armand HOUPIN	Créer un boitier	Gérer les boitiers	Consultation
Armand HOUPIN	Modifier un boitier	Gérer les boitiers	Consultation
Armand HOUPIN	Supprimer un boitier	Gérer les boitiers	Consultation
Armand HOUPIN	Consultation à une campagne de mesure	Consulter une campagne de mesures	Consultation
Armand HOUPIN	Alerter quand une batterie est faible	Alerte niveau de batterie faible	Consultation

DÉPLOIEMENT ou RECETTES / VALIDATION

1. Site de consultation (Armand)

Site déployé sur OVH : <http://kolantr.sn-kastler.fr:8000>

Compte admin par défaut : identifiant : adminKolantr2021

mot de passe : password

Compte client par défaut : identifiant : c21fpoknptot8416

mot de passe : 21y9yvjq6zr21gewnbs1qv

CONCLUSION Partie Commune

Bilan technique Site de consultation (Armand)

- ✓ Gérer comptes utilisateurs
- ✓ Gérer campagnes de mesures
- ✓ Gérer boîtiers
- ✓ Consulter une campagne de mesure depuis un compte client
- ✓ Gérer l'alarme batterie

Bilan technique IHM sur E-paper (Xavier)

- ✓ IHM sur epaper
- ✓ Affichage des Stats
- ✓ Paramétriser la collecte

Bilan technique (Hugo)

- ✓ Calcule des statistiques
- ✓ Fabrication des trames
- ✓ Envoye des trames

Bilan technique (Arthur)

- ✓ Générer les modèles de véhicule
- ✓ Configurer véhicule
- ✓ Générer les scénarios de simulations

✓ Configurer un scénario

✓ Calculer stats scénario

✓ Transférer un scénario

DOSSIER TECHNIQUE

PARTIE PERSONNELLE

2^{ème} année BTS système et numérique

Option Informatique et Réseau

Projet Kolantr



I. Etudiant Xavier HERVIER

A. Description partie personnelle

Ce projet se définit en trois temps :

- ✗ une partie simulation,
- ✗ une partie recueil de données sur le terrain,
- ✗ une partie visualisation (analyse).

Sous système de Collecte : c'est un collecteur de trafic de campagne de mesure. Il permet de calculer, enregistrer, envoyer et calculer des statistiques

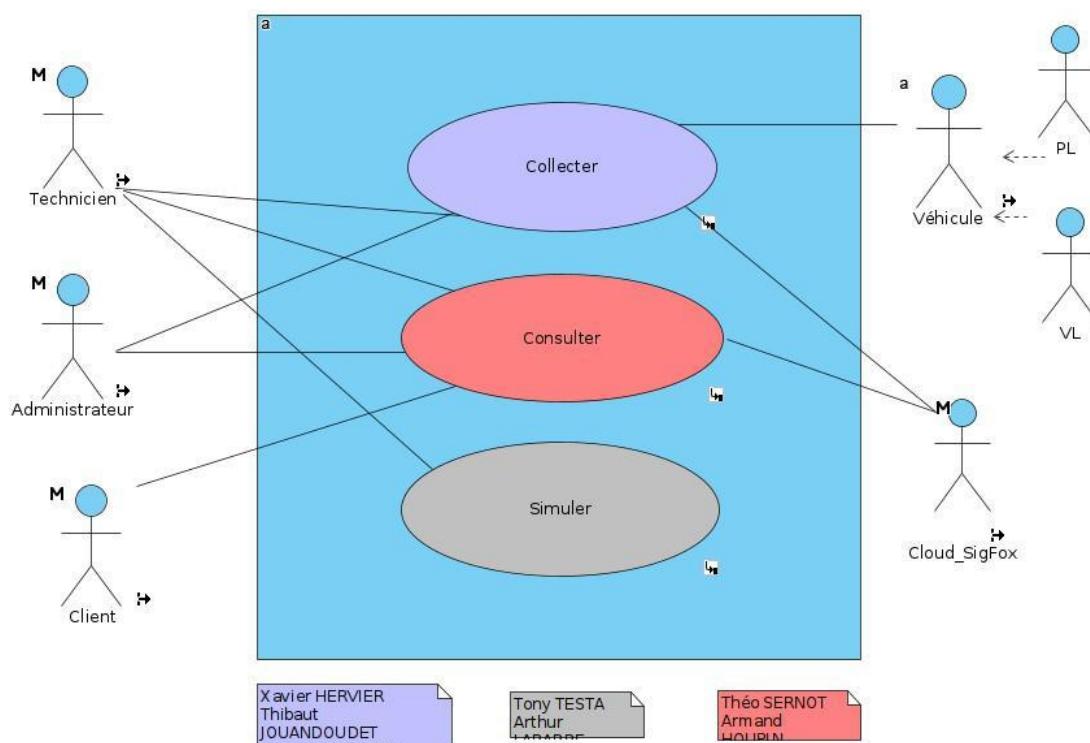
Les statistiques sont envoyées périodiquement par le réseau Sigfox.

Une IHM sur e-paper permet de configurer la collecte et observer les statistiques collectées, tout cela en utilisant une liaison SPI.

Une liaison **SPI** (pour **Serial Peripheral Interface**) est un bus de données série synchrone baptisé ainsi par Motorola, au milieu des années 1980 et qui opère en mode full-duplex.

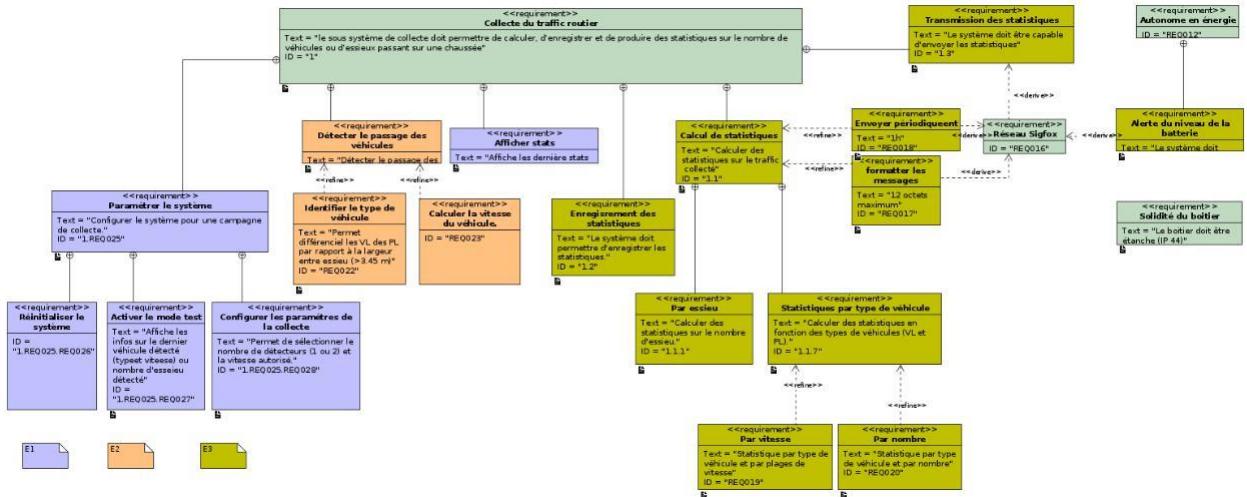
Les circuits communiquent selon un schéma maître-esclave, où le maître contrôle la communication.

Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée « Slave Select (SS) ».

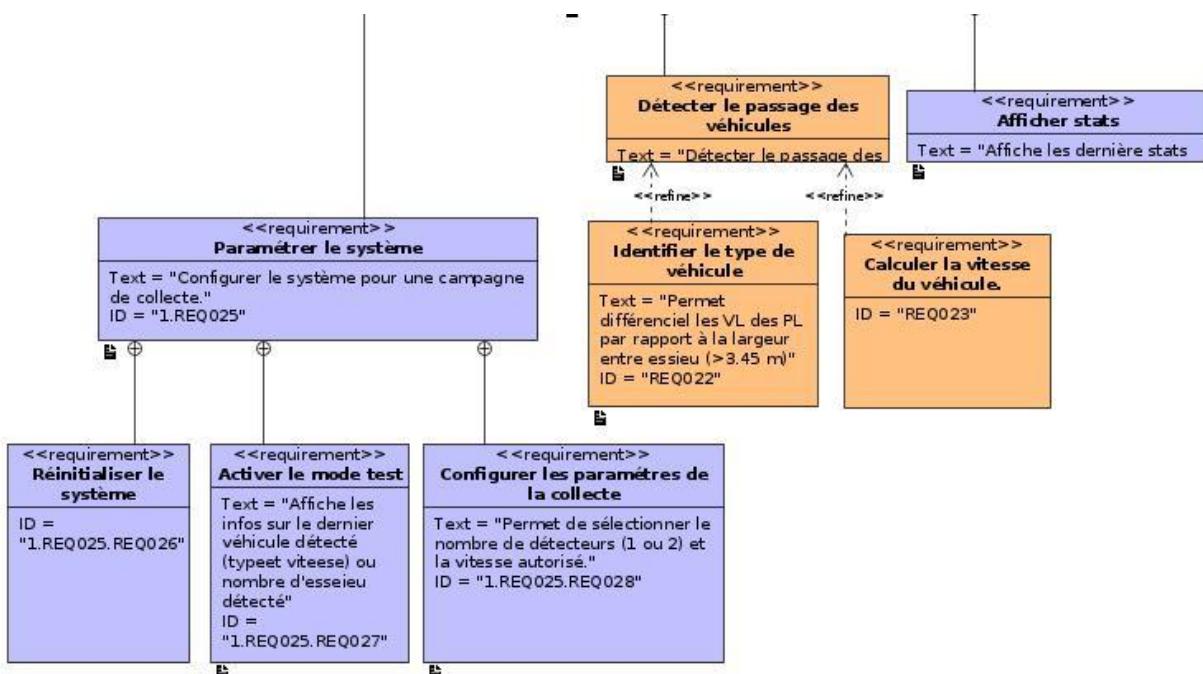


Ma partie est celle en Violet : « Collecter »

B. Conception détaillée



Nous nous concentrerons ici sur la partie embarquée soit sur les exigences suivantes (violet) :



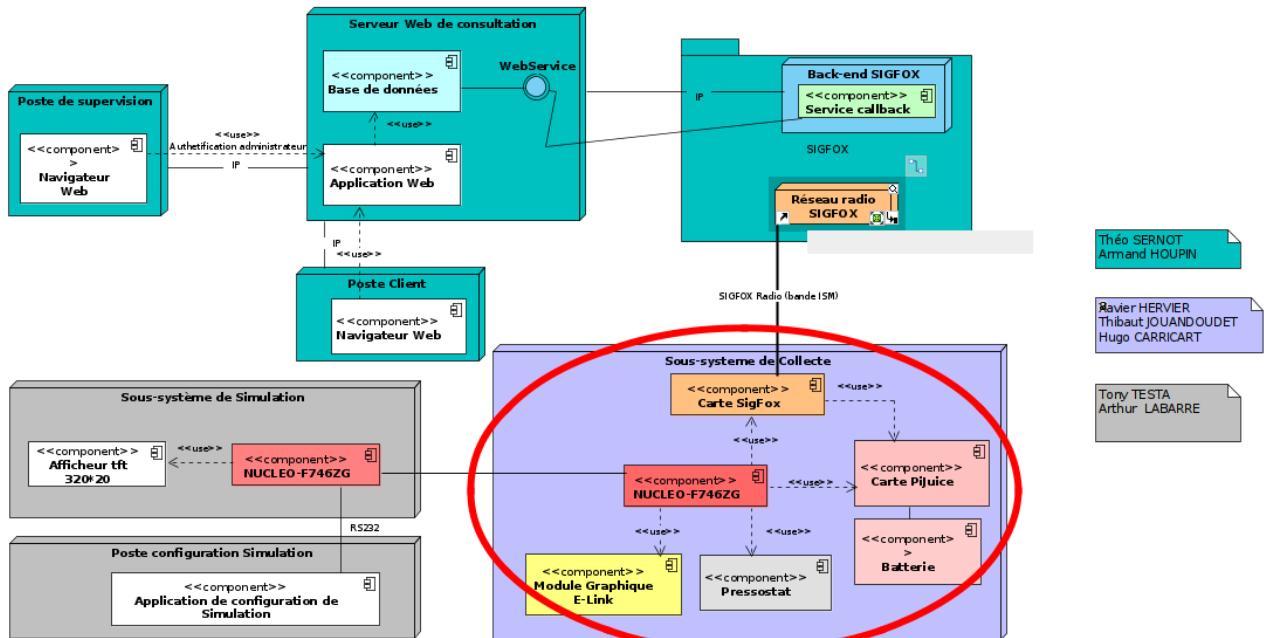


Diagramme de déploiement Collecte

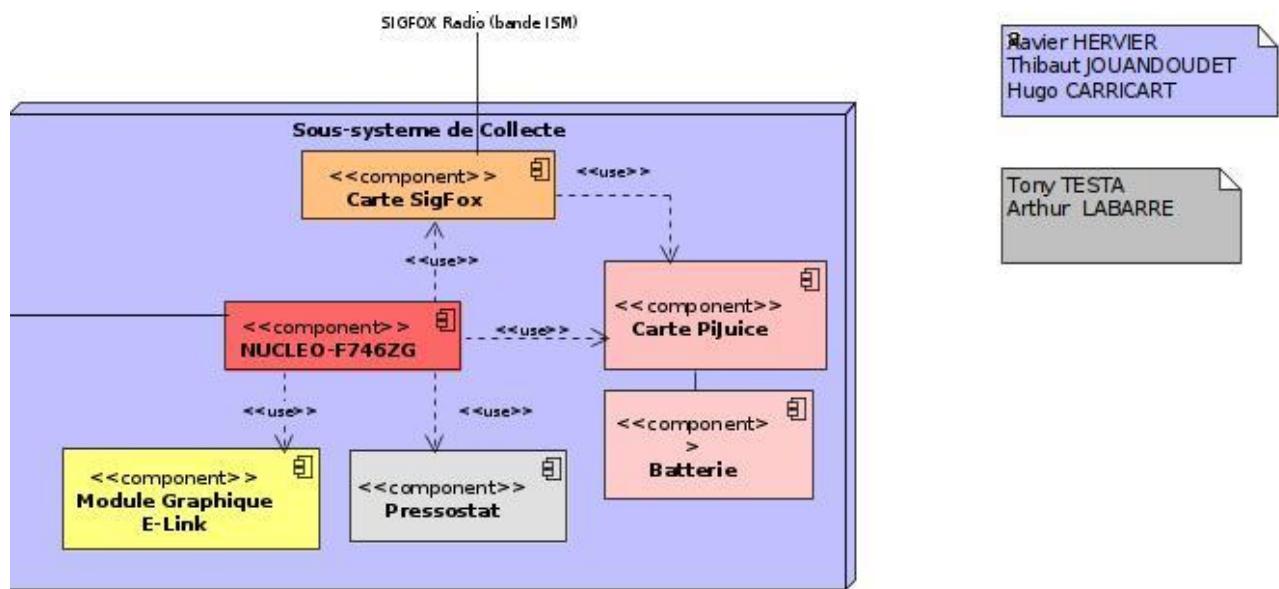
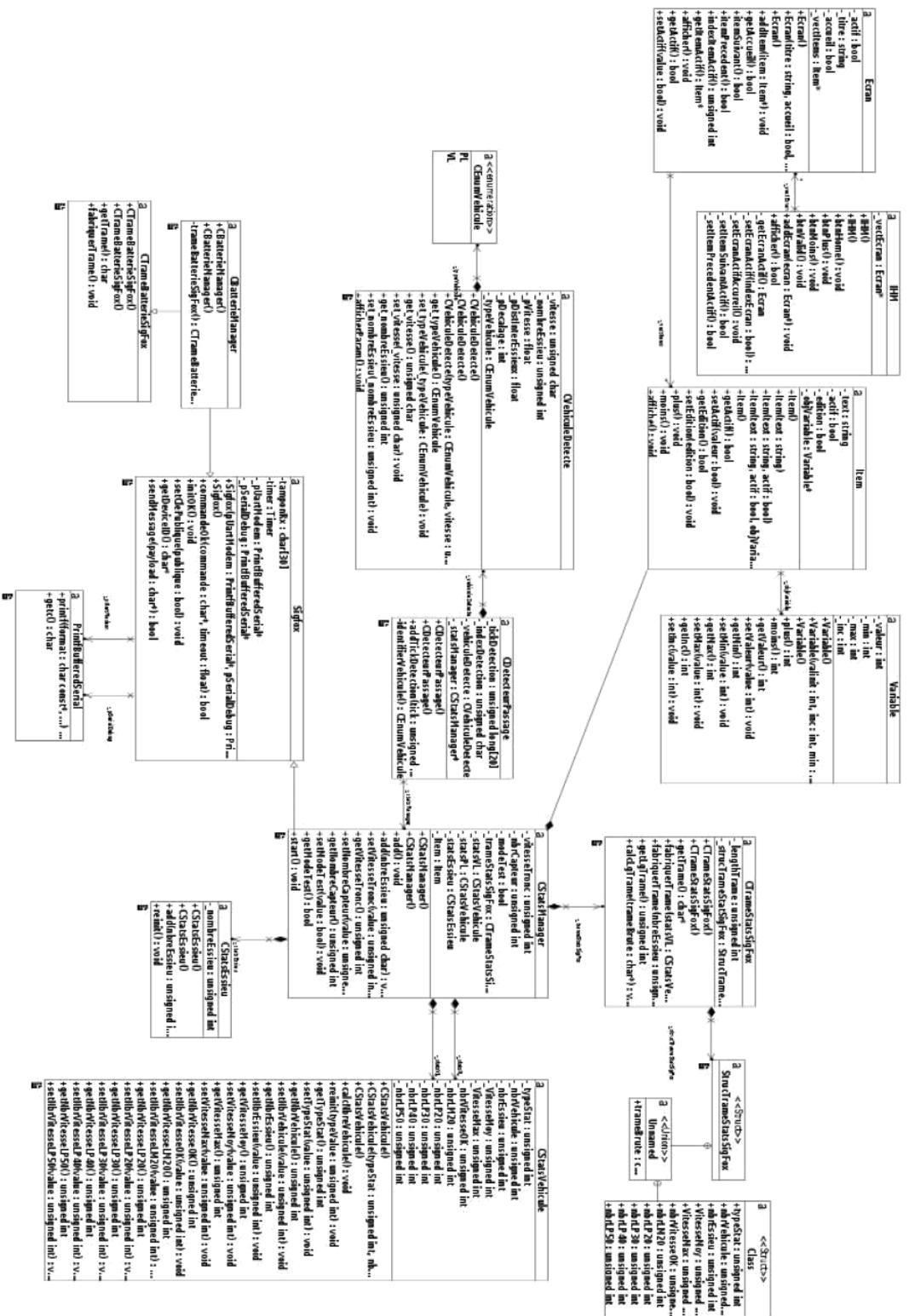


Diagramme de déploiement Collecte zoomé sur la partie personnelle



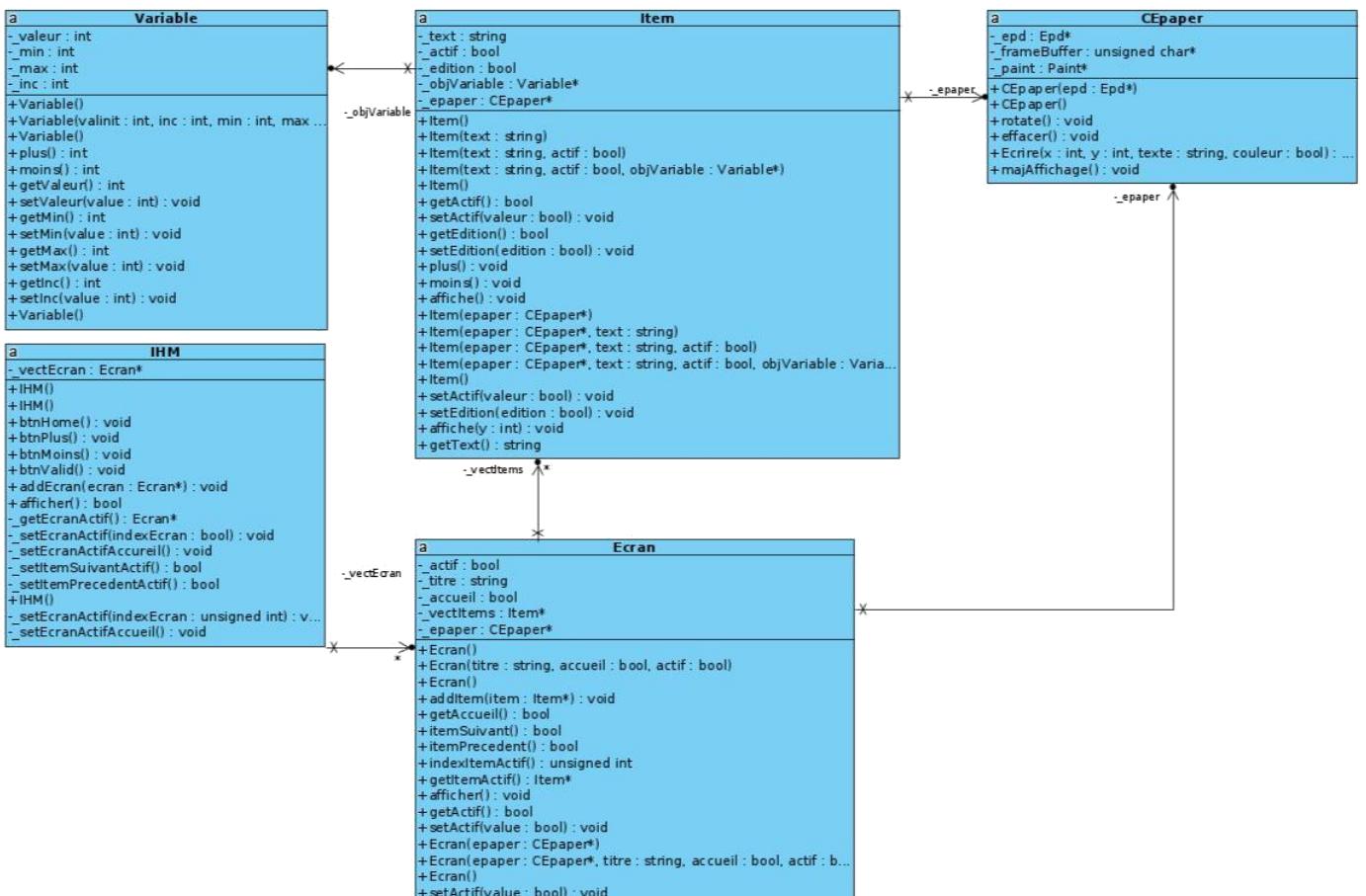
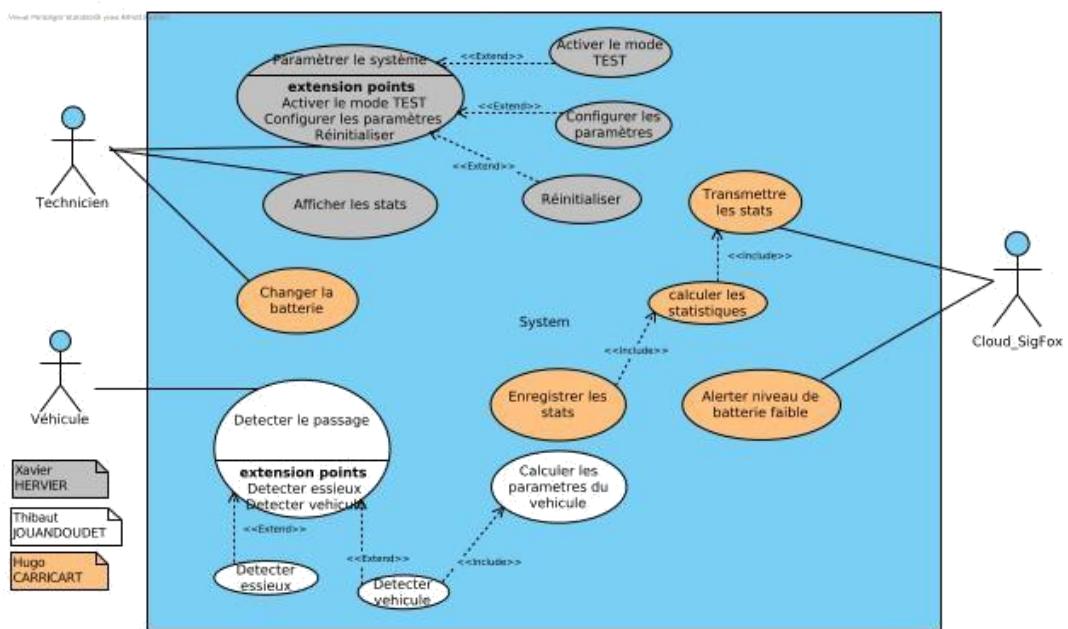


Diagramme de classe de l'étudiant

Classe	Fonction
IHM	Cette classe permet l'utilisation des boutons, contient les différents écrans puis peut les afficher.
Ecran	Cette classe gère les différents écrans de l'IHM, elle contient des Items et utilise un objet de CEpaper. Permet d'afficher les différents Ecrans.
Item	Cette classe contient un objet de la classe Variable et de la classe CEpaper. Permet d'afficher les différents Items.
Variable	Permet d'incrémenter les différentes valeurs et de leur attribuer une valeur maximale et minimale. Il contient les différentes variables : - Nombre de Capteurs - Vitesse Autorisée - Vitesse Moyenne - Vitesse Maximale
CEpaper	Cette classe gère l'affichage sur l'écran en mettant à jour l'écran e-paper

1. Diagramme de cas d'utilisation_Collecte



● 2. Activer le mode TEST

ID: UC30

Affiche les derniers véhicules détectés (type et vitesse) ou nombre d'essieu détecté

2.1. Extended from

● Paramétrier le système

2.2. Scenario (Test du mode test)

Steps	Procedures	Expected Results
1. Mode test	Lancer le mode test	Afficher des valeurs valides par rapport au mode test

● 3. Afficher les stats

ID: UC37

Affiche les dernières stats de la collecte.

3.1. Primary Actors

✗ Technicien

3.2. Scenario (Test Menu afficher stats)

Steps	Procedures	Expected Results
1. Menu sélectif	Utiliser boutons Up et Down	Se déplacer dans le menu de haut en bas
2. Entrer dans menu Afficher stats	Utiliser bouton ok sur afficher stats	Voir le menu Afficher stats
3. Affichage de stats	Vérifier stats	Stats valide

● 4. Configurer les paramètres

ID: UC29

Permet de sélectionner le nombre de détecteurs (1 ou 2) et la vitesse autorisé

4.1. Extended from

● Paramètrer le système

4.2. Scenario (Test configurer les paramètres)

Steps	Procedures	Expected Results
1. Modifier les valeurs	Sélection la variable à modifier et cliquer pour modifier	Possibilité de modifier
2. Vérifier les valeurs si valeurs valide	Activant le mode test	Valeur valide en afficher stats

● 5. Paramètrer le système

ID: UC28

Menu Paramétré le système avec les différents sous menu : Activer le mode TEST, Configurer les paramètres, Réinitialiser.

5.1. Primary Actors

♀ Administrateur, ♀ Technicien

5.2. Extend

● Activer le mode TEST, ● Configurer les paramètres, ● Réinitialiser

5.3. Scenario (Test du menu paramètre)

Steps	Procedures	Expected Results
1. Menu sélectif	Utiliser boutons Up et Down	Se déplacer dans le menu de haut en bas
2. Afficher menu paramètre	Cliquer bouton ok sur paramètre	Affichage menu paramètre
3. Accès au différent sous menu	Utiliser bouton sélection sur les sous menus	Accès au différents sous menu

● 6. Réinitialiser

ID: UC31

Réinitialiser les valeurs des paramètres (vitesse autorisé, nombre de capteurs)

6.1. Extended from

- Paramètrer le système

6.2. Scenario (Test renitialiser)

Steps	Procedures	Expected Results
1. Test de renitialiser	Sélection renitialiser puis bouton ok	Valeur de configurer repasser par defaut

‡ 7. Cloud_SigFox

ID: AC18

7.1. Use Cases

- Consulter, ● Transmettre les stats, ● Alerter niveau de batterie faible, ● Collecter les stats, ● Alerter niveau de batterie faible, ● Collecter

‡ 8. Technicien

ID: AC23

8.1. Use Cases

- Collecter, ● Consulter, ● Simuler, ● Paramètrer le système, ● Afficher les stats, ● Changer la batterie, ● Afficher parametre de simulation, ● Lancer une simulation, ● Gérer les scénarios, ● Gérer profil de véhicule, ● Configurer le service SigFox, ● S'autentifier

8.2. Sub Use Cases

‡ Administrateur

‡ 9. Véhicule

ID: AC14

9.1. Use Cases

- Detecter le passage, ● Collecter

C. Implémentation

- **Mise en œuvre des techniques / technologies utilisées :**

Mbed : Procédure et mise en place de la chaîne de cross-compilation :

Outils utilisés :

- Visual Studio Code : <https://code.visualstudio.com/sha/download?build=stable&os=win32-x64-user>

- Server GDB SEGGER :

https://www.segger.com/downloads/jlink/JLink_Windows_V700a.exe

- Arm-gcc-none-eabi : <https://developer.arm.com/-/media/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-win32.zip?revision=ffcaa06c-940d-4319-8d7e-d3070092d392&la=en&hash=130196DDF95B0D9817E1FDE08A725C7EBFBFB5B8>

- Mbed – cli : <https://github.com/ARMmbed/mbed-cli-windows-installer/releases/latest>

Ordre de la procédure (sous Windows) :

1) Installer Visual Studio Code et l'extension Cortex-Debug

2) Installer le SEGGER

3) Décompresser l'archive d'ARM à l'emplacement de son choix

4) Installer mbed-cli

(ATTENTION : Redémarrer l'ordinateur et tester dans l'invite de commande la commande : mbed)

5) SI commande applet impossible alors voir version de Python et tester un upgrade de pip (avec la commande : python -m pip install --upgrade pip // Redémarrer à nouveau)

6) Cloner un template fonctionnelle de mbed :

http://gogs.lyceekastler.fr/hug.Carricart/Mbed_Template/

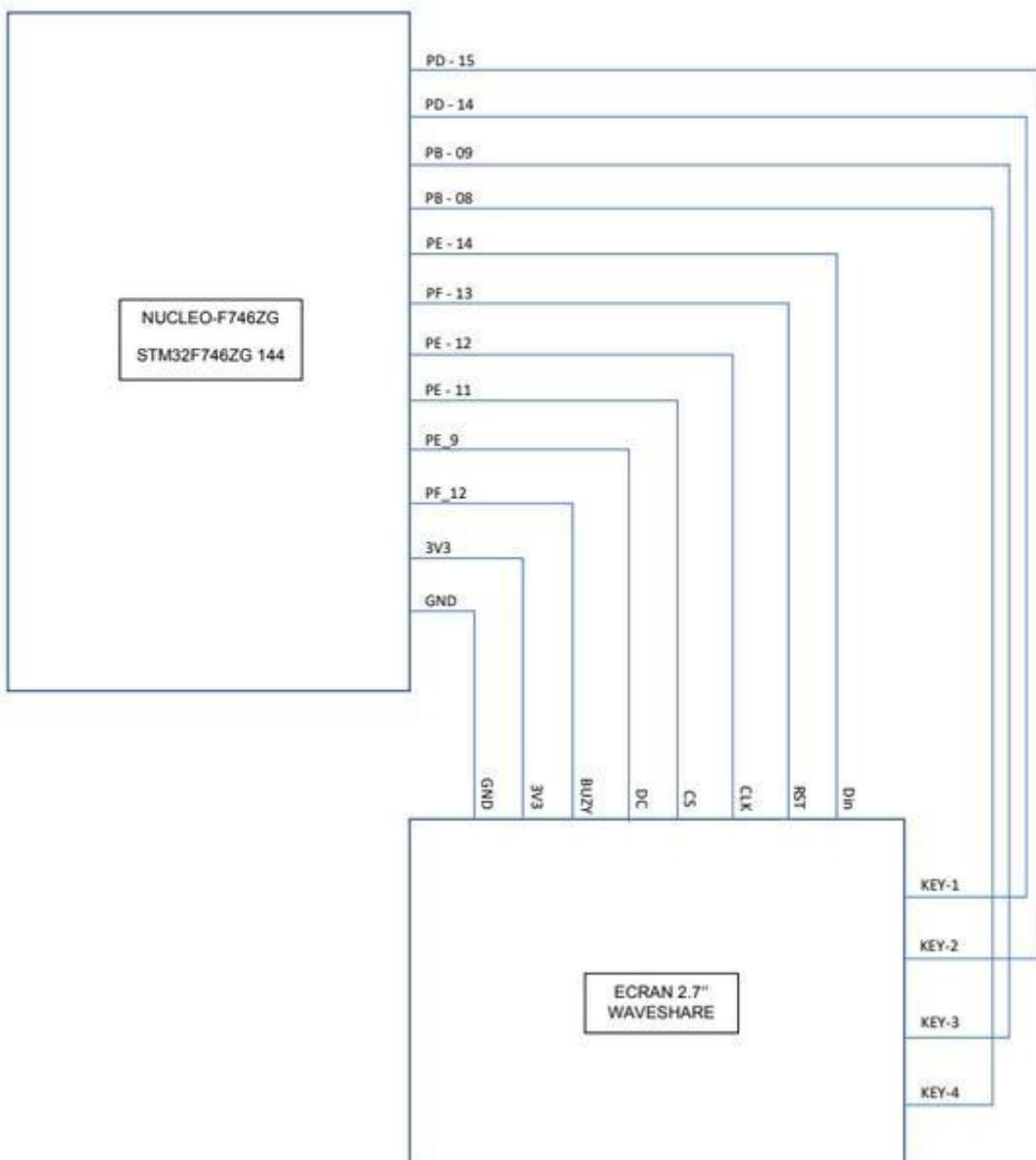
7) Réadapter le fichier de configuration : launch.json et c_cpp_properties.json

8) Coder !

E-paper : Mise en œuvre de l'e-paper

Outils utilisées : Librairie epd2in7, ecran e-paper.

SCHEMA DE CABLAGE ENTRE LE MODULE STM32 ET L'ECRAN 2.7" WAVESHARE



Pour les boutons

Bouton	Zio
Key1 (b29 - GPIO5)	PD_14
Key2 (b31 - GPIO6)	PD_15
Key3 (b33 - GPIO13)	PB_9
Key4 (b35 - GPIO19)	PB_8

Pour l'affichage : communication SPI plus lignes de contrôle

Ligne	zio
Busy	PF_12
RST	PF_13
DC	PE_9
CS	SPI4_CS - PE_11
CLK	SPI4_CLK - PE_12
Din(spi mosi)	SPI4_MOSI - PE_14
GND	GND
3,3 V	3v3

Affectation des broches pour la communication spi et pour les boutons :

```
//affectation des broches pour la communication spi
PinName _Mosi = PE_14 ; //SPI4_MOSI;
PinName _Miso = PE_13 ; // SPI4_MISO non connecté
PinName _Sclk = PE_12 ; // SPI4_SCK
PinName _Cs   = PE_11 ; // SPI4_SC
//affectation des broches de controle de l'afficheur
PinName _Rst  = PF_13 ;
PinName _Dc   = PE_9 ;
PinName _Busy = PF_12 ;

#define BT_HOME    PD_15
#define BT_PLUS    PD_14
#define BT_MOINS   PB_9
#define BT_SELECT  PB_8
```

Création d'une IHM :

Utilisation de la Librairie epd2in7 :

```
Epd::Epd(PinName mosi,
          PinName miso,
          PinName sclk,
          PinName cs,
          PinName dc,
          PinName rst,
          PinName busy
        ):EpdIf(mosi, miso, sclk, cs, dc, rst, busy){

    width = EPD_WIDTH;
    height= EPD_HEIGHT;
    rotate = ROTATE_0;

}
```

Création d'un Objet Epaper avec un Objet Epd (contenant les broches pour l'écran) :
CEpaper.hpp

```
CEpaper(Epd *epd);
```

main.c

```
Epd * gEpd = new Epd(_Mosi, _Miso, _Sclk, _Cs, _Dc, _Rst, _Busy);
CEpaper *Epaper = new CEpaper(gEpd);
```

Création d'un objet IHM avec l'initialisation de l'Epaper :

main.c

```
IHMpt = new IHM() ;
//initialisation epaper
if (gEpd->Init() != 0)
{
    myledR = 1;
    return -1;
}
else
    myledV = 1;
```

Création des différents objets Ecran :

main.c

```
Ecran * EcranAc = new Ecran(Epaper,"Accueil",true,true);
Ecran * EcranPara = new Ecran(Epaper,"Parametre",false,false);
Ecran * EcranSta = new Ecran(Epaper,"Stats",false,false);
Ecran * EcranTes = new Ecran(Epaper,"Test",false,false);
```

Ajout des Ecrans dans l'IHM, ajout de Item dans chaque Ecran et ajout d'une Variable dans chaque Items

Ecran.hpp

```
/**  
 * @brief Construit un objet Ecran  
 */  
Ecran(CEpaper *epaper);  
/**  
 * @brief Construit un objet Ecran  
 * @param epaper  
 * @param titre  
 * @param accueil  
 * @param actif  
 */  
Ecran(CEpaper *epaper, string titre, bool accueil, bool actif);
```

Item.hpp

```
/**  
 * @brief Construit un objet Item  
 * @param epaper  
 */  
Item(CEpaper *epaper);  
/**  
 * @brief Construit un objet Item  
 * @param epaper  
 * @param text      Titre objet Item  
 */  
Item(CEpaper *epaper, string text);  
/**  
 * @brief Construit un objet Item  
 * @param epaper  
 * @param text      Titre objet Item  
 * @param actif     Si Item actif ou non actif  
 */  
Item(CEpaper *epaper, string text, bool actif);  
/**  
 * @brief Construit un objet Item  
 * @param epaper  
 * @param text      Titre objet Item  
 * @param actif     Si Item actif ou non actif  
 * @param objVariable  Variable de l'Item  
 */  
Item(CEpaper *epaper, string text, bool actif, Variable *objVariable);
```

Variable.hpp

```
/**  
 * @brief Constructeur objet Variable  
 */  
Variable();  
/**  
 * @brief Construit un objet Variable  
 * @param valinit      Valeur initiale  
 * @param inc          Valeur de l'incrémentation  
 * @param min          Valeur Minimale  
 * @param max          Valeur Maximale  
 */  
Variable(int valinit , int inc, int min, int max);
```

main.c

```
IHMpt->addEcran(EcranAc);  
EcranAc->addItem(new Item (Epaper,"Parametre",true));  
EcranAc->addItem(new Item (Epaper,"Stats",false));  
EcranAc->addItem(new Item (Epaper,"Test",false));  
  
IHMpt->addEcran(EcranPara);  
EcranPara->addItem new Item (Epaper,"Capteur",true, new Variable(2,1,0,2));  
EcranPara->addItem new Item (Epaper,"Vitesse",false, new Variable(20,10,20,130));  
  
IHMpt->addEcran(EcranSta);  
EcranSta->addItem(new Item (Epaper,"Stats",true));  
  
IHMpt->addEcran(EcranTes);  
EcranTes->addItem(new Item (Epaper,"Mode Test",true));
```

Fonctionnement des Boutons :

IHM.cpp

```
void IHM::btnHome()
{
    _setEcranActifAccueil();
    afficher();
};

void IHM::btnPlus()
{
    Item *itemActif = _getEcranActif()->getItemActif();
    if(itemActif != nullptr){
        _getEcranActif()->itemSuivant();
        afficher();
    }
};

void IHM::btnMoins()
{
    Item *itemActif = _getEcranActif()->getItemActif();
    if(itemActif != nullptr){
        _getEcranActif()->itemPrecedent();
        afficher();
    }
};

void IHM::btnValid()
{
    if(_getEcranActif()->getAccueil() == true){
        unsigned int ItemActif = _getEcranActif()->indexItemActif();
        _setEcranActif(ItemActif);
        afficher();
    }
};
```

Fonctionnement des méthodes pour passer d'un Item à un autre que ce soit pour Item suivant ou Item précédent :

Ecran.cpp

```

bool Ecran::itemSuivant()
{
    for (int i = 0; i < _vectItems.size(); i++)
    {
        if (_vectItems[i]->getActif() == true) {
            _vectItems[i]->setActif(false);
            if(i+1 < _vectItems.size()){
                _vectItems[i+1]->setActif(true);
            }else{
                _vectItems[0]->setActif(true);
            }
            return true;
        }
    }
    return false;
};

bool Ecran::itemPrecedent()
{
    for (int i = 0; i < _vectItems.size(); i++)
    {
        if (_vectItems[i]->getActif() == true) {
            _vectItems[i]->setActif(false);
            if(i-1 >= 0){
                _vectItems[i-1]->setActif(true);
            }else{
                _vectItems[_vectItems.size()-1]->setActif(true);
            }
            return true;
        }
    }
    return false;
};

```

Liste tous les Items pour regarder lequel est actif

Rend l'Item actif en non actif
Récupère l'Item suivant sur le vecteur puis le rend actif à son tour

Bouton	Fonction
btnHome()	Permet le retour à l'accueil
btnPlus()	Récupère l'Item actif et sélectionne l'Item suivant pour le mettre actif à son tour et met l'ancien Item en non actif
btnMoins()	Récupère l'Item actif et sélectionne l'Item précédent pour le mettre actif à son tour et met l'ancien Item en non actif
btnValid()	Permet l'accès à un Item ou de rentrer dans le mode édition sur un Item contenant une Variable et en ayant l'édition active

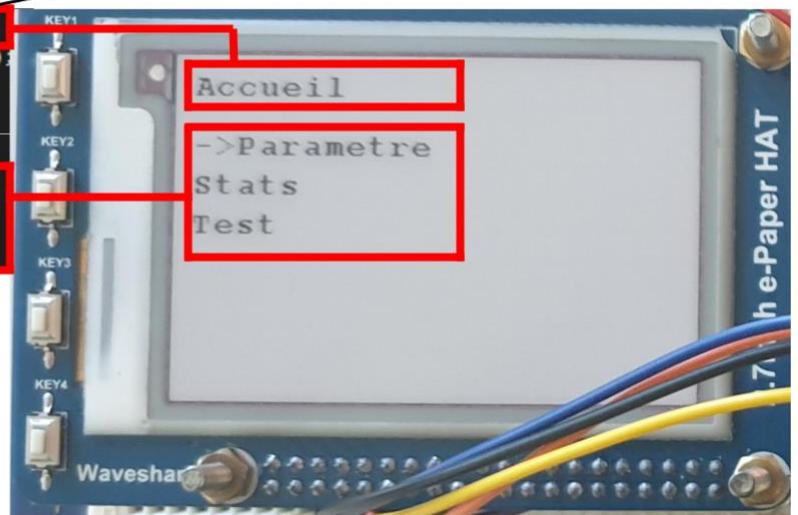
D. Test

Test unitaire : Affichage sur écran

Création d'un objet type écran avec comme Titre Accueil étant actif comme accueil et actif sur l'écran

```
Ecran * EcranAc = new Ecran(Epaper,"Accueil",true,true);
Ecran * EcranPara = new Ecran(Epaper,"Parametre",false,false);
Ecran * EcranSta = new Ecran(Epaper,"Stats",false,false);
Ecran * EcranTes = new Ecran(Epaper,"Test",false,false);

IHMpt->addEcran(EcranAc);
EcranAc->addItem(new Item (Epaper,"Parametre",true));
EcranAc->addItem(new Item (Epaper,"Stats",false));
EcranAc->addItem(new Item (Epaper,"Test",false));
```



Création des Objets Item dans l'écran Accueil.

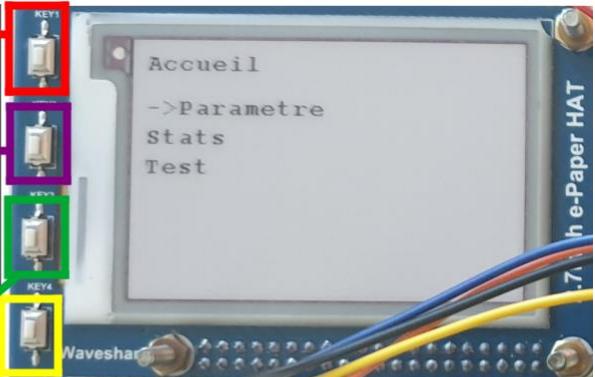
Test unitaire : Les différents boutons

```
void IHM::btnHome()
{
    _setEcranActifAccueil();
    afficher();
};

void IHM::btnPlus()
{
    Item *itemActif = _getEcranActif()->getItemActif();
    if(itemActif != nullptr){
        _getEcranActif()->itemSuivant();
        afficher();
    }
};

void IHM::btnMoins()
{
    Item *itemActif = _getEcranActif()->getItemActif();
    if(itemActif != nullptr){
        _getEcranActif()->itemPrecedent();
        afficher();
    }
};

void IHM::btnValid()
{
    if(_getEcranActif()->getAccueil() == true){
        unsigned int ItemActif = _getEcranActif()->indexItemActif();
        _setEcranActif(ItemActif);
        afficher();
    }
};
```



On appuie donc sur le bouton plus (btnPlus())

On se retrouve sur l'item Test

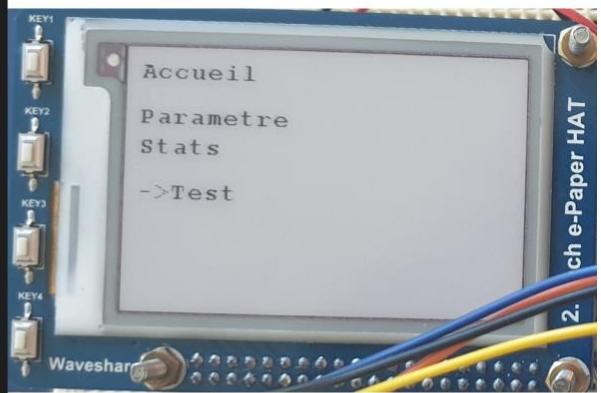
```
void Ecran::afficher()
{
    string texteItem ;

    printf("%s \r\n", _titre.c_str());
    _epaper->effacer();
    _epaper->Ecrire(10, 10, _titre.c_str(), COLORED);

    for(int i=0;i<_vectItems.size();i++)
    {
        if (_vectItems.at(i)->getActif() == true)
        {
            texteItem = "->" + _vectItems.at(i)->getText();
            _epaper->Ecrire(10, 40 + i*25, texteItem.c_str(), COLORED);
        }
        else
        {
            texteItem = _vectItems.at(i)->getText();
            _epaper->Ecrire(10, 40 + i*20, texteItem.c_str(), COLORED);
        }

        printf("%s \r\n", texteItem.c_str());
        ThisThread::sleep_for(SLEEP_TIME);
    }

    _epaper->majAffichage();
}
```



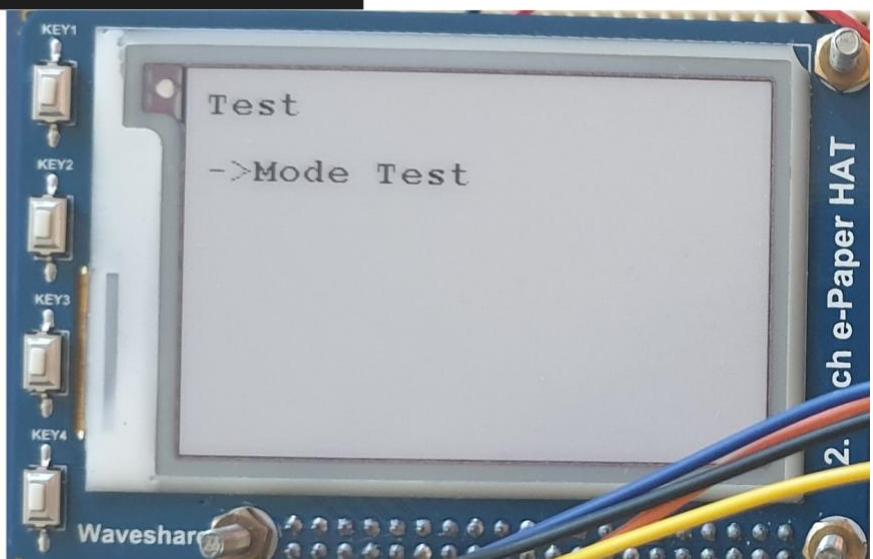
On récupère l'Item suivant actif
puis on met à jour l'écran.

On appuie alors sur le bouton Valid (btnValid())

On va alors afficher l'Item actif et l'écran qui lui est associé (Là, ce sera l'Item Test)

```
void IHM::btnValid()
{
    if(_getEcranActif()->getAccueil() == true){
        unsigned int ItemActif = _getEcranActif()->indexItemActif();
        _setEcranActif(ItemActif);
        afficher();
    }
};
```

L'écran se met à jour sur
l'Ecran Test avec l'item Mode Test.



E. Bilan technique

Tache à réaliser :

- Interface homme machine sur l'e-paper avec différents menu (Paramétrier ,Stats ,Test et Accueil)
- Mode édition pour éditer le nombre de capteur pour la collecte
- Réinitialisation système pour remise des valeurs des capteurs par défaut

Tache réalisée :

- Interface homme machine sur l'e-paper avec différents menu (Paramétrier ,Stats ,Test et Accueil) avec fonctionnement des boutons (Valide, Home, Plus et Moins)

L'interface ayant été finie, il ne reste plus) à faire que le mode édition et le réinitialiser système.

F. Partie Physique

Fonctionnement d'un écran Epaper :

Le composant primaire est une microcapsule qui contient des particules blanches chargées positivement et des particules noires chargées négativement ;

- Lorsque l'on applique un champ électrique négatif, les particules blanches se placent sur une extrémité de la capsule et les noires sur l'autre ;
- En plaçant des millions de ces capsules sur une surface et en les commandant par des champs électriques, on peut générer une image en deux couleurs ;
- Par simple adjonction d'une matrice de filtres on obtient une version couleur (4 096 couleurs).
- Le système peut avoir quatre champs électriques pour chaque microcapsule permettant d'avoir trois niveaux de gris en plus du blanc et du noir : 100 % blanc ; 75 % blanc et 25 % noir ; 50 % blanc, 50 % noir ; 25 % blanc, 75 % noir et 100 % noir.

Les principaux avantages de ce système sont :

- Sa très faible consommation de courant électrique : ce système consomme uniquement lors du changement de page, ensuite la page reste telle quelle sans consommation d'énergie ; comme pour un livre, l'éclairage vient de la lumière ambiante.
- La possibilité d'avoir des écrans souples, comme l'est le papier.

Ce système est bistable, une seule impulsion de polarisation suffit à définir si le pixel est « allumé » ou non. De plus, le très fort contraste d'affichage rend l'éclairage direct ou indirect inutile, tout ceci entraîne un gain de consommation énergétique appréciable. Enfin, le support peut être semi-souple, ce qui est une évolution par rapport aux procédés d'affichage classiques.

Contrairement aux techniques d'affichage classiques qui nécessitent un rétroéclairage ou l'émission de photons, le papier électronique est purement réflectif et utilise la lumière ambiante de la même manière que le papier classique.

Un papier électronique doit pouvoir afficher du texte et des images indéfiniment, sans consommer d'énergie, que ce soit pour l'affichage ou pour un éventuel système de traitement de données, et doit permettre le changement de ce qu'il affiche.

La plupart des papiers électroniques consomment de l'énergie uniquement lorsque le contenu affiché est modifié.

Les pixels d'un tel système doivent donc posséder plusieurs états distincts stables, de manière à garder intact le contenu affiché en l'absence de source d'énergie.

Schéma de fonctionnement des pixels :

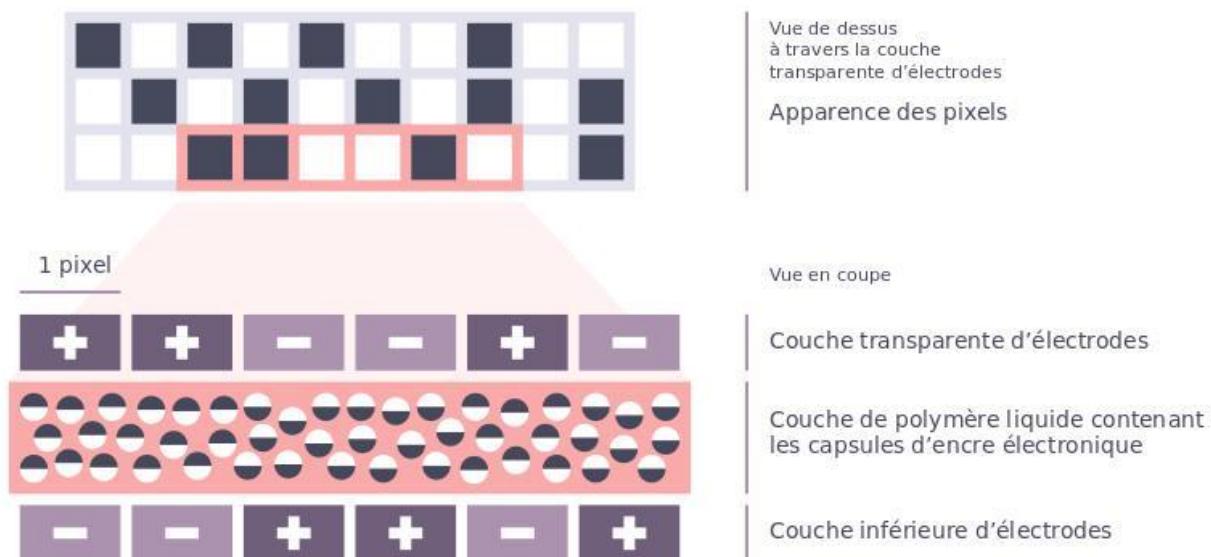
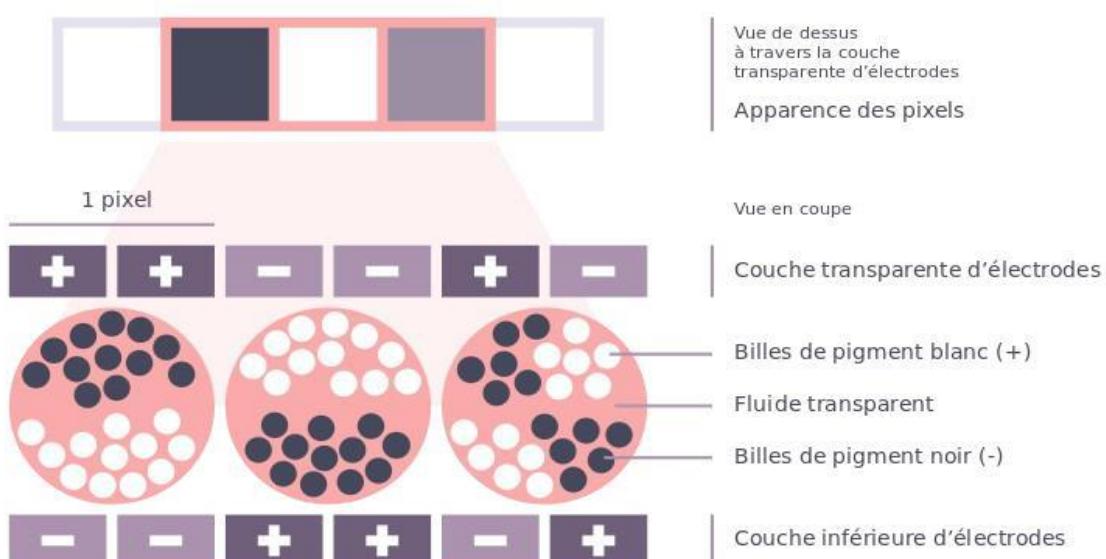


Schéma de fonctionnement des pixels zoomé :



Il existe plusieurs approches au papier électronique, et plusieurs compagnies développant différentes technologies indépendamment. Le papier électronique a ainsi connu des applications aussi diverses que :

- liseuse ;
- remplaçant d'un écran à cristaux liquides (lecteurs portables, téléphones...) ;
- remplaçant d'un écran à affichage électro-chimique ;
- écran magique, etc.

II. Etudiant Thibaut Jouandoudet

A. Description partie personnelle

C'est un collecteur de trafic campagne de mesures). Il doit permettant de **calculer**, **d'enregistrer** , **d'envoyer** et de **calculer** des statistiques sur le nombre de véhicules ou d'essieux passant sur une chaussée.

Les statistiques sont envoyées périodiquement par l'intermédiaire du réseau Sigfox.

Ce sous système est prévu pour être installé au bord d'une chaussée. Il doit être autonome en énergie.

Des tuyaux pneumatiques permettent de détecter le passage d'un essieu par variation de pression. Des détecteurs pneumatiques sont positionnés aux extrémités des tuyaux afin de transformer la variation de pression en signal électrique.

Ce système peut être installé avec :

- un tuyau, dans ce cas seul de comptage d'essieu est possible,
- deux tuyaux : dans ce cas il est possible de différentier les voitures des poids lourds (distance entre essieux supérieur à 3,45m) et de calculer la vitesse des véhicules.

But : Développer un système qui vise à détecter les passages des véhicules , Identifier le type de véhicule et Calculer vitesse du véhicule.

B. Conception détaillée

Diagramme de déploiement

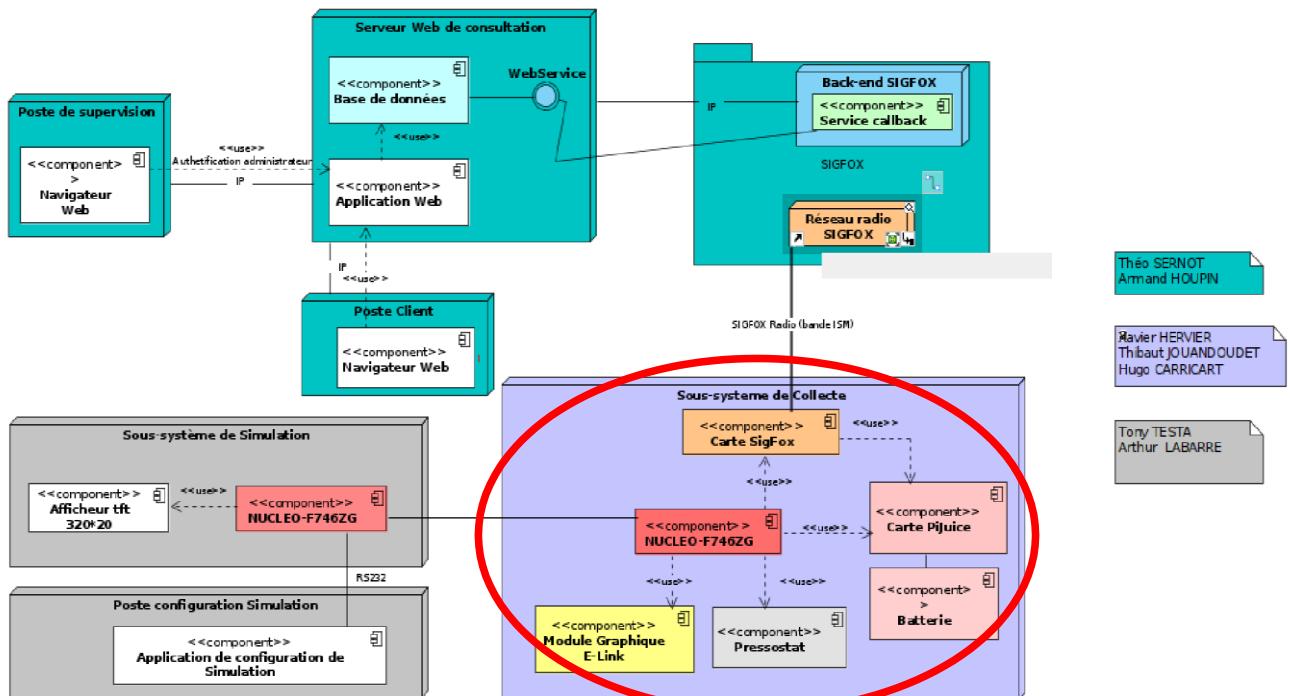


Diagramme de déploiement Collecte

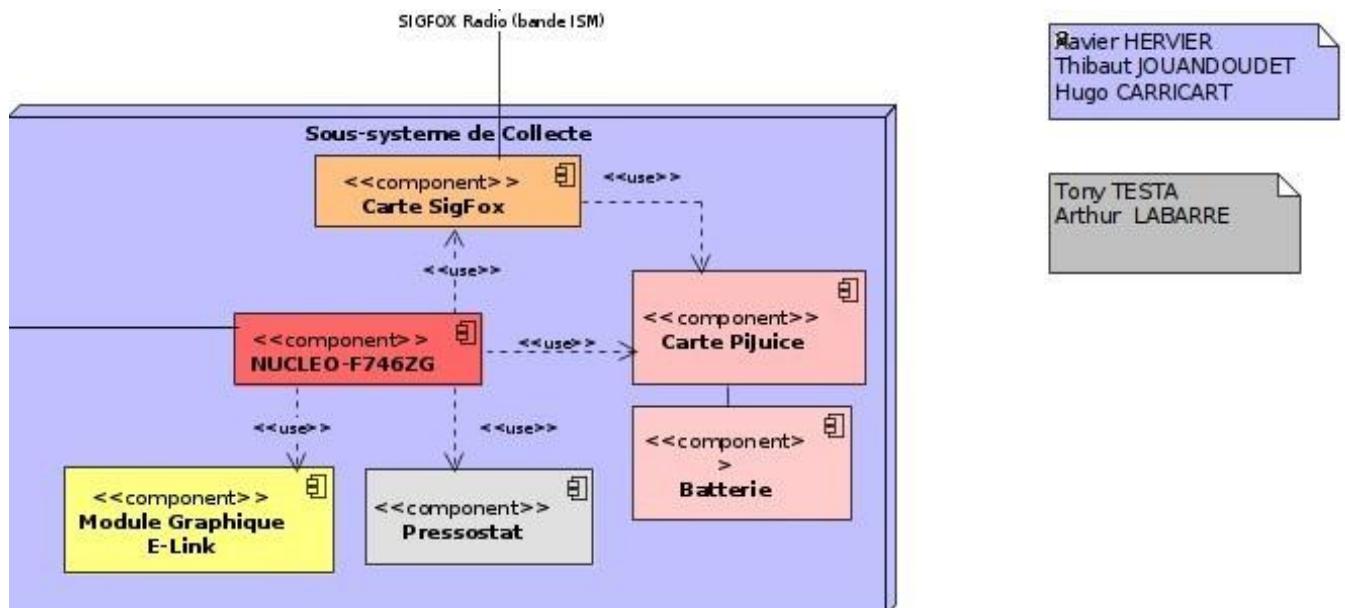
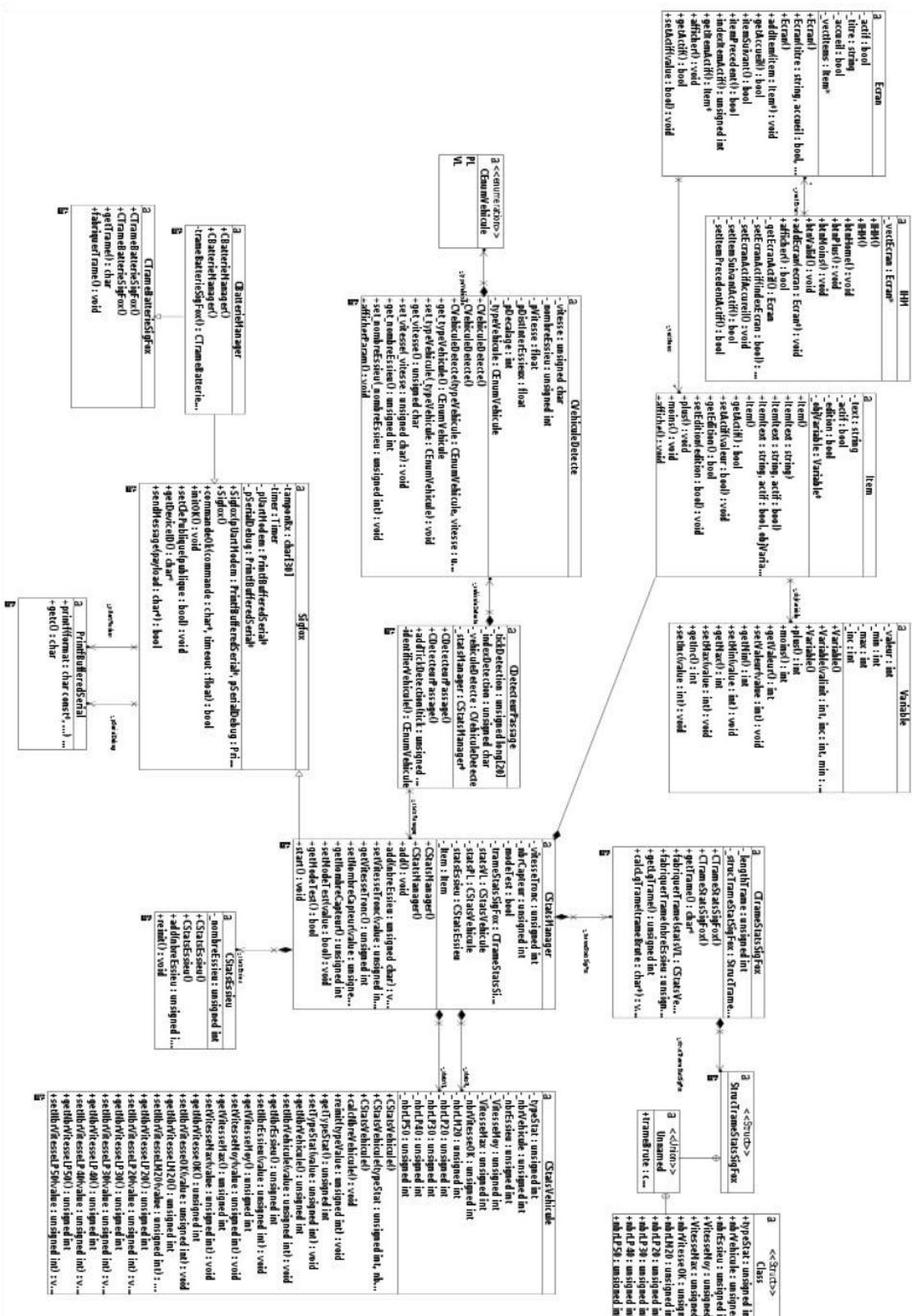


Diagramme de déploiement Collecte zoomer sur partie personnelle

Diagramme de classe Collecte



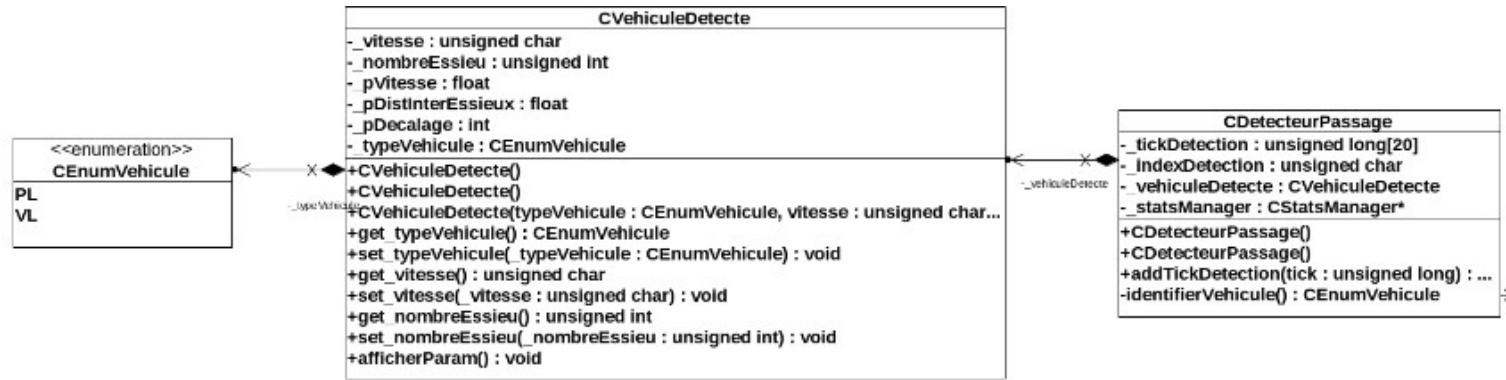
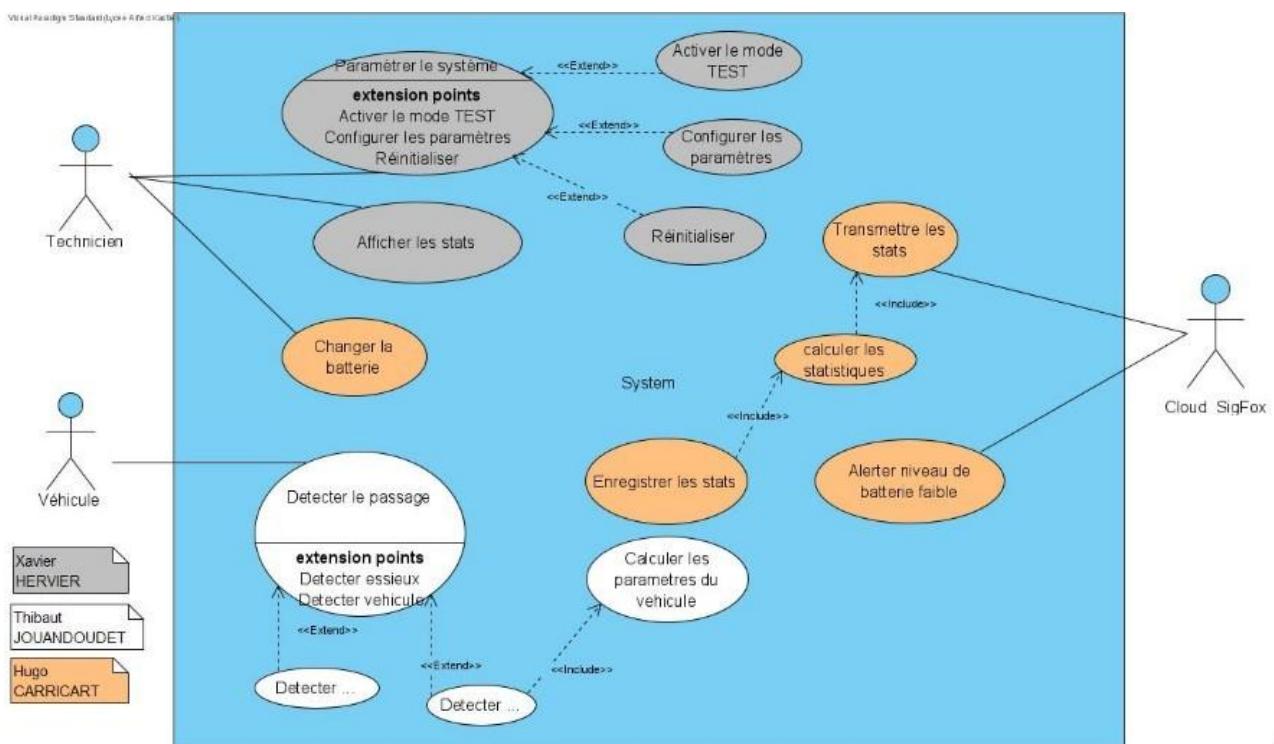


Diagramme de classe de l'étudiant

Classe	Fonction
CEnumVehicule	Cette classe permet d'énumérer PL (Poids lourd) et VL (Véhicule Léger)
CVehicleDetecte	Cette classe permet de détecter un véhicule et d'obtenir sa vitesse , le type de véhicule (PL / VL), le nombre d'éssieux et utilise un objet de CEnumVehicule.
CDetecteurPassage	Cette classe de detecter le passage d'un véhicule et de l'identifier et utilise un objet de la classe CvehiculeDetecte.

Diagramme de cas d'utilisation Collecte



Déetecter essieux

Déetecte le nombre d'essieux d'un véhicule ayant roulé sur les détecteurs pneumatiques.

Steps	Procedures	Expected Results
1. Déetecter essieux	Faire passer un véhicule sur les détecteurs pneumatiques (pressostats)	Ouverture / fermeture du pressostats provocant un frontmontant

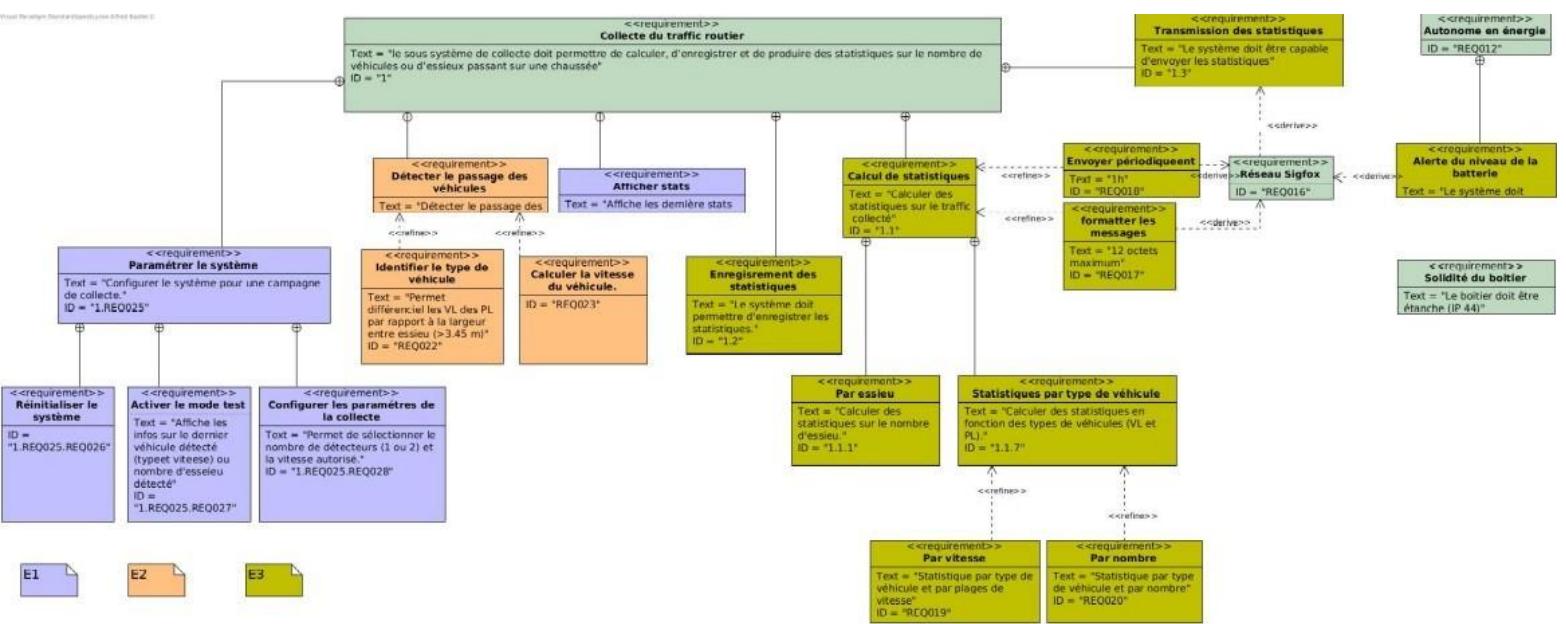
Déetecter véhicule

Steps	Procedures	Expected Results
1. Déetecter véhicule	Faire passer un véhicule sur les détecteurs pneumatiques (pressostats)	Ouverture / fermeture du pressostats provocant un frontmontant

Calculer les paramètres du véhicule

Steps	Procedures	Expected Results
1. Calculer les paramètres du véhicule	Faire passer un véhicule sur les détecteurs pneumatiques (pressostats) et détermineront empattement.	Empattement > 3,45 m (PL) Sinon VL

Diagramme d'exigence



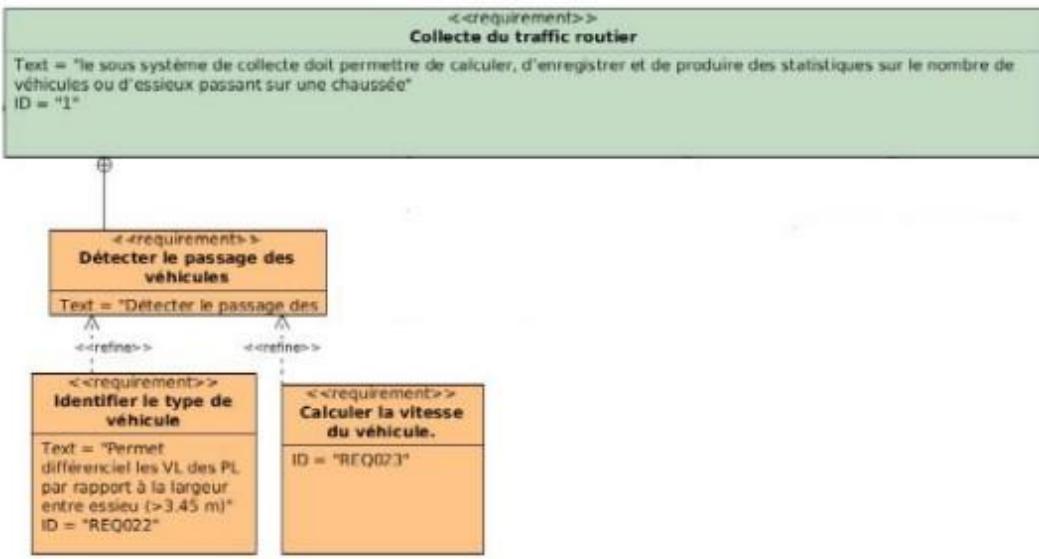


Diagramme d'exigence zoomer sur partie personnelle

C. Implémentation

→ Mise en œuvre des techniques / technologies utilisées.

Mbed : Procédure et mise en place de la chaîne de cross-compilation.



Outils utilisés :

- Visual Studio Code :

[https://code.visualstudio.com/sha/download?
build=stable&os=win32-x64-user](https://code.visualstudio.com/sha/download?build=stable&os=win32-x64-user)

- Server GDB SEGGER : https://www.segger.com/downloads/jlink/JLink_Windows_V700a.exe

- Arm-gcc-none-eabi : <https://developer.arm.com/-/media/Files/Downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-win32.zip?revision=ffcaa06c-940d-4319-8d7ed3070092d392&la=en&hash=130196DDF95B0D9817E1FDE08A725C7EBFBFB5B8>

- Mbed – cli : <https://github.com/ARMmbed/mbed-cli-windows-installer/releases/latest>

→ Ordre de la procédure (sous Windows) :

- 1) Installer Visual Studio Code et l'extension Cortex-Debug
- 2) Installer le SEGGER
- 3) Décompresser l'archive d'ARM à l'emplacement de son choix
- 4) Installer mbed-cl (ATTENTION : Redémarrer l'ordinateur et tester dans l'invite de commande la commande : mbed)
- 5) Si commande applet impossible alors voir version de Python et tester un upgrade de pip avec la commande : python -m pip install --upgrade pip // Redémarrer à nouveau)
- 6) Cloner un template fonctionnelle de mbed :
http://gogs.lyceekastler.fr/hug.Carricart/Mbed_Templat e/

7) Réadapter le fichier de configuration : launch.json et c_cpp_properties.json

Désignation :	Caractéristiques techniques :
8) Coder ! Sous système de collecte	Platine nucléo langage de programmation C++ / mbed-os

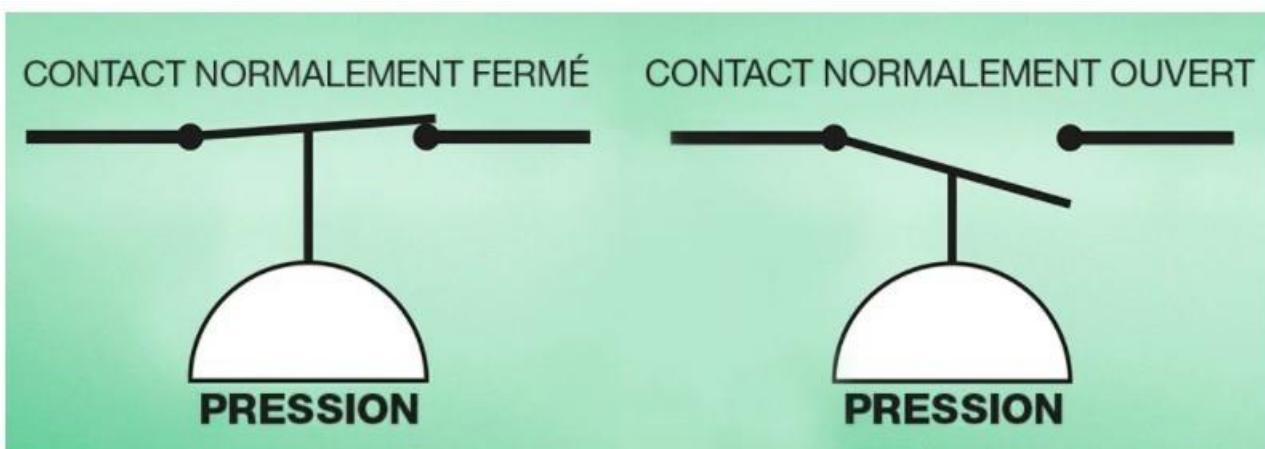
D. Test

Mise en oeuvre du pressostat



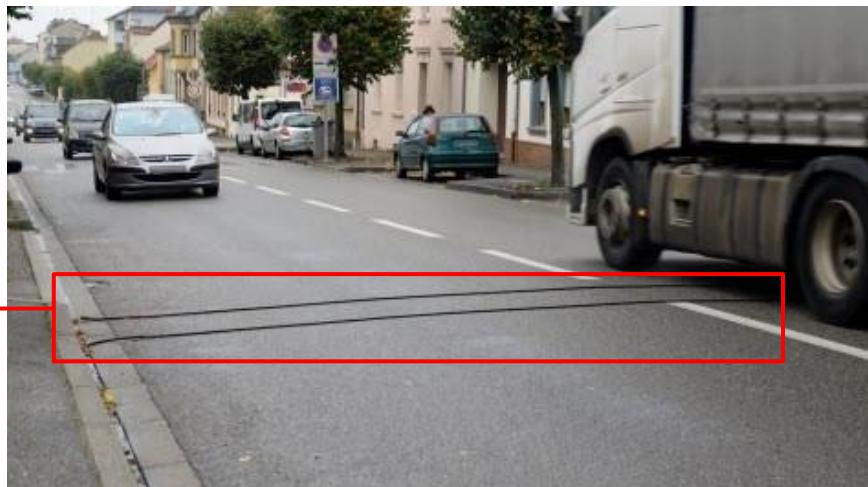
Pour détecter le passage d'un véhicule nous utilisons des pressostats, ce sont des détecteurs pneumatiques qui sont positionnés aux extrémités des tuyaux afin de transformer la variation de pression en signal électriques.

Ce pressostat est conçu comme une interface économique entre les systèmes de pression et les circuits imprimés.

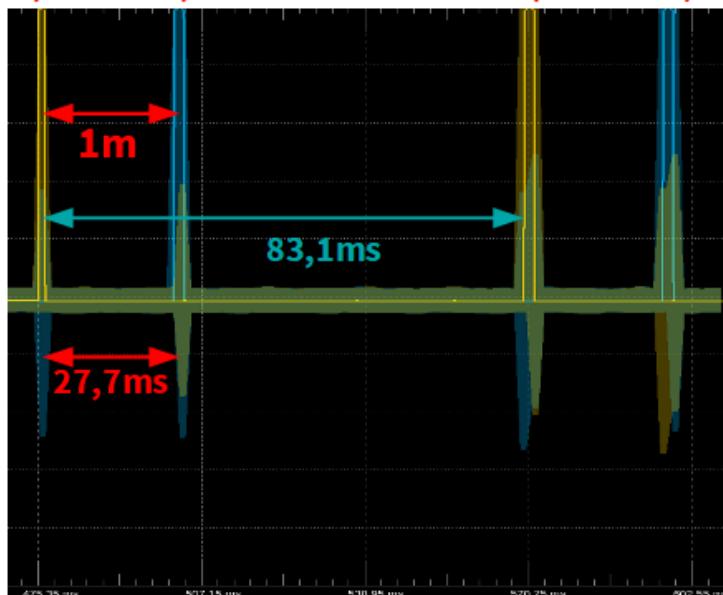


Un pressostat normalement ouvert est en position ouverte quand il n'est soumis à aucune pression. Quand il est soumis à une pression donnée, le contact passe en position fermée

Nous positionnons les deux tuyaux d'un espace de 1 m



E1/P1 E1 / P2 E2/P1 E2/P2



↔ **Calcul de la vitesse** = 1/temps (E1P1 & E1P2)
Exemple : $1\text{m} / 27,7 \text{ ms} = 0,0361 \text{ m/ms} * 3600 = 130\text{KM/h}$

↔ **Empattement** = Temps entre E1P1 & E2P1 / Temps entre E1P1 & E1P2
Exemple : $83,1 \text{ ms} / 27,7\text{ms} = 3\text{m}$

Cet extrait de code sera utile pour effectuer des calculs de temps afin d'identifier le véhicule.

```
void detectP1(){
    CptP1++ ;
    CptTicker[indexCptTickerP1++][0] = us_ticker_read();
    if (indexCptTickerP1==3) indexCptTickerP1 = 0 ;
}

void detectP2(){
    CptP2++ ;
    CptTicker[indexCptTickerP2++][1] = us_ticker_read();
    if (indexCptTickerP2==3) indexCptTickerP2 = 0 ;
}
```

Timer en us

→ Pour DetectP1 le timer range ses valeurs de temps dans la colonne 0

→ Pour DetectP2 le timer range ses valeurs de temps dans la colonne 1

```
printf ("\n\r\t --- tab CptTicker ---");
for (int i = 0 ; i <3 ; i++) {
    if ( CptTicker[i][1] >= CptTicker[i][0] )
        dif = CptTicker[i][1]- CptTicker[i][0] ;
    else
        dif = 0 ; //CptTicker[i][0]- CptTicker[i][1] ;
```

Différence des temps

Si la colonne 1 est supérieure ou égale à la colonne 0 alors on soustrait celle-ci

```
printf ("\n\r\t %d : %lu ; %lu ; %lu", i, CptTicker[i][0], CptTicker[i][1], dif) ;
```

Affichage des temps dans deux colonnes.

Aucune manipulation particulière n'est nécessaire pour les temps passés car le code de minuterie commun le détectera et appellera

```
void affCptTicker()
{
    unsigned long dif = 0 ;

    double vitCalule ;
    double interessieu ;
    double interessieu1 ;
    double interessieu2 ;
    double vitCalcule;
    double vitCalcul;
```

Déclaration des variables afin de tester la détection.

```

59 // Permet de calculer la vitesse en KM/H
60 vitCalule = dif/1000 ;
61 vitCalcule = (1/vitCalule);
62 vitCalcul = vitCalcule * 3600;

```

```

66 interessieu1 = (vitCalcul * 1000 ) / 3600000 ; // Pasage en km/h en m/ms
67 interessieu2 = (CptTicker[0][1]- CptTicker[0][0] + CptTicker[1][1]- CptTicker[1][0] + CptTicker[2][1]- CptTicker[2][0] ) / 1000 ; // différence des capteurs en ms
68 interessieu = interessieu2 * interessieu1 ; // différence des capteurs * la vitesse en m/ms

```

```

| printf ("\n\r\tVitesse Calculer: = %lf KM/h" ,vitCalcul) ;
| printf ("\n\r\tInteressieu: = %lf m\n\r" , interessieu) ;

```

```

cpt1 : 6 ; cpt2 : 6
--- tab CptTicker ---
0 : 67359696 ; 67386704 ; 27008
1 : 67455707 ; 67482716 ; 27009
2 : 55031654 ; 55058664 ; 27010
Vitesse Calculer: = 133.333333 KM/h
Interessieu: = 3.000000 m

```

Affichage de la vitesse calculée ainsi que la distance inter-essieux

```

75 | if (interessieu >= 3.45 )
76 {
77 printf("Vehicule Lourd\r\n");
78 }
79
80 else
81 {
82 printf("Vehicule Leger\r\n");
83 }
84
85 }

```

```

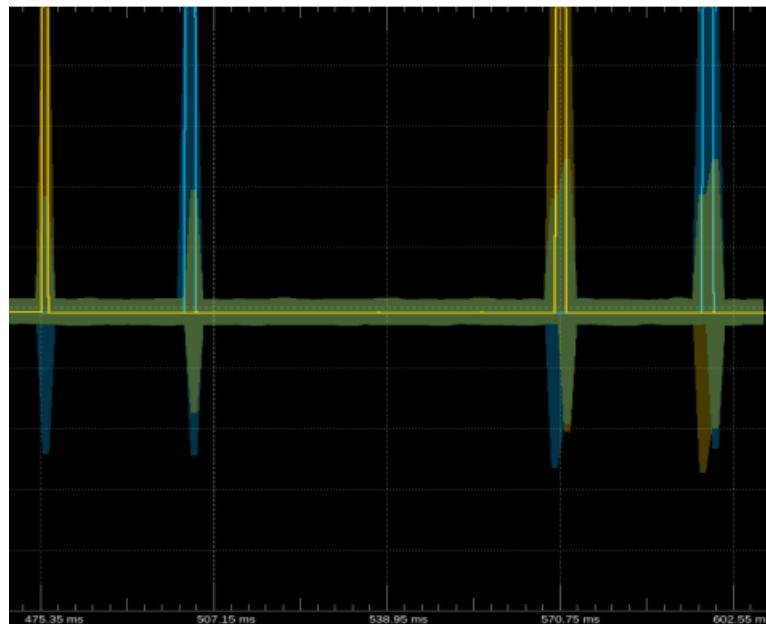
cpt1 : 6 ; cpt2 : 6
--- tab CptTicker ---
0 : 67359696 ; 67386704 ; 27008
1 : 67455707 ; 67482716 ; 27009
2 : 55031654 ; 55058664 ; 27010
Vitesse Calculer: = 133.333333 KM/h
Interessieu: = 3.000000 m
Vehicule Leger

```

Affichage des paramètres du véhicule

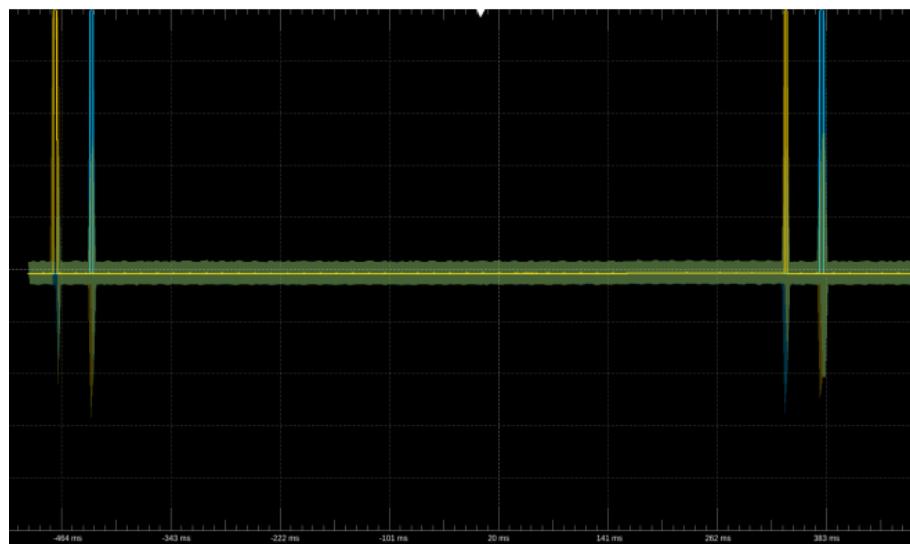
Inter-essieux calculé = 3m , donc Véhicule Léger

Passage à 130km/h
2,44 d'empattement



Nous constatons que la detection des essieux est courte car le véhicule détecté à une distance inter-essieux seulement de 2,44m avec une grande vitesse

Passage à 90km/h
3,7 d'empattement



Ici nous constatons l'inverse , plus l'empattement du véhicule est grand avec une allure moyenne, plus la detection des essieux sera longue.

E. Bilan Technique

Tache à réaliser :

- Adapter mon code au diagramme de classe fourni.

Le principe de détection et le calcul du véhicule étant fini il ne reste plus que à modifier et adapter mon programme.

F. Partie Physique

La partie de notre projet en physique est la mise en œuvre du pressostat , l'analyse du signal aux bornes du pressostat puis l'élimination du rebond.

Description du pressostat

Dans notre projet nous utilisons un pressostat de référence : **LEFOO – LFS-03-15mbar**

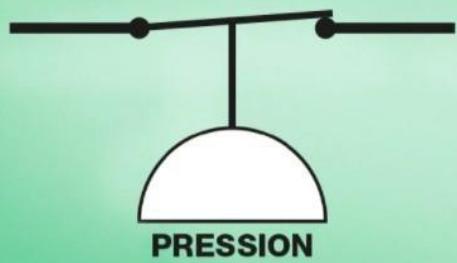
Gamme de pression	Pression: +15 à +2500 mbar Tolérance: +/- 15%
Fonction électrique	1 pôle NO 1 pôle NC



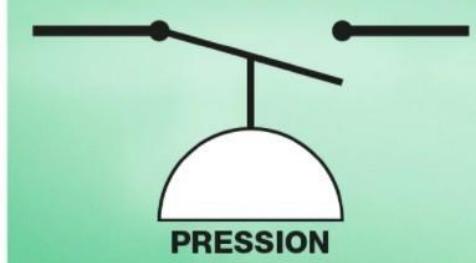
Certains pressostats renferment des contacts en position ouverte quand ils ne sont pas soumis à une pression, on les appelle « normalement ouverts » (NO).

À l'inverse, quand les contacts sont en position fermée quand ils ne sont soumis à aucune pression, on les appelle « normalement fermés » (NF).

CONTACT NORMALEMENT FERMÉ



CONTACT NORMALEMENT OUVERT



Analyse du signal aux bornes du pressostat

Nous avons d'abord effectué un test à 30km/h (représentation du premieressieu)

Nous constatons que nous avons plusieurs rebonds.

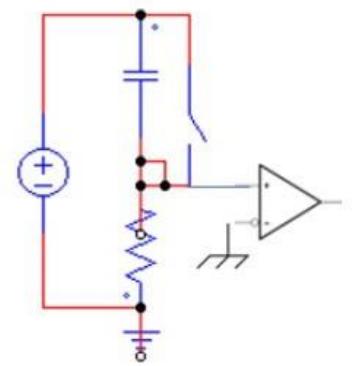
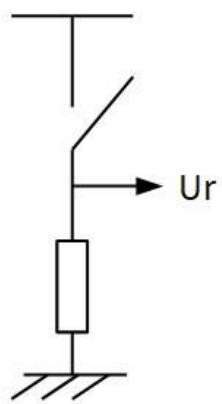


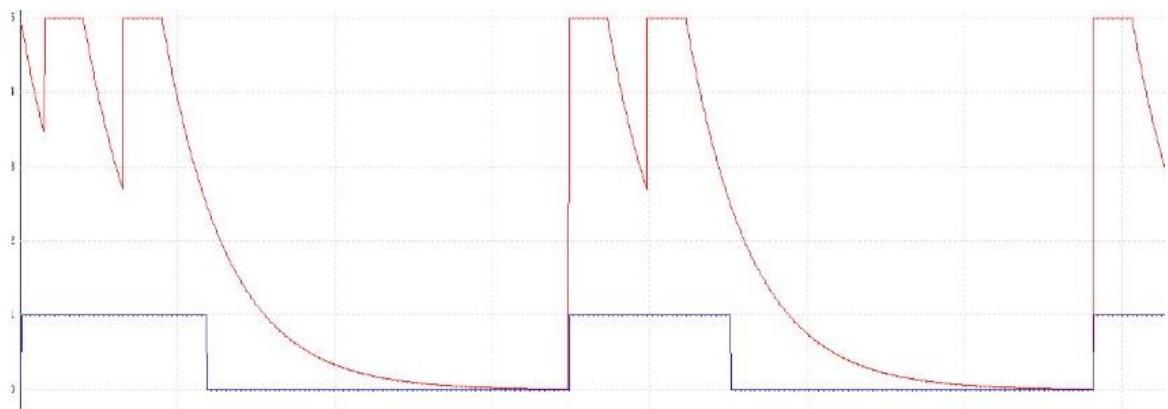
Nous avons décidé de mettre en place un circuit RC afin de lisser le signal de maniere à ce que ce rebond ne puisse pas être interprétré comme une variation de tension

→ Si une voiture est détectée , le pressostat est fermé donc $U_r = 5V$

S'il y a aucune voiture , $U_r = R * I$ or $I = 0$

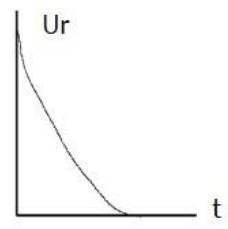
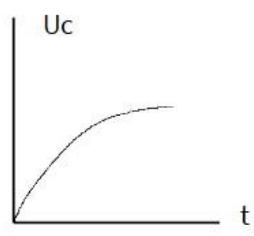
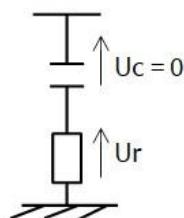
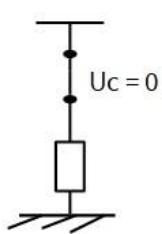
$$U_r = 0$$





Si interrupteur fermé $V = 1$

Si interrupteur $V = 0$



III. Etudiant Hugo CARRICART

A. Description partie personnelle

- Situation du travail :

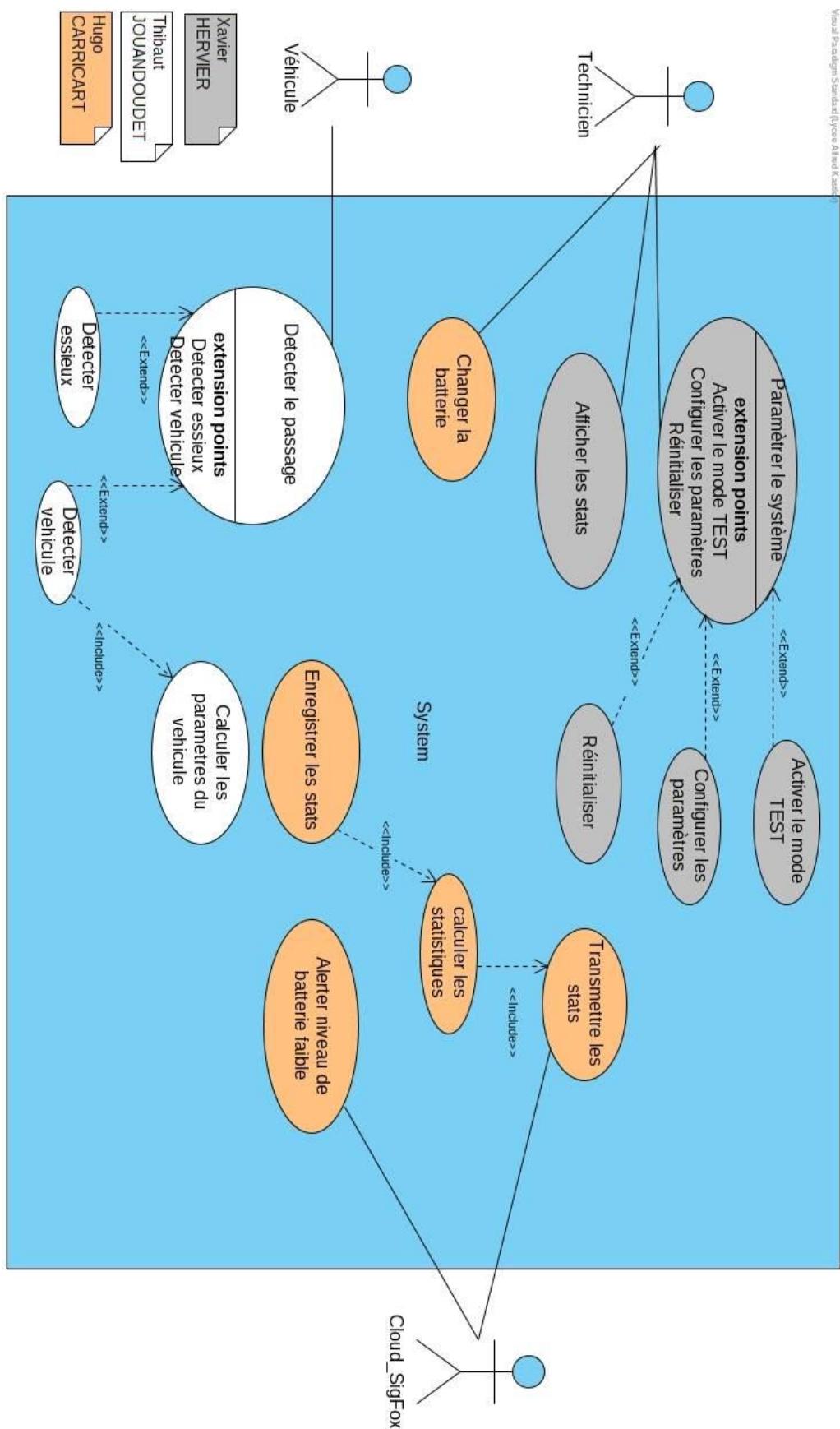
(Sous -système de collecte) :

Le sous-système de collecte a pour objectif d'acquérir les données d'une campagne de mesure et dans établir des statistiques. Une fois celle-ci calculer, elles sont envoyées au serveur OVH via le réseau Sigfox.

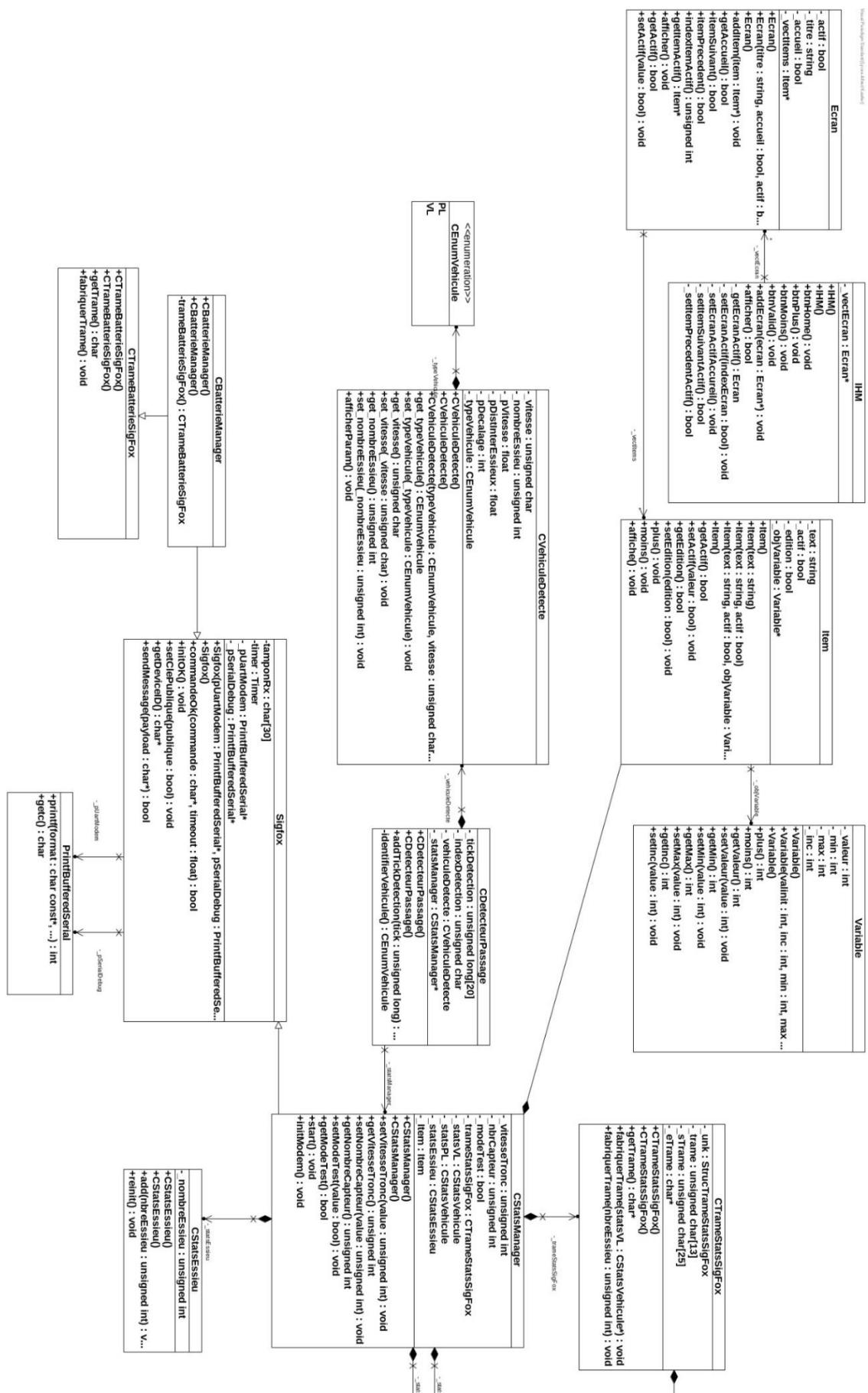
Pour ma part j'étais chargé de développer les classes permettant la communication de la carte avec le modem Sigfox, la fabrication des trames de données et le calcul des statistiques de la campagne de mesure, ainsi qu'une classe permettant la communication entre la carte Nucléo et la batterie.

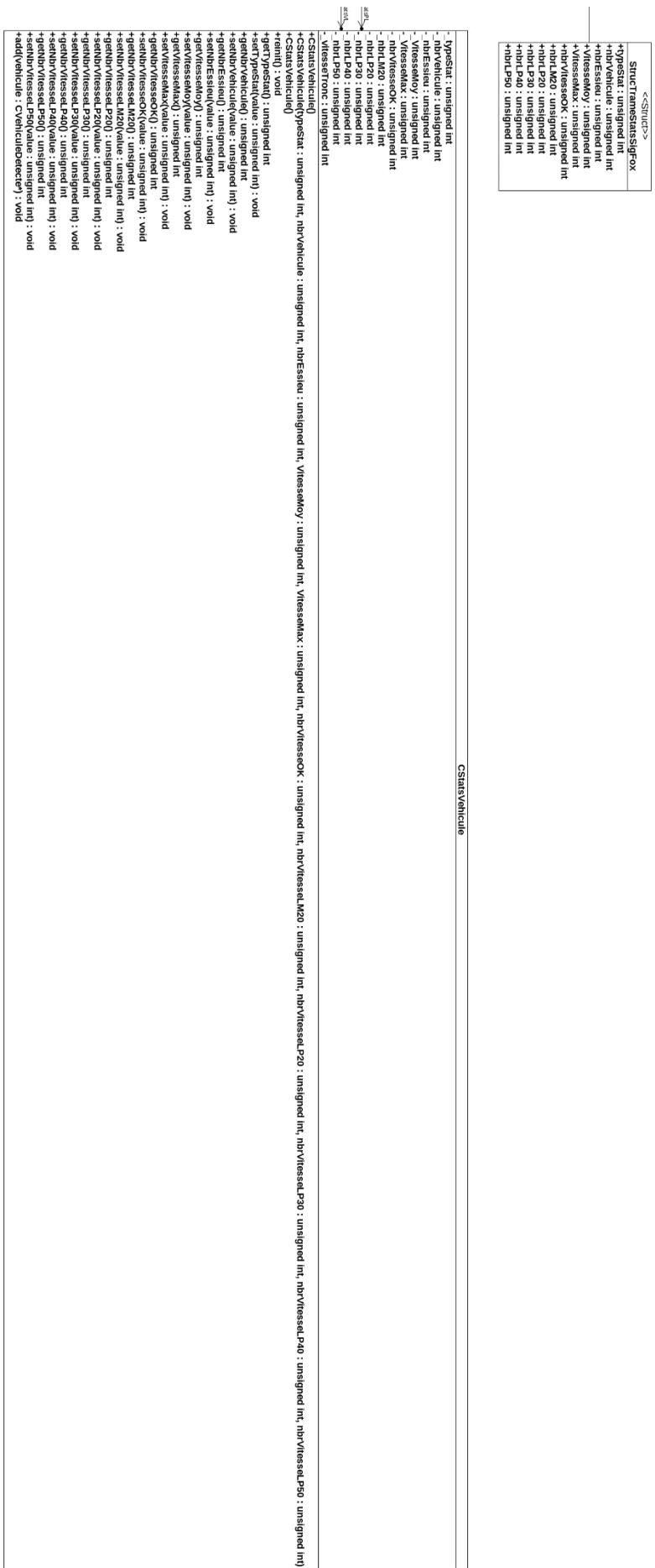
B. Conception détaillée

- Diagramme de cas d'utilisation

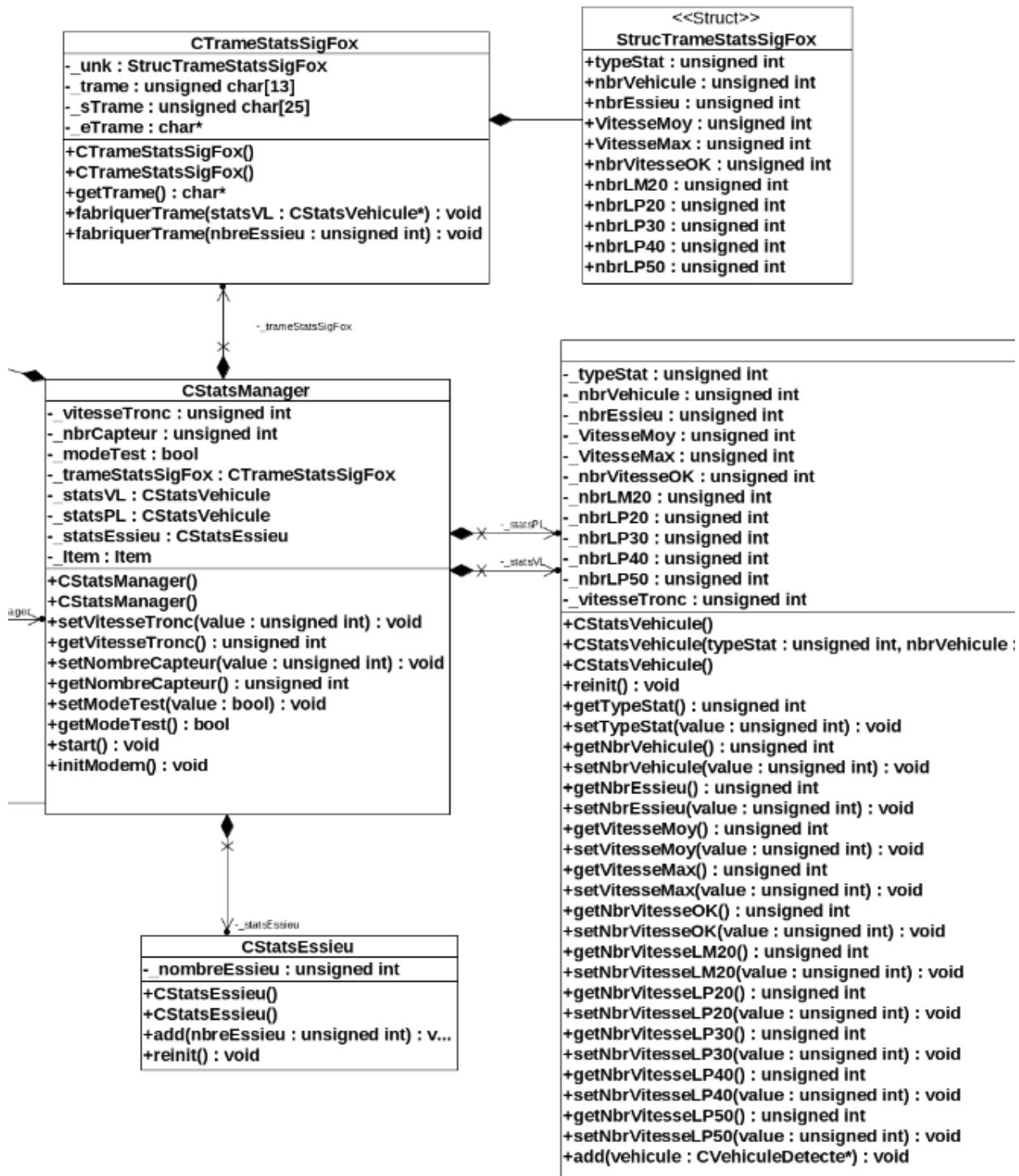


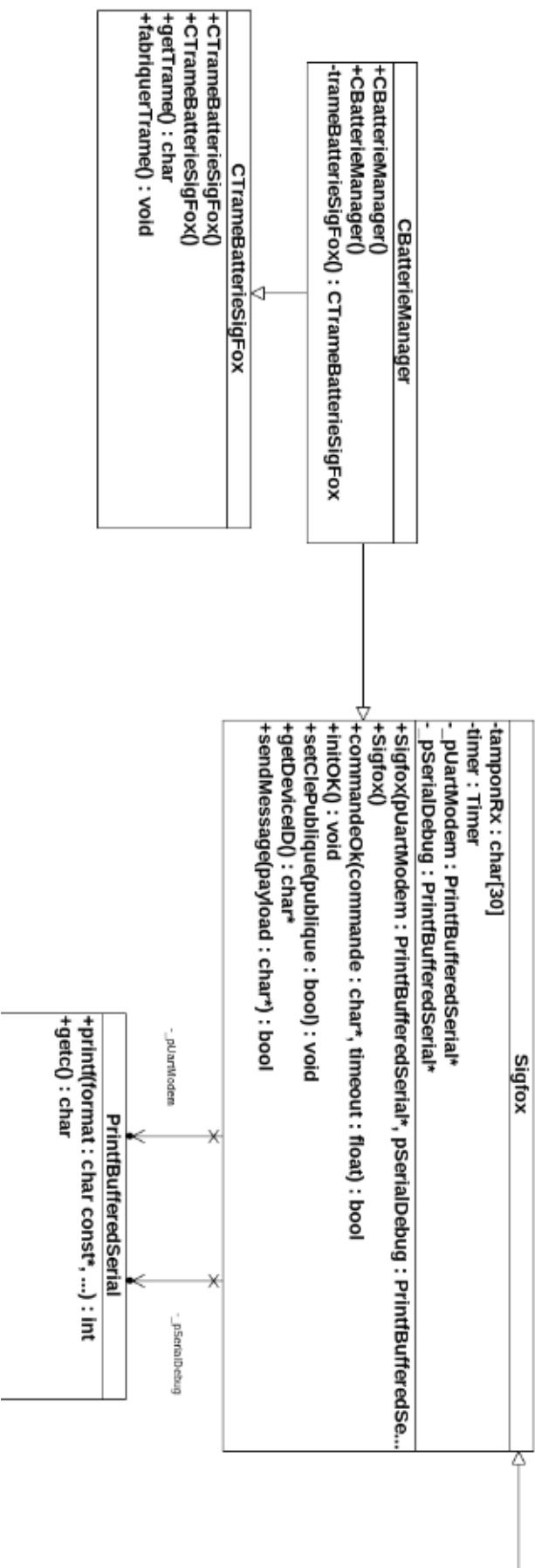
- Diagramme de classe de conception du Sous-système de Collecte :





- Diagramme de classe (Mes classes) :





Description :

● **Alerter niveau de batterie faible**

ID : UC64.UC15

Le système doit alerter à l'administrateur si la batterie atteint un niveau faible.

● **6. Calculer les statistiques**

ID : UC64.UC16

Calculer les statistiques sur le trafic collecté. /

Les statistiques peuvent être trier par essieu, par type de véhicule, qui sont eux trier sois par nombre ou par vitesse.

Include

● [Transmettre les stats](#), ● [calculer les statistiques](#)

Included by

● [Calculer les statistiques](#), ● [Enregistrer les stats](#)

● **Changer la batterie**

ID : UC64.UC32

Changer la batterie à partir du moment où le niveau de batterie est faible.

● **Réinitialiser**

ID : UC31

Réinitialiser les valeurs des paramètres (vitesse autorisé, nombre de capteurs)

● **Transmettre les stats**

ID : UC64.UC14

Le système doit être capable d'envoyer les statistiques



Cloud Sigfox

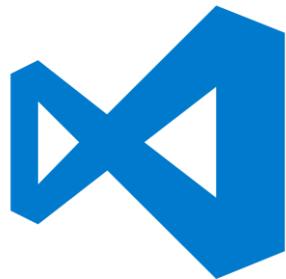
ID: AC18

Use Cases

● Consulter, ● [Transmettre les stats](#), ● [Alerter niveau de batterie faible](#), ● Collecter les stats, ● Alerter niveau de batterie faible, ● Collecter

C. Implémentation

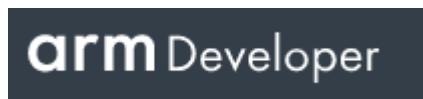
Outils utilisés et matériel utilisés :



- Visual Studio Code :
<https://code.visualstudio.com/sha/download?build=stable&os=win32-x64-user>



- Server GDB SEGGER :
https://www.segger.com/downloads/jlink/JLink_Windows_V700a.exe



- Arm-gcc-none-eabi :
<https://developer.arm.com/-/media/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-win32.zip?revision=ffcaa06c-940d-4319-8d7e-d3070092d392&la=en&hash=130196DDF95B0D9817E1FDE08A725C7EBFBFB5B8>



- Mbed – cli :
<https://github.com/ARMmbed/mbed-cli-windows-installer/releases/latest>

Matériel :

Carte Nucléo : (sous mbed-os)



Modem Sigfox :

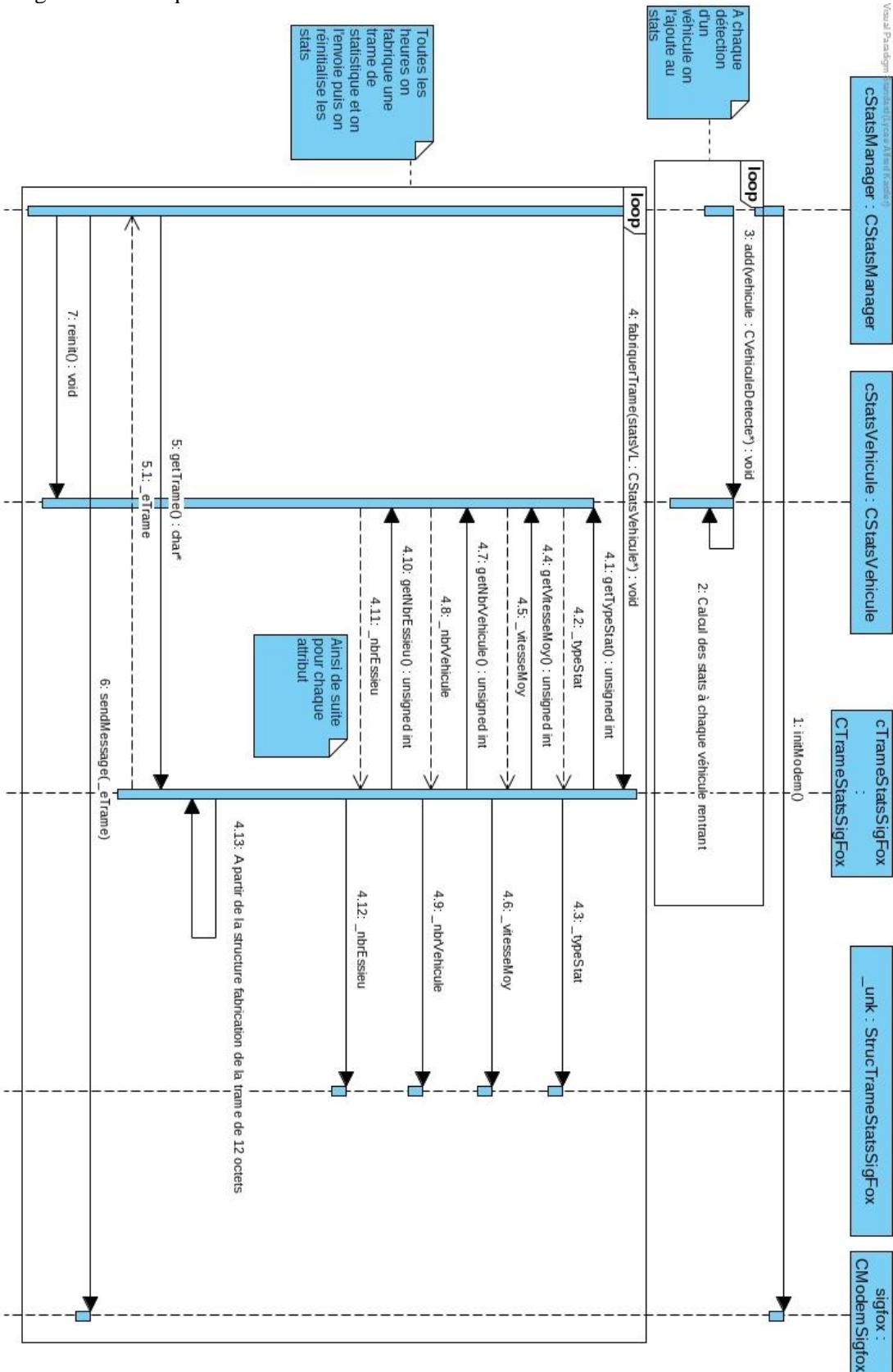


HAT Pi-Juice + Batterie BP7X :



Fonctionnement :

Diagramme de séquence :



Classe CStatsManager :

```
void CStatsManager::start()
{
    initModem();

    _statsPL.setVitesseTronc(_vitesseTronc);
    _statsVL.setVitesseTronc(_vitesseTronc);

    // Début de la boucle

    // event détection du véhicule

    if (&vehicule.get_typeVehicule == VL){
        _statsVL.add(&vehicule);
    }
    if (&vehicule.get_typeVehicule == PL){
        _statsPL.add(&vehicule);
    }

    // Au bout d'une heure
    _trameStatsVL.fabriquerTrame(&_statsVL);
    _trameStatsPL.fabriquerTrame(&_statsPL);

    printf("Trame apres get : %s \r\n", _trameStatsVL.getTrame());

    _modemSigFox.sendMessage(_trameStatsVL.getTrame());

    // On réinitialise les stats.
    _statsVL.reinit();
    _statsPL.reinit();

    // Fin de boucle
}

void CStatsManager::initModem()
{
    // initialisations
    pc.set_baud(BAUDS_DEBUG);
    uartSigfox.set_baud(BAUDS_SIGFOX);
    pc.printf("Demarrage du programme Kolantr \r\n");
    pc.printf("Initialisation modem \r\n");

    // active le Modem
    _modemSigFox.initOK();
    // sélectionne le réseau
    _modemSigFox.setClePublique(SIGFOX_SIMUL);
}
```

Dans un premier temps on vient initialiser le modem, en suivant on vient définir la vitesse du tronçon pour calculer les statistiques.

Puis selon si le véhicule est un véhicule léger ou un poids lourd on l'ajoute dans les statistiques lui correspondant.

Ensuite fabrication de la trame des statistiques.

Et en dernier on envoie la trame récupérer.

Puis on réinitialise.

Et on recommence.

Définition de la vitesse de communication
Vérifie si le modem est actif.
Défini que le modem doit fonctionner sur le simulateur Sigfox.

Classe CModemSigfox :

```
void Sigfox::setClePublique(bool publique)
{
    while (true)
    {
        char *commande;
        char *info;
        if (publique)
        {
            commande = "ATS410=1";
            info = "reseau simulateur";
        }
        else
        {
            commande = "ATS410=0";
            info = "reseau SIGFOX";
        }

        if (commandeOk(commande, 2))
        {
            if (DEBUG_SIGFOX)
                _pSerialDebug->printf("modem sur %s OK\r\n", info);
            break;
        }
        else if (DEBUG_SIGFOX)
            _pSerialDebug->printf("erreur modem sur %s\r\n", info);
        ThisThread::sleep_for(1000);
    }
}

bool Sigfox::sendMessage(char *payload)
{
    char chaine[31];
    sprintf(chaine, "AT$SF=%s", payload);
    if (DEBUG_SIGFOX)
        _pSerialDebug->printf("envoi message Sigfox : %s\r\n", payload);
    bool result = commandeOk(chaine, 10);
    if (DEBUG_SIGFOX)
        if (result)
            _pSerialDebug->printf("message Sigfox envoye\r\n");
        else
            _pSerialDebug->printf("erreur envoi message Sigfox\r\n");
    return result;
}
```

setClePublique permet de définir si le modem doit communiquer sur le réseau Sigfox ou le réseau du simulateur.
La commande « ATS410= » 1 ou 0 après le signe égale permet de définir cela.
1 correspondant au réseau simulateur
0 correspondant au réseau sigfox
Ici nous fonctionnons sur le réseau simulateur.

sendMessage permet l'envoie des trames de statistiques.
La commande « AT\$SF= » permet l'envoie de caractère hexadécimal.
On charge à la fin de la commande notre trame.

Classe CStatsVéhicule :

```
void CStatsVéhicule::add(CVéhiculeDetecte *véhicule)
{
    // Ensuite ajout +nbrEssieu à _nbrEssieu
    _nbrEssieu = _nbrEssieu + véhicule->get_nombreEssieu();

    // Calcule vitesse moyenne
    // additionner toute les valeurs /diviser/ par le nombre de valeurs
    _VitesseMoy = ( (_VitesseMoy*_nbrVéhicule) + véhicule->get_vitesse() ) / (_nbrVéhicule + 1);
    //nombre d'itération ;

    // Ensuite ajout +1 à _nbrVéhicule
    _nbrVéhicule = _nbrVéhicule + 1;

    // Calcule vitesse max
    // valeur la plus élevée
    if(_VitesseMax < véhicule->get_vitesse())
    {
        _VitesseMax = véhicule->get_vitesse();
    }

    // Calcule nbrVitesseOK
    // valeur <= vitesseTronc
    if(véhicule->get_vitesse() <= _vitesseTronc)
    {
        _nbrVitesseOK = _nbrVitesseOK + 1;
    }

    // Calcule nbrVitesseLM20
    // vitesseTronc < valeur < vitesseTronc+20
    if(_vitesseTronc < véhicule->get_vitesse() && véhicule->get_vitesse() <= _vitesseTronc + 20)
    {
        _nbrLM20 = _nbrLM20 + 1;
    }

    // Calcule nbrVitesseLP20
    // vitesseTronc + 20 <= valeur < vitesseTronc + 30
    if(_vitesseTronc + 20 < véhicule->get_vitesse() && véhicule->get_vitesse() <= _vitesseTronc + 30)
    {
        _nbrLP20 = _nbrLP20 + 1;
    }

    // Calcule nbrVitesseLP30
    // vitesseTronc + 30 <= valeur < vitesseTronc + 40
    if(_vitesseTronc + 30 <= véhicule->get_vitesse() && véhicule->get_vitesse() < _vitesseTronc + 40)
    {
        _nbrLP30 = _nbrLP30 + 1;
    }

    // Calcule nbrVitesseLP40
    // vitesseTronc + 40 <= valeur < vitesseTronc + 50
```

add : permet d'ajouter le véhicule rentrant, et ensuite de mettre à jour les statistiques

Exemple :

Calcule de la vitesse moyenne :

On prend la (vitesse moyenne * le nombre de véhicule) + la vitesse du véhicule actuel que l'on vient diviser par le nombre de véhicule +1

.

```

if(_vitesseTronc + 40 <= vehicule->get_vitesse() && vehicule->get_vitesse() < _vitesseTronc + 50)
{
    _nbrLP40 = _nbrLP40 + 1;
}

// Calcule nbrVitesseLP50

// vitesseTronc + 50 <= valeur
if(_vitesseTronc + 50 <= vehicule->get_vitesse())
{
    _nbrLP50 = _nbrLP50 + 1;
}

}

void CStatsVehicule::reinit()
{
    _nbrVehicule = 0;
    _nbrEssieu = 0;
    _VitesseMoy = 0;
    _VitesseMax = 0;
    _nbrVitesseOK = 0;
    _nbrLM20 = 0;
    _nbrLP20 = 0;
    _nbrLP30 = 0;
    _nbrLP40 = 0;
    _nbrLP50 = 0;
}

```

reinit() : permet de remettre à zéro chaque variable des statistiques.

Classe TrameStat :

```

void CTrameStatsSigFox::fabriquerTrame(CStatsVehicule *statsV)
{
    _unk.typeStat = statsV->getTypeStat();
    _unk.nbrVehicule = statsV->getNbrVehicule();
    _unk.nbrEssieu = statsV->getNbrEssieu();
    _unk.VitesseMoy = statsV->getVitesseMoy();
    _unk.VitesseMax = statsV->getVitesseMax();
    _unk.nbrVitesseOK = statsV->getNbrVitesseOK();
    _unk.nbrLM20 = statsV->getNbrVitesseLM20();
    _unk.nbrLP20 = statsV->getNbrVitesseLP20();
    _unk.nbrLP30 = statsV->getNbrVitesseLP30();
    _unk.nbrLP40 = statsV->getNbrVitesseLP40();
    _unk.nbrLP50 = statsV->getNbrVitesseLP50();

    unsigned int c;
    unsigned int d;
    unsigned int u;
    unsigned int e;

    _trame[0] = (_unk.typeStat);
    _trame[0] <<=7;
    c = _unk.nbrVehicule & 0x700;
    c >>=4;
    d = _unk.nbrVehicule & 0x0f0;
    d >>=4;
    _trame[0] = _trame[0] + c + d;

    u = _unk.nbrVehicule & 0x00f;
    u <<=4;
    c = _unk.nbrEssieu & 0x01f00;
    c >>=9;
    _trame[1] = u + c;

    c = _unk.nbrEssieu & 0x01ff;
    c >>=1;
    _trame[2] = c;

    u = _unk.nbrEssieu & 0x01;
    u <<=7;
    d = _unk.VitesseMoy & 0xfe;
    d >>=1;
    _trame[3] = u + d;

    u = _unk.VitesseMoy & 0x01;
    u <<=7;
    d = _unk.VitesseMax & 0xff;
    d >>=1;
    _trame[4] = d + u ;
}

```

fabriquerTrame() :

Passage des valeurs de statVéhicule dans la structure.

INFO :

Sigfox ne permet d'envoyer que des trames de 12 octets au maximum (96bits)

Nous réalisons donc un décalage des bits de chaque variable avec l'application d'un filtre pour l'adapter selon un schéma prédéfini ([voir tableau plus bas](#)).

```

u = _unk.VitesseMax & 0x01;
u <<=7;
c = _unk.nbrVitesseOK & 0x7ff;
c >>=4;
_trame[5] = c + u ;

u = _unk.nbrVitesseOK & 0x00f;
u <<=4;
c = _unk.nbrLM20 & 0x780;
c >>=7;
_trame[6] = u + c ;

d = _unk.nbrLM20 & 0x0f0;
d <<=1;
c = _unk.nbrLM20 & 0x00f;
c <<=1;
u = _unk.nbrLP20 & 0x200;
u >>=9;
_trame[7] = d + c + u ;

c = _unk.nbrLP20 & 0x100;
c >>=1;
d = _unk.nbrLP20 & 0x0f0;
d >>=1;
u = _unk.nbrLP20 & 0x00f;
u >>=1;
_trame[8] = d + c + u ;

u = _unk.nbrLP20 & 0x001;
u <<=7;
c = _unk.nbrLP30 & 0x0100;
c >>=2;
d = _unk.nbrLP30 & 0x0f0;
d >>=2;
e = _unk.nbrLP30 & 0x00c;
e >>=2;
_trame[9] = d + c + u + e;

u = _unk.nbrLP30 & 0x003;
u <<=6;
d = _unk.nbrLP40 & 0x07e;
d >>=1;
_trame[10] = d + u ;

u = _unk.nbrLP40 & 0x001;
u <<=7;
d = _unk.nbrLP50 & 0x07f;
_trame[11] = d + u ;

_trame[12] = '\0';

```

```
// Affichage après fabrication
```

```
printf("Trame apres fabrication : ");
for(int i = 0; i < 13 ; i++)
{
    printf("%x ", _trame[i]);
}
printf("\r\n");
```

```
// Partie conversion
```

```
// Passage de l'hexa en code ASCII (pour faire une chaîne de caractère)
for(int i = 0; i < 12; i++){
    _sTrame[i*2] = (_trame[i] >> 4) + 0x30;
    if( (_trame[i] >> 4) >= 0x0A){
        _sTrame[i*2] += 7;
    }
    _sTrame[(i*2)+1] = (_trame[i] & 0x0F) + 0x30;
    if( (_trame[i] & 0x0F) >= 0x0A){
        _sTrame[(i*2)+1] += 7;
    }
}
_sTrame[24] = '\0';

printf("Affichage sTrame : %s \r\n", _sTrame);

_eTrame =(char*)_sTrame; // CAST

printf("Affichage eTrame : %s \r\n", _eTrame);
}
```

Affichage de la trame après la fabrication.

Conversion de la trame en caractère ASCII pour faire une chaîne de caractère. Et qui sera compréhensible pour le modem.

Cast de la variable pour le mettre en char*

Tableau exemple pour une trame :

T	Nb Véhicule	Nb essieux		Vit Moy	Vit Max	Vit OK	Vit LM20			Vit LP20		Vit LP30	Vit LP40	Vit LP50	0	1	2	3	0	1	2	0	1	1	0	0	9
0	3	5	5	0	7	1	0	5	0	7	0	3	0	0	1	2	3	0	1	2	0	1	1	0	0	9	
0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	
0	0	1	1	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	

Explication :

Ligne du haut : Représentation de la taille des champs de bits selon la variable.

Ligne du bas : Représentation de chaque octet.

T = Type de stat : 1 bits

VitLP20 : 10 bits

Nb Véhicule : 11 bits

VitLP30 : 9 bits

Nb essieux : 13 bits

VitLP40 : 7 bits

Vit Moy : 8 bits

VitLP50 : 7 bits

Vit Max : 8 bits

// Un total de 96 bits.

VitOK : 11 bits

VitLM20 : 11 bits

D. Test

Test d'intégration : interclasse

Objectif : Si un véhicule est détecté alors :

- Savoir si c'est un poids lourd ou véhicule léger
- L'ajouter dans CStatsVehicule // vérification avec affichage
- Fabrication de la trame à partir des stats // vérification avec affichage
- Envoyer avec le modem Sigfox // vérification avec affichage et simulateur Sigfox.

```
void CStatsManager::start()
{
    /**
     * Ordre de la procédure :
     * - Initialisation du modem
     * Lorsque Véhicule détecter, ajouter ces caractéristiques dans
     * CStatsVéhicule
     * - Calculer les statistiques par rapport à tous les véhicules légers et poids
     * lourds
     * - Fabrication de la trame des statistiques
     * - Envoyer
     */
    initModem();

    /**
     * TEST
     *
     * _statsPL.setVitesseTronc(50); // Vitesse tronçon à 50 km/h
     * _statsVL.setVitesseTronc(50);

     * CEnumVehicule VL;
     * CEnumVehicule PL;
     * CVehicleDetecte Renault(VL,50,2); // VL
     * CVehicleDetecte Citroen(VL,50,2); // VL
     * CVehicleDetecte Jaguar(VL,50,2); // VL
     * CVehicleDetecte Porsche(VL,50,2); // VL
     * CVehicleDetecte Audi(VL,50,2); // VL
     * CVehicleDetecte Peugeot(VL,50,2); // VL
     *
     * CVehicleDetecte Mercedes(PL,43,3); //PL
     * CVehicleDetecte Scania(PL,45,6); //PL
     * CVehicleDetecte Volvo(PL,46,6); //PL
    */
}
```

Initialisation de plusieurs véhicules

Ensuite vérification si c'est un poids lourd ou véhicule léger.

Exemple :

```

// Lorsque véhicule détecter :
// Savoir si véhicule léger ou poids lourd
if(Renault.get_typeVehicule() == VL){
    _statsVL.add(&Renault);
}
if(Renault.get_typeVehicule() == PL){
    _statsPL.add(&Renault);
}

if(Citroen.get_typeVehicule() == VL){
    _statsVL.add(&Citroen);
}
if(Citroen.get_typeVehicule() == PL){
    _statsPL.add(&Citroen);
}

if(Jaguar.get_typeVehicule() == VL){
    _statsVL.add(&Jaguar);
}
if(Jaguar.get_typeVehicule() == PL){
    _statsPL.add(&Jaguar);
}

if(Porsche.get_typeVehicule() == VL){
    _statsVL.add(&Porsche);
}
if(Porsche.get_typeVehicule() == PL){
    _statsPL.add(&Porsche);
}

if(Audi.get_typeVehicule() == VL){
    _statsVL.add(&Audi);
}
if(Audi.get_typeVehicule() == PL){
    _statsPL.add(&Audi);
}

if(Peugeot.get_typeVehicule() == VL){
    _statsVL.add(&Peugeot);
}
if(Peugeot.get_typeVehicule() == PL){
    _statsPL.add(&Peugeot);
}

```

```
pc.printf("StatVL : \r\n");
pc.printf("%d \r\n", _statsVL.getTypeStat());
pc.printf("%d \r\n", _statsVL.getNbrVehicule());
pc.printf("%d \r\n", _statsVL.getNbrEssieu());
pc.printf("%d \r\n", _statsVL.getVitesseMoy());
pc.printf("%d \r\n", _statsVL.getVitesseMax());
pc.printf("%d \r\n", _statsVL.getNbrVitesseOK());
pc.printf("%d \r\n", _statsVL.getNbrVitesseLM20());
pc.printf("%d \r\n", _statsVL.getNbrVitesseLP20());
pc.printf("%d \r\n", _statsVL.getNbrVitesseLP30());
pc.printf("%d \r\n", _statsVL.getNbrVitesseLP40());
pc.printf("%d \r\n", _statsVL.getNbrVitesseLP50());
```

Affichage des statistiques
// Pour vérifier qu'elle soit bien calculer avec la méthode add()

Fabrication des trames

Envoie de la trame VL

```
/*
if (&vehicule.get_typeVehicule == VL){
    _statsVL.add(&vehicule);
}
if (&vehicule.get_typeVehicule == PL){
    _statsPL.add(&vehicule);
}
*/
_trameStatsVL.fabriquerTrame(&_statsVL);
_trameStatsPL.fabriquerTrame(&_statsPL);

_modemSigFox.sendMessage(_trameStatsVL.getTrame());
}
```

E. Partie physique

Modèle : BP7X

Capacité : 1820 mAh

Tension : 3.7 V

Technologie : Lithium-Ion



Avantages ? :

- Léger / Peu encombrant
- Densité énergétique élevée :

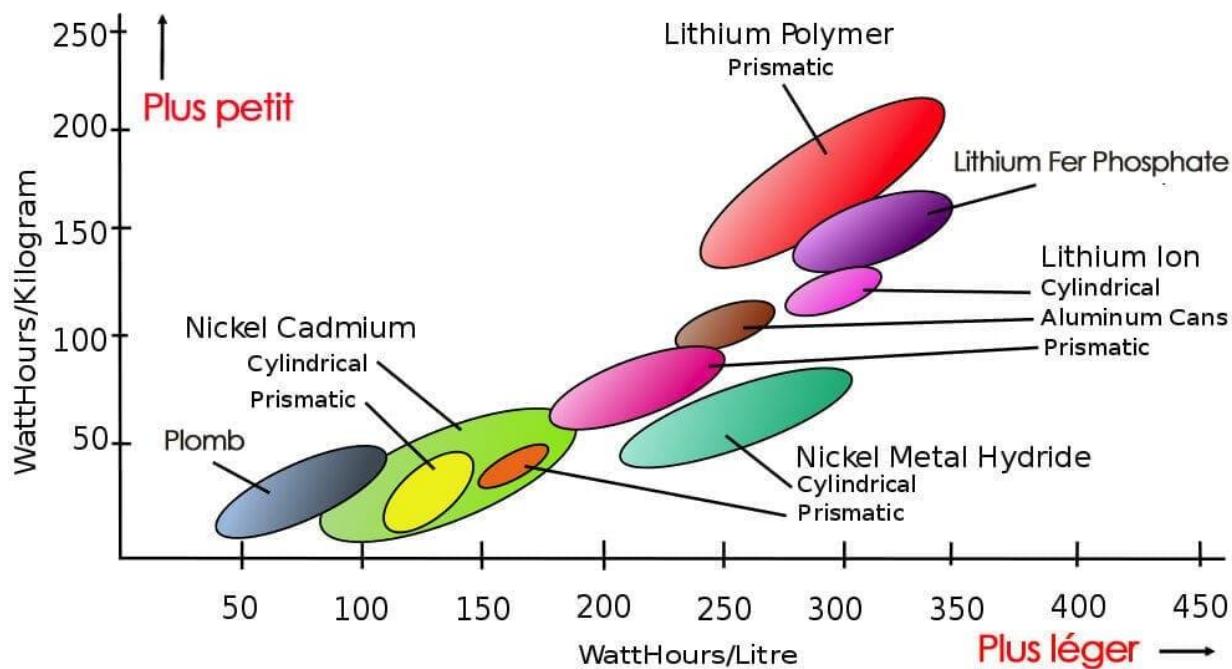
Lithium-Ion : 120 – 140 Wh/kg

Plomb : 30 – 50 Wh/kg

- Nécessite peu de maintenance

Inconvénients ? :

- Ne pas laisser la batterie se décharger de trop.



F. Bilan technique :

Tâche réalisé :

- Fabrication de la trame
- Communication avec le modem Sigfox
- Envoie de la trame
- Intégrations avec le sous-système consultation

Tâche restante :

- Classe de contrôle de la batterie et des trames de la batterie
- Classe des statistiques des essieux
- Intégrations totales du sous-système collecte

IV. Etudiant Testa Tony

A. Description personnelle

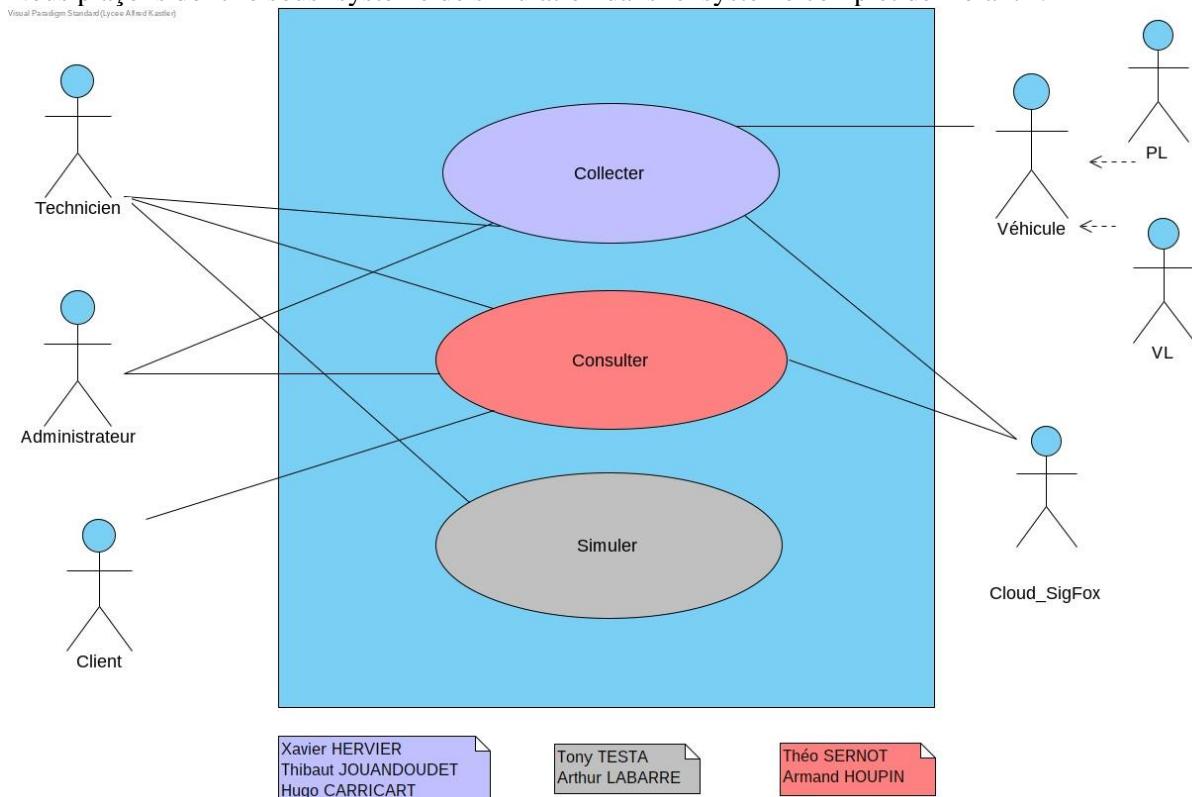
Situation (sous-système)

Le système KolantR se doit d'être opérationnel au cours d'une campagne de mesure, il faut au technicien, lors de la procédure d'installation d'un boîtier de mesure, un moyen d'être certain que les différents sous-systèmes, Collecte et Consultation, fonctionnent correctement, c'est-à-dire, pour le sous-système de Collecte, qu'il mesure des données cohérentes et pour le sous-système de consultation, que les données soient bien les mêmes que celles acquises et qu'elle soit consultable à distance.

Le moyen le plus simple de s'assurer que le système KolantR est fonctionnel serait de simuler le passage d'un ou plusieurs véhicules pendant la procédure d'installation (Avant le placement des tuyaux sur la route) afin de vérifier le bon fonctionnement de chaque sous-système.

C'est pour répondre à ce besoin que le sous-système de simulation a été ajouté au projet, nous verrons ici donc en quoi il consiste et comment il fonctionne.

Nous plaçons donc le sous-système de simulation dans le système complet de KolantR.



Le sous-système de simulation se décompose en deux parties :

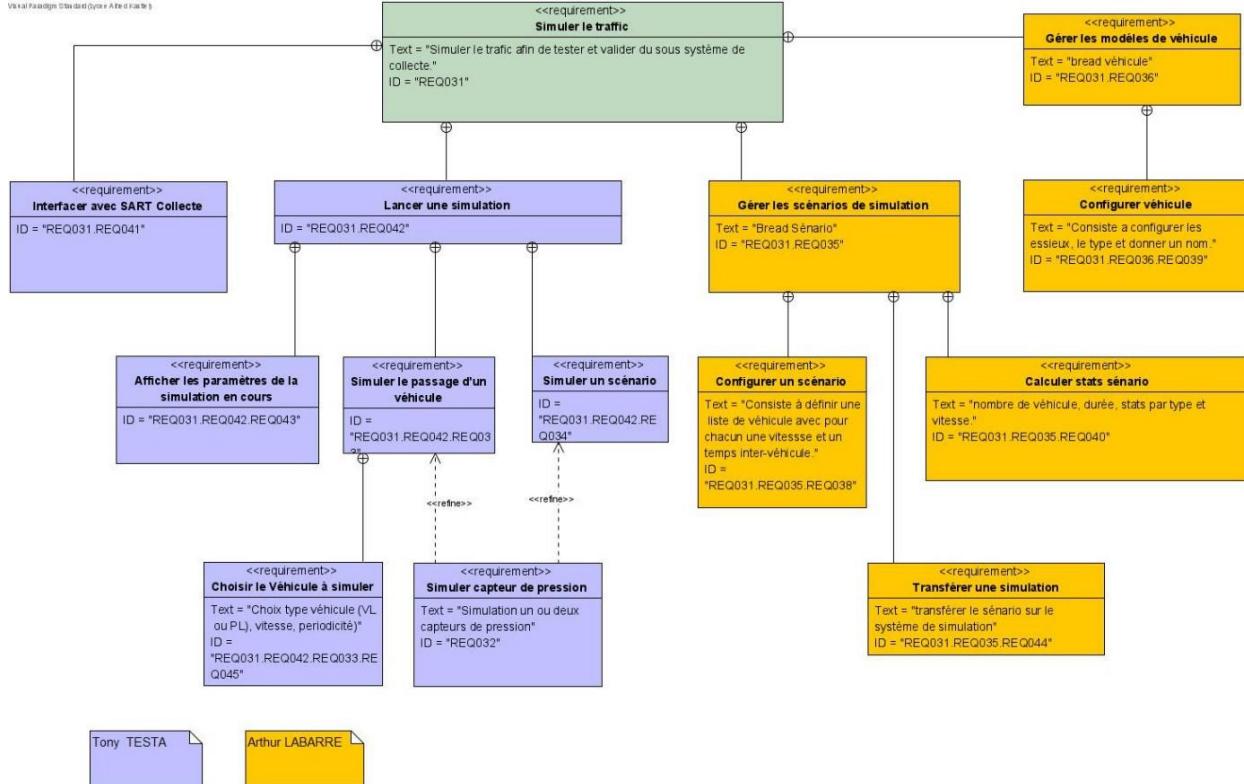
-La partie embarquée (matérielle, qui se branche sur le sous-système de collecte et produit les signaux)

-La partie logicielle (une application PC qui permet de gérer et créer des profils de véhicules ainsi que des scénarios (trafic routier) afin de l'envoyer sur la carte pour qu'elle simule ce scénario)

Nous nous attarderons ici sur la partie embarquée du sous-système de simulation.

B. Conception détaillée

Les exigences pour ce sous-système sont les suivantes :



Nous nous concentrerons ici sur la partie embarquée soit sur les exigences suivantes :

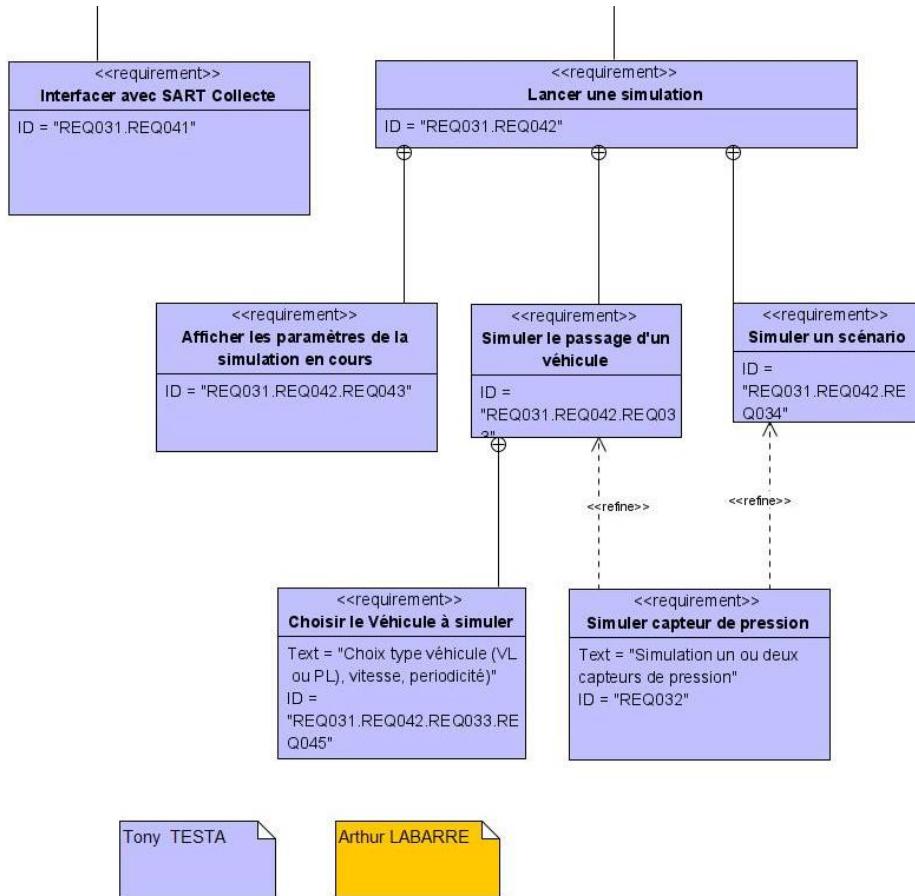
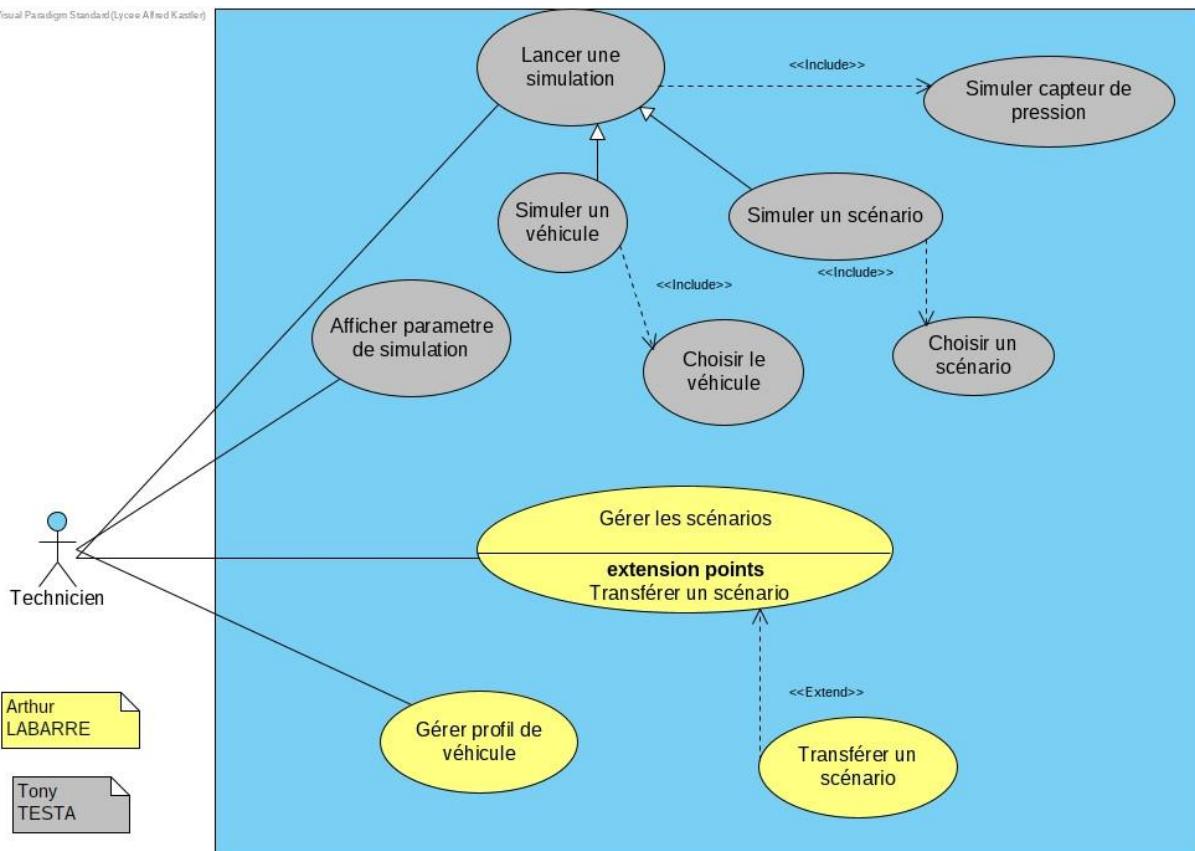


Diagramme de Cas d'utilisation de la partie



Description des cas d'utilisation de la partie embarquée du sous-système :

Acteur (1) : Technicien

Choisir Le Véhicule

Steps	Procedures	Expected Results
1. Choisir véhicule	Utiliser IHM pour Choisir un véhicule (VL, PL)	Véhicule correctement sélectionné et paramètres du véhicule corrects
2. Choisir type (vl ou pl)	naviguer dans le menu à l'aide des boutons	le type choisi s'affiche
3. choisir cenari	à l'aide des boutons, incrémenter ou décrémenter la vitesse du véhicule	la vitesse varie selon le choix de l'utilisateur
4. choisir fréquence de passage	incrémenter ou décrémenter la fréquence à l'aide des boutons	La fréquence varie selon le choix de l'utilisateur

Afficher Paramètres de Simulation

Steps	Procedures	Expected Results
1. Afficher type de Simulation	affiché automatiquement	type affiché
2. afficher nombre de véhicule	affiché automatiquement	nombre affiché
3. afficher type de véhicule	affiché automatiquement	type affiché

Steps	Procedures	Expected Results
4. afficher frequence de passage	affiché automatiquement	frequence affichée

Lancer une simulation

Steps	Procedures	Expected Results
1. Afficher choix type de scénario	Afficher automatiquement	affiche « Véhicule » et « recevoir un scenario »
2. choisir un type simulation	Appuyer sur les boutons pour naviguer dans le menu	un curseur indique le choix actuel
3. Lancer simulation	Choisir “lancer”	La simulation demarre conformément au type de simulation souhaitée

Choisir scénario

Steps	Procedures	Expected Results
1. Un ecran « en attente de scénario s'affiche »	affiché automatiquement	“en attente affiché”
2. Scénario reçus, l'utilisateur peut lancer la simulation	naviguer avec les boutons pour lancer la simulation	simulation lancée

Simuler Capteur de pression

Steps	Procedures	Expected Results
1. Génération d'un signal électrique sur une sortie numérique (simulation d'une tension)	Lancer un scenario/simulation	la simulation est effectuée (système détecte un véhicule)

Le diagramme de conception, pour cette partie du sous-système est le diagramme de classe de cette dernière.

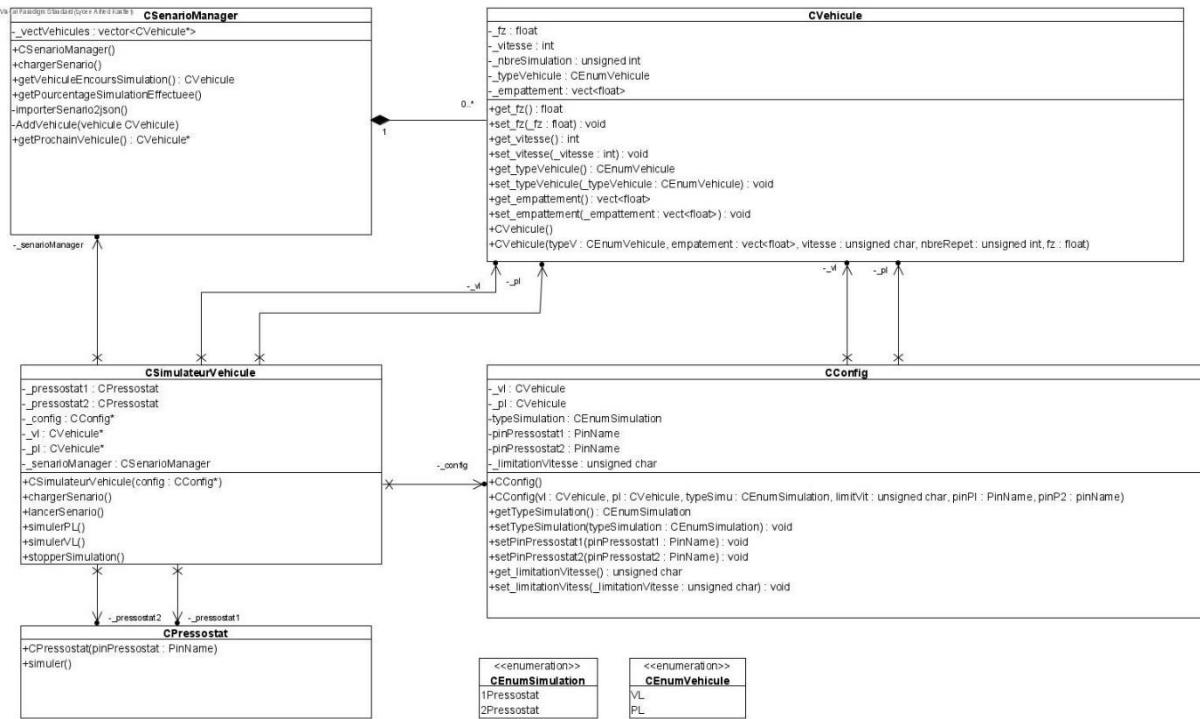
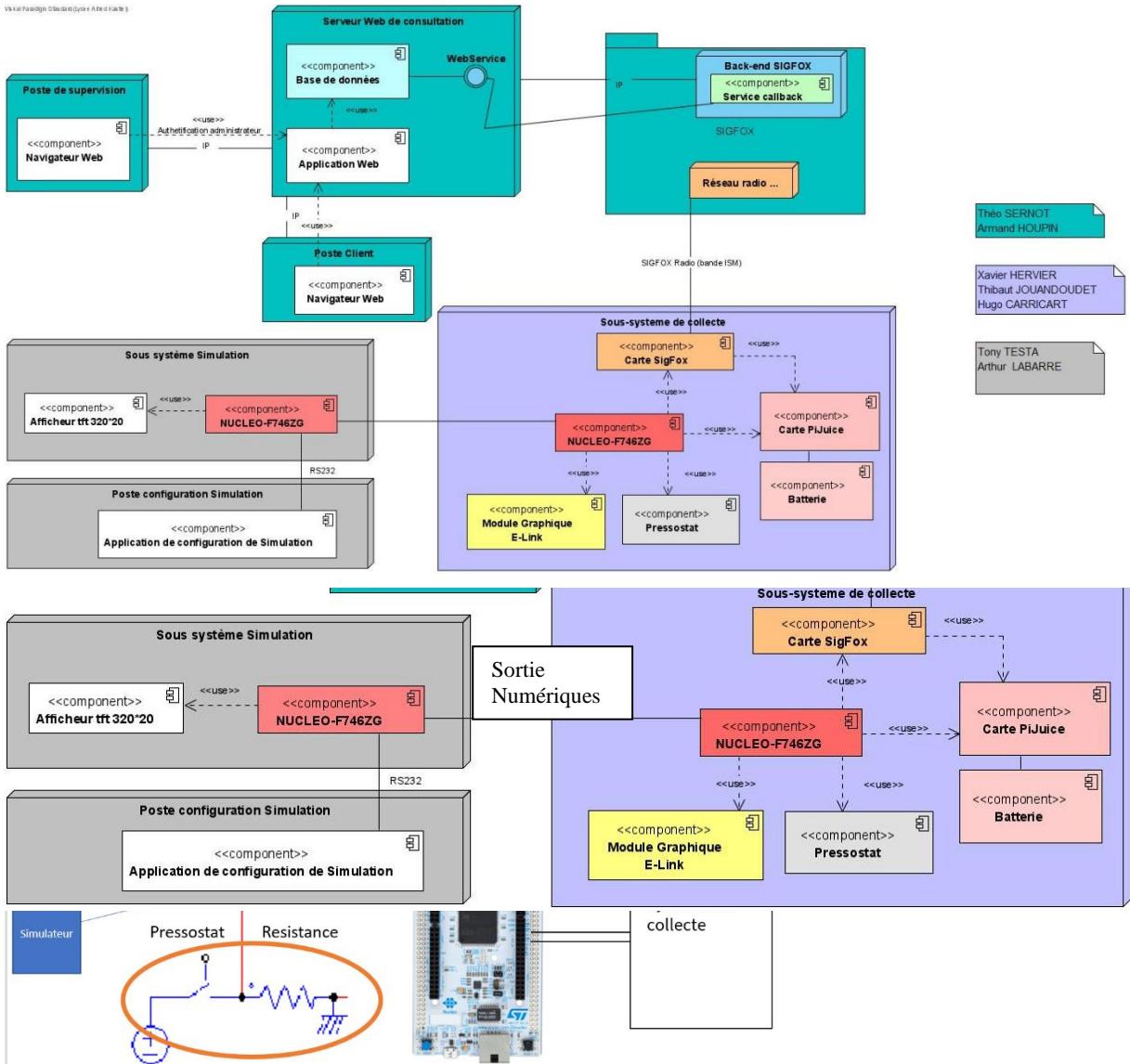


Diagramme de déploiement



C. Implémentation

Mise en œuvre des techniques

Mise en œuvre de la communication avec l'application

La communication avec l'application PC du sous-système de simulation se fera via une liaison série RS232



Afin de pouvoir recevoir des données, il faut faire en sorte que la carte Mbed puisse utiliser ce genre de communication.

```
//nombre d'elements max que peut prendre le buffer
#define MAXIMUM_BUFFER_SIZE 1024

//sortie numerique led, change d'état à chaque caractère reçu
static DigitalOut led(LED1);

// créer un bufferedserial à la vitesse par défaut
static BufferedSerial serial_port(D1, D0);
```

On peut créer un buffer, ici d'une taille maximale de 1024 caractères il recevra les caractères pour ensuite les entrer dans un tableau., on crée une sortie led qui utilise la LED1, la led utilisateur de la carte

Ensuite, on définit grâce à la ligne static BufferedSerial serial_port(D1, D0) ; les ports sur lesquels la communication série se fera, ici D1 et D0.

serial_port.set_baud(9600); Ici, on définit la vitesse de la communication série en bauds, on choisit 9600 bauds qui est la fréquence par défaut utilisée par l'application.

```
char buf[MAXIMUM_BUFFER_SIZE] = {0};
bool cont = true;
int i = 0;
char tramerecue[2048];
```

On initialise un entier i à 0, on crée un tableau de 2048 « cases » nommé tramerecue.

```

while (cont){
    if (uint32_t num = serial_port.read(buf, sizeof(buf))
        if (buf[0] == 0x04){
            cont = false;
            tramerecue[i] = 0x00;
        }
        else {
            tramerecue[i] = buf[0];
            i = i+1;
        }
        // Toggle the LED.
        led = !led;

        // Echo the input back to the terminal.
        serial_port.write(buf, num);
}

```

Cette boucle while permet de récupérer les caractères envoyés, plus précisément :

if (buf[0] == 0x04){ Ici on teste si le caractère vaut le code ASCII 0x04, si c'est le cas, la réception s'arrête, ce code ASCII correspond donc au caractère d'arrêt.

```

else {
    tramerecue[i] = buf[0];
    i = i+1;
}
// Toggle the LED.
led = !led;

// Echo the input back to the terminal.
serial_port.write(buf, num);

```

Si le caractère reçu ne correspond pas au caractère d'arrêt, le caractère reçu, prend la place dans le tableau qui correspond à la valeur de i.

tramerecue[i] = buf[0];

Suite à cela, on incrémente i

```

tramerecue[i] = buf[0];
i = i+1;

```

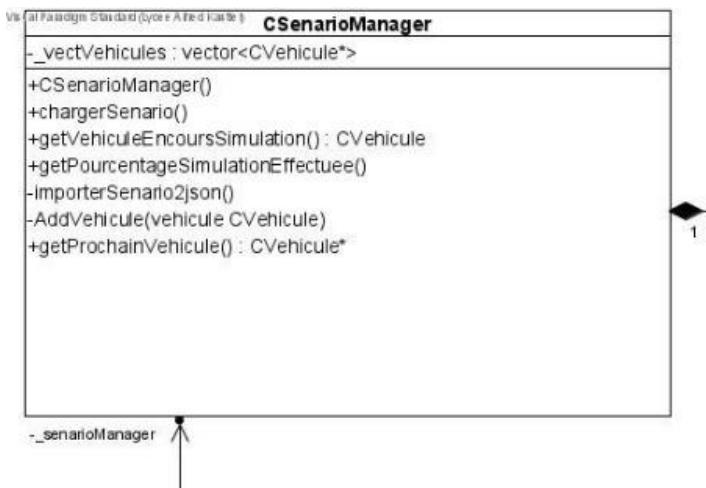
Et on passe la led à l'état contraire de celui où elle était, ce qui fait qu'elle change d'état à chaque caractère reçu.

```

tramerecue[i] = buf[0];
i = i+1;
}
// Toggle the LED.
led = !led;

```

Tous les caractères ont donc été reçus et sont enregistrés dans le tableau tramerecue.
Cette partie du programme s'intègre dans la classe CSenarioManager



Interprétation des caractères reçus

Les données sont transmises, mais elles doivent être exploitables pour la carte, il faut donc choisir un format et le format choisi est le JSON.

Pour interpréter le JSON, nous pouvons le faire également dans la classe CScenarioManager (La méthode ImporterScénario2json())

Après avoir reçu le scénario et l'avoir stocké dans le tableau trammerecue, il est possible de l'interpréter grâce à cet extrait de code

```

puts( str );
json_t mem[32];
json_t const* json = json_create( str, mem, sizeof mem / sizeof *mem );
if ( !json ) {
    puts("Error json create.");
    return EXIT_FAILURE;
}

json_t const* nomScenario = json_getProperty( json, "nomScenario" );
if ( !nomScenario || JSON_TEXT != json_getType( nomScenario ) ) {
    puts("Error, the first name property is not found.");
    return EXIT_FAILURE;
}
char const* nomScenarioVal = json_getValue( nomScenario );
printf( "Nom Scenario: %s.\n", nomScenarioVal );

char const* NbreEssieux = json_getPropertyValue( json, "NbreEssieux" );
if ( !NbreEssieux ) {
    puts("Error, the last name property is not found.");
    return EXIT_FAILURE;
}
printf( "Nombre d'essieux: %s.\n", NbreEssieux );

```

Dans cet extrait de code la variable nommée « str » correspond à la trame reçue

```
puts( str );
```

Grace à la librairie ./tiny-json.h, certaines méthodes en rapport avec l'exploitation de trames en JSON sont utilisables.

```
json_t mem[32];
json_t const* json = json_create( str, mem, sizeof mem / sizeof *mem );
if ( !json ) {
    puts("Error json create.");
    return EXIT_FAILURE;
}
```

Ici, nous créons un fichier temporaire JSON qui contient les données de la trame, le format de la trame ne correspond pas à du JSON (condition du « if ») on a une erreur, sinon les données sont bien en JSON et pourront donc être exploitées par la suite.

Exemple de trame en JSON

```
{
    "nomScenario": "Scenario_3",
    "passages": [
        {
            "emplacement": [
                3.2
            ],
            "nbreEssieux": 2,
            "tempsInterVehicule": 10,
            "typeVehicule": "vl",
            "vitesseVehicule": 67
        },
        {
            "emplacement": [
                5.2
            ],
            "nbreEssieux": 2,
            "tempsInterVehicule": 15,
            "typeVehicule": "vl",
            "vitesseVehicule": 59
        },
    ],
}
```

```
json_t const* nomScenario = json_getProperty( json, "nomScenario" );
if ( !nomScenario || JSON_TEXT != json_getType( nomScenario ) ) {
    puts("Error, the first name property is not found.");
    return EXIT_FAILURE;
}
```

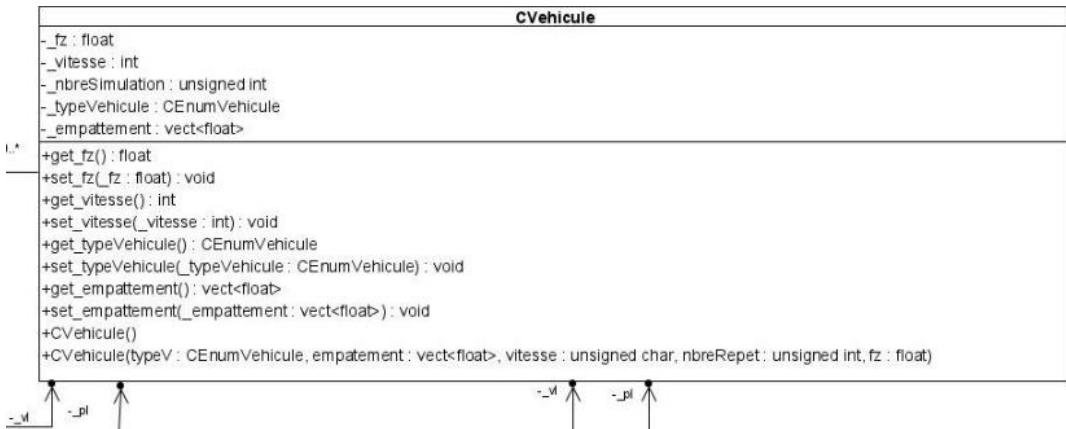
Ici, on sélectionne le champ nommé « nomScenario » pour en récupérer les données, dans cette trame « Scenario_3 »

Avec cet exemple, nous pouvons effectuer la même opération pour tous les champs afin d'en extraire les données.

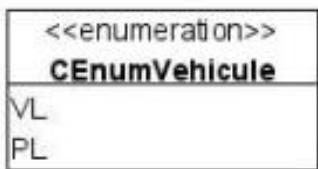
Paramétriser un véhicule

Chaque véhicule est différent (emplacement, vitesse, nombre d'essieux), il faut donc pouvoir les paramétrer, dans cette partie, nous nous attarderons sur la façon dont les paramètres des véhicules sont entrés dans le programme.

Les véhicules sont des objets qui seront instanciés grâce à la classe CVehicule



Nous pouvons voir dans cette classe les différents paramètres d'un véhicule comme l'empattement, le nombre de fois qu'il passe ainsi que son type qui est défini grâce à la classe d'énumération CEnumVehicule :



Pour commencer, nous pouvons instancier un objet de type CVehicle qui dans cet exemple se nomme « multipla »

```

CVehicle multipla;
CConfig Sim;

multipla.set_vitesse(130.00);
multipla.set_empattement(2.44);

```

Suite à cela, nous pouvons utiliser les méthodes de la classe CVehicle pour entrer des paramètres comme la vitesse avec NomObjet.méthodeSet() ; ici multipla.set_vitesse(130.00) ;

Ses paramètres deviennent exploitables, nous pouvons aussi créer un objet de la classe CConfig qui est comme ceci :

*	*
CConfig	
-_vl : CVehicule	
-_pl : CVehicule	
-typeSimulation : CEnumSimulation	
-pinPressostat1 : PinName	
-pinPressostat2 : PinName	
-_limitationVitesse : unsigned char	
+CConfig()	
+CConfig(vl : CVehicule, pl : CVehicule, typeSimu : CEnumSimulation, limitVit : unsigned char, pinPl : PinName, pinP2 : pinName)	
+getTypeSimulation() : CEnumSimulation	
+setTypeSimulation(typeSimulation : CEnumSimulation) : void	
+setPinPressostat1(pinPressostat1 : PinName) : void	
+setPinPressostat2(pinPressostat2 : PinName) : void	
+get_limitationVitesse() : unsigned char	
+set_limitationVitesse(_limitationVitesse : unsigned char) : void	

La classe CConfig elle sert à définir les paramètres de la simulation comme les « pinPressostat » qui sont les ports des sorties numériques utilisées pour générer les signaux électriques ou encore la limitation de vitesse.

```
Sim.setTypeSimulation(CEnumSimulation::pressostat2);

Sim.set_limitationVitesse(30); //Milletipla.set_type()
multipla.afficherParam();
```

Ici, la simulation donne une limitation de vitesse de 30 km/h et grâce à la classe CEnumSimulation, nous pouvons définir le type de simulation (si on simule un seul ou deux pressostats) la classe CEnumSimulation se présente comme ceci :

<<enumeration>>
CEnumSimulation
1Pressostat
2Pressostat

Création des signaux de simulation

Le principal intérêt du sous-système de simulation est de produire les mêmes signaux électriques que ceux que produisent les véhicules qui passent sur les tuyaux reliés aux pressostats afin de vérifier le bon fonctionnement du système de collecte.

Pour avoir les bons signaux, il faut calculer les temps de passage, temps de passage sur le tuyau, temps pour parcourir un mètre, temps que met le véhicule pour parcourir la distance correspondante à son empattement.

Ces paramètres sont dans la classe CConfig

```

7
8     /*-----*/  

9  

10    vector<double> emp;  

11    emp.push_back(3.00);  

12    _vl.set_empattement(emp);  

13    _vl.set_vitesse(100.0);  

14    _vl.set_typeVehicule(CEnumVehicule::VL);  

15    _pl.set_vitesse(90);  

16    _pl.set_typeVehicule(CEnumVehicule::PL);  

17    emp.clear();  

18    emp.push_back(3.85);  

19    emp.push_back(4.10);  

20    emp.push_back(1.3);  

21    emp.push_back(1.3);  

22    _pl.set_empattement(emp);
23 }

```

Suite à cela, dans la classe CSimulateurVehicule, nous pouvons récupérer les paramètres de la simulation.

```

9
0  CSimulateurVehicule::CSimulateurVehicule(CConfig *config){  

1      printf("création de la sim\n\r");  

2      _config = config;  

3  }

```

Et donc les paramètres des véhicules également.

Suite à cela nous pouvons donc calculer les valeurs de vitesses et de temps

```

9
0      double vitessems = 0.0;
1      double timeP = 3600.0;
2      double tuyau = 0.01;
3
4      double vitkmh = _config->get_vl()->get_vitesse();
5      //calcul de la vitesse en m/ms
6      vitessems = _config->get_vl()->get_vitesse() / timeP;
7
8      double tps1m = 1.0/vitessems;
9
10     double tpsT = 0.1*tps1m;
11
12     double tpsEmpattement = _config->get_vl()->get_empattement()[0]*tps1m;
13     printf("temps en empattement = %lf\n\r", tpsEmpattement);
14

```

Pour ensuite mettre nos sorties numériques 1 et 2 à l'état haut et bas de la manière correspondant à une acquisition de signal électrique

```

DigitalOut Tube1 = D8;
DigitalOut Tube2 = D9;

```

```

ThisThread::sleep_for(1000);
Tube1 = 1;
Tube2 = 0;
ThisThread::sleep_for(tpstuyau);
Tube1 = 0;
ThisThread::sleep_for(tps1metre-tpstuyau);
Tube2 = 1;
ThisThread::sleep_for(tpstuyau);
Tube2 = 0;
ThisThread::sleep_for(temp);
Tube1 = 1;
ThisThread::sleep_for(tpstuyau);
Tube1 = 0;
ThisThread::sleep_for(tps1metre-tpstuyau);
Tube2 = 1;
ThisThread::sleep_for(tpstuyau);
Tube2 = 0;

```

Plutôt que d'utiliser un wait_ms(), nous utilisons la méthode sleep_for() pour « mettre en pause » le thread, en faisant comme cela, nous pouvons faire les pauses entre chaque changements d'états en fonctions des données du véhicules et de cette manière cela est fait en « économisant » du temps processeur.

D. Tests

Liaison Série RS232

Pour tester la réception des trames en JSON, on peut observer grâce à des « breakpoints » et le débugger la valeur de tramerecue

Le scénario à recevoir est le suivant :

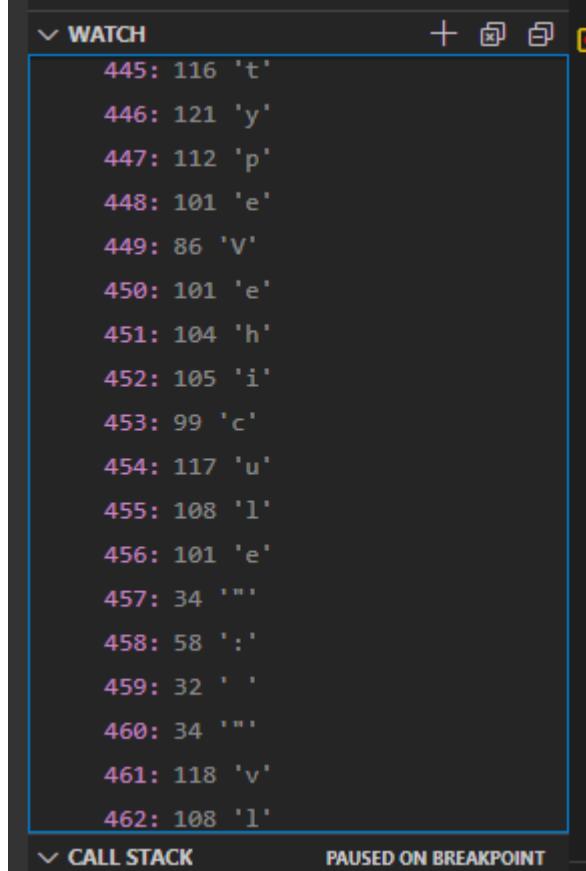
```
{
  "nomScenario": "Scenario_3",
  "passages": [
    {
      "emplacement": [
        3.2
      ],
      "nbreEssieux": 2,
      "tempsInterVehicule": 10,
      "typeVehicule": "vl",
      "vitesseVehicule": 67
    },
    {
      "emplacement": [
        5.2
      ],
      "nbreEssieux": 2,
      "tempsInterVehicule": 15,
      "typeVehicule": "vl",
      "vitesseVehicule": 59
    }
  ]
}
```

```

},
{
  "emplacement": [
    2.6
  ],
  "nbreEssieux": 2,
  "tempsInterVehicule": 11,
  "typeVehicule": "vl",
  "vitesseVehicule": 55
}

```

La trame est reçue et nous pouvons observer un champ en particulier celui coloré dans la trame.



Nous pouvons voir que cette fonctionnalité est opérationnelle car le scénario est bien « copié » dans la variable tramerecue

Interprétation JSON

Pour tester cette fonctionnalité nous allons renvoyer une trame simple puis vérifier que cela fonctionne en sélectionnant certains champs et en affichant les valeurs de ces champs.

```
{
    "NomVehicule": "Lamborghini URUS",
    "TypeVehicule": "VL",
    "NbreEssieux": 2,
    "Passage": {
        "Empattement": "3.00 m",
    }
}
```

Trame reçue

Exploitation de cette trame

```
{
    "NomVehicule": "Lamborghini URUS",
    "TypeVehicule": "VL",
    "NbreEssieux": 2,
    "Passage": {
        "Empattement": "3.00 m",
    }
}

nom du véhicule: Lamborghini URUS.
Type du véhicule: VL.
Nombre d'essieux: 2.
```

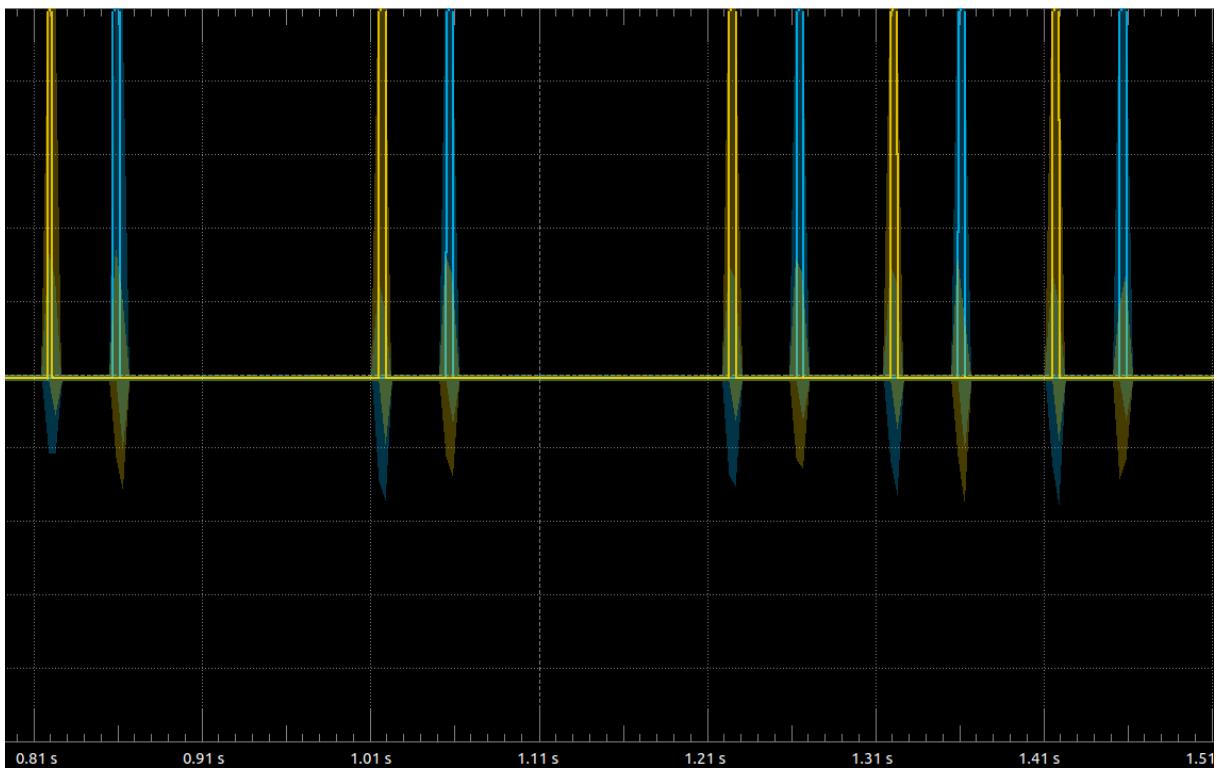
Génération des signaux électriques

Les signaux électriques, doivent être générés pour pouvoir imiter le passage d'un véhicule nous avons vu précédemment que l'utilisation d'une attente passive pour créer toutes les « pauses » entre chaque changements d'états était préférable, ce signal doit donc être cohérent avec les données que peuvent fournir un véhicule réel lors du passage.

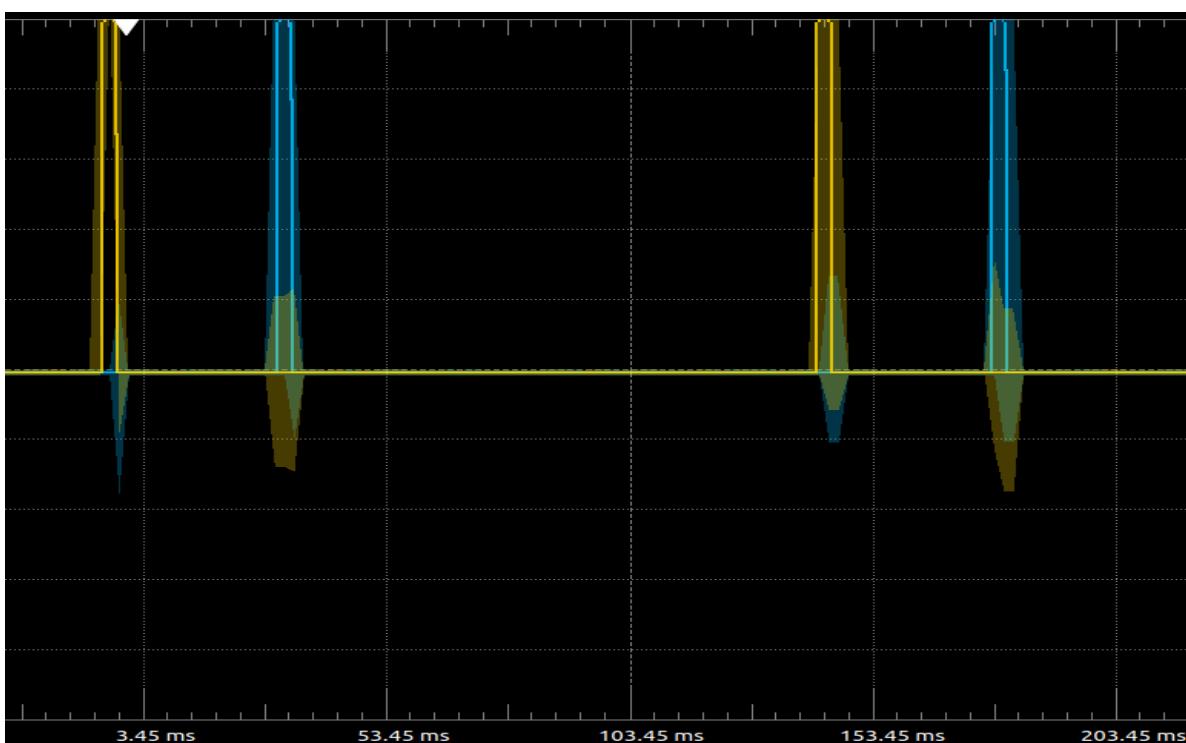
Test pour le passage d'un poids lourd à 5 essieux à une vitesse de 90 km/h d'empattement 3.85m, 4.1m puis 1.3m entre chaque roues de remorques (3)

Sa vitesse en m/s = 25 m/s il mettra 40 ms pour parcourir un metre le temps qu'il mettra à parcourir l'empattement entre le deuxième essieu du tracteur et le premier essieu de la remorque sera de 164 ms

Démonstration avec l'échelle de temps sur le chronogramme suivant :

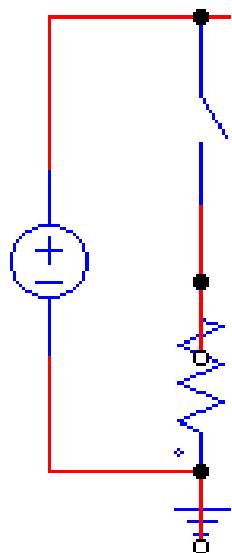
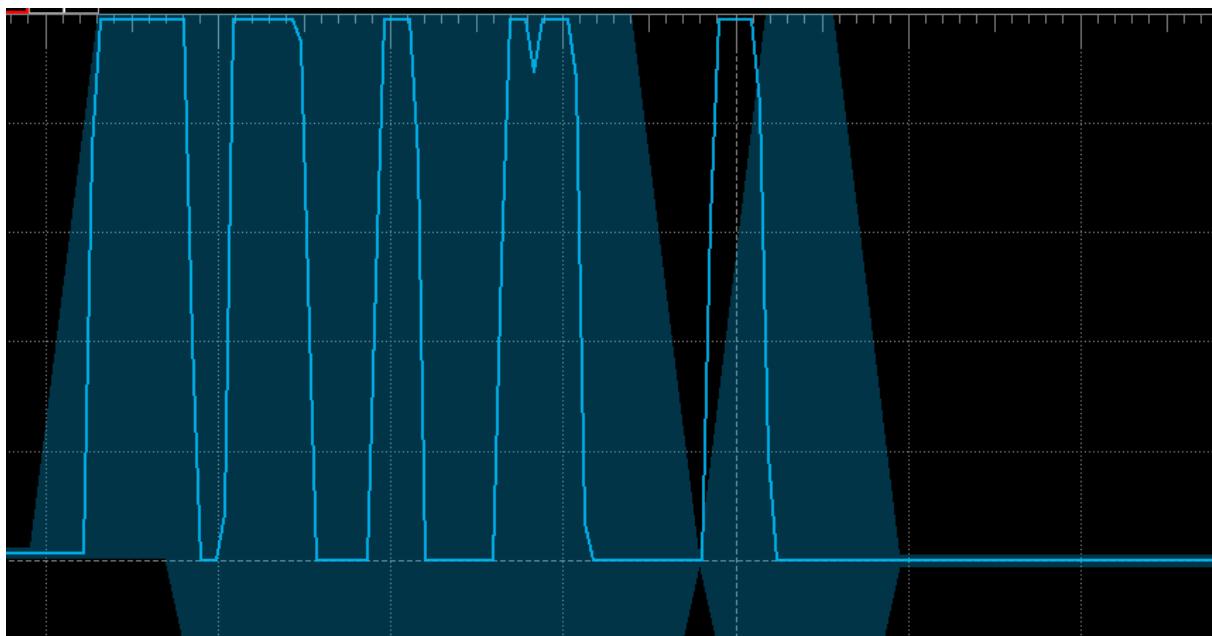


Nous pouvons faire exactement les mêmes calculs et la même chose avec une voiture d'empattement de 2.44m roulant à une vitesse de 100km/h

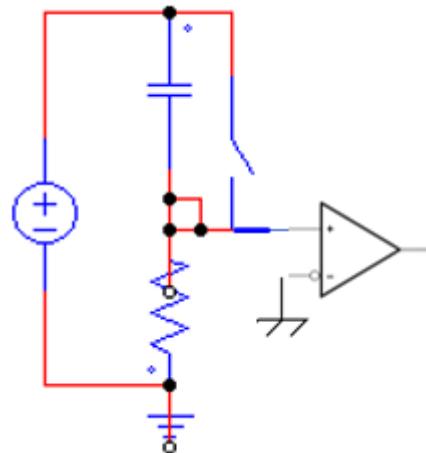


E. Etude Physique

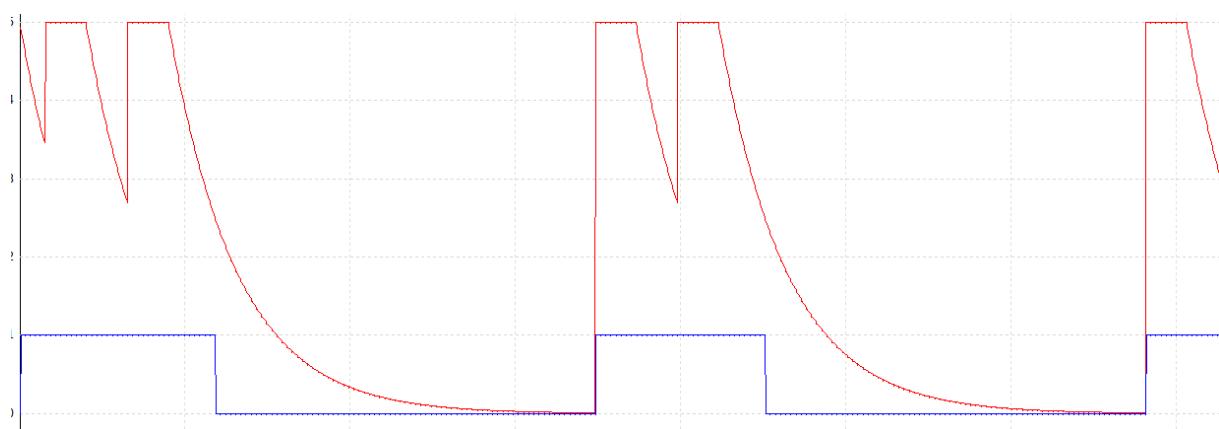
Le problème du système réel est l'émission de rebonds et donc de parasites électriques par le capteur



Pour résoudre ce problème, la mise en place d'un circuit RC est nécessaire pour filtrer les tensions pour donner ceci



Ce circuit RC est composé d'une résistance de 220 KOhms et d'un condensateur de 127 nF puis d'un comparateur de tensions pour avoir une tension de seuil ou lorsque la tension y est inférieure, cela vaut un état bas, et lorsque la tension est supérieure à la tension de seuil, alors cela vaut un état haut.



Dans cet exemple, nous avons la tension filtrée (en rouge) et l'état logique (haut ou bas) en bleu.

V. Etudiant Arthur LABARRE

A. Description partie personnelle

Sous système de simulation sert à simuler un trafic routier afin de tester et valider le sous-système de collecte.

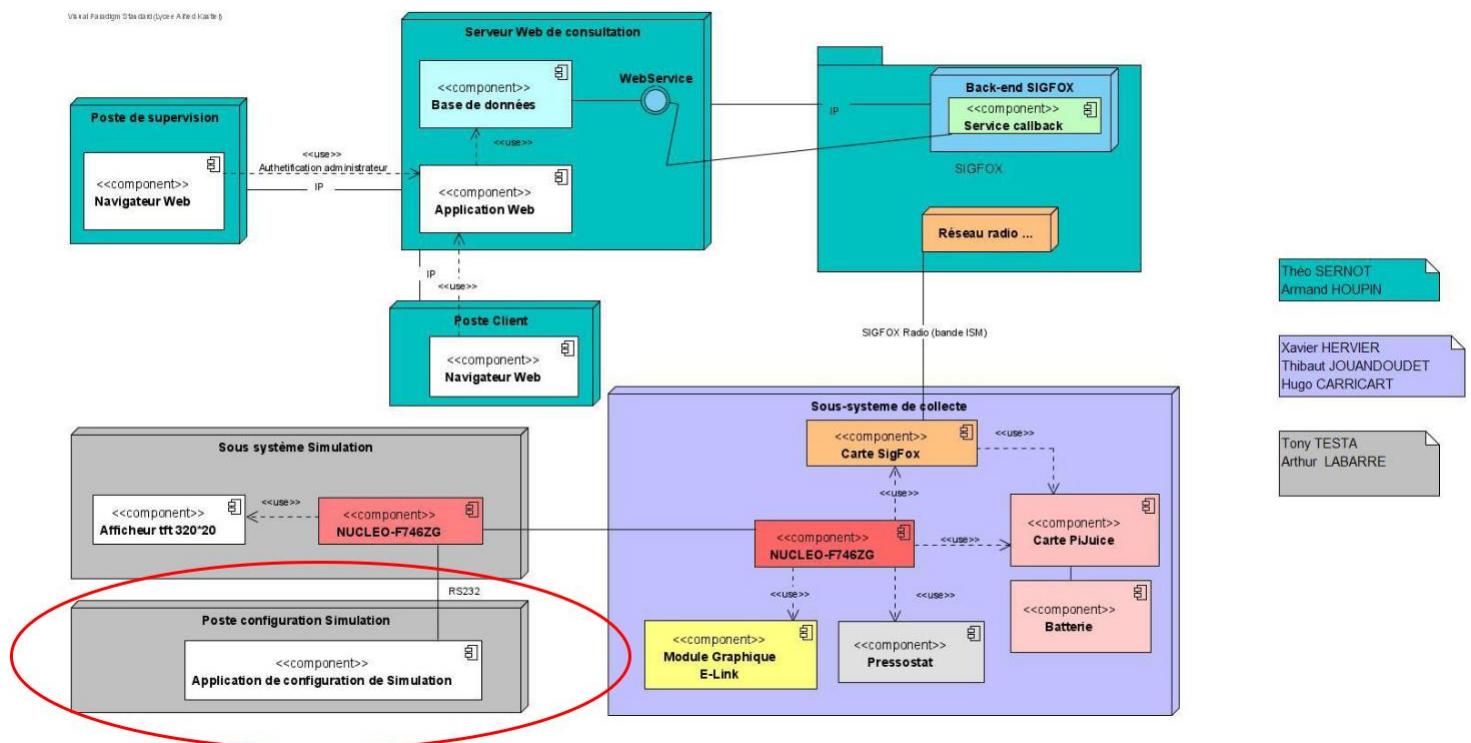
But : Développer une application « Système de simulation trafic » qui vise à configurer un trafic routier afin de connaître ses statistiques.

Les trafics routiers sont appelés scénario. Cette application permet de créer différents scénarios de passages de véhicules. Elle interagit sur une base de données ayant 3 tables : les scénarios, les passages et les véhicules.

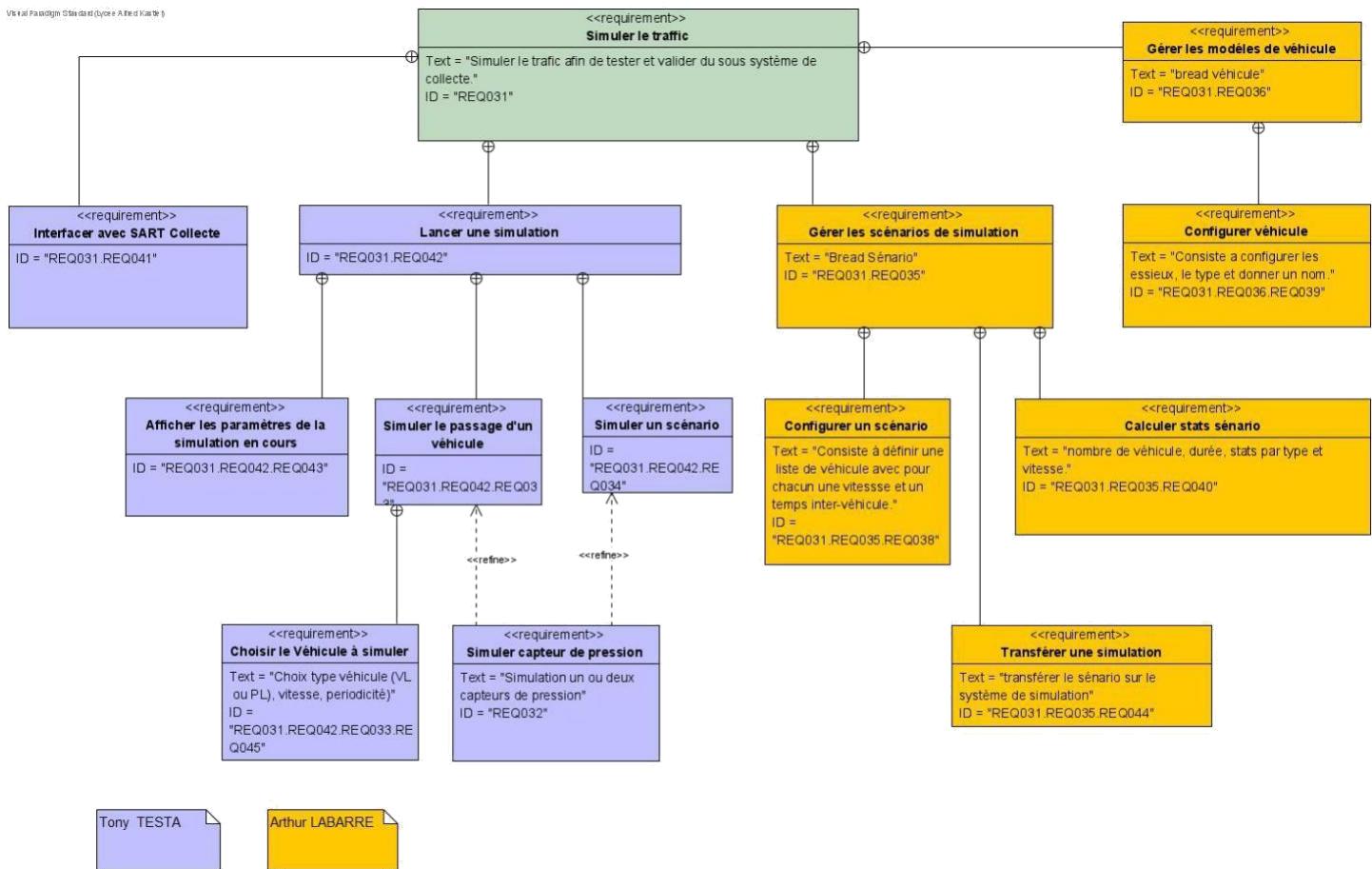
Les scénarios portent un nom. Les passages ont une vitesse et un temps véhiculaire. Les véhicules possèdent un nom, un type (poids-lourd ou véhicules léger), un nombre d'essieux et la taille d'empattement par essieux. Un véhicule est affecté à un passage et, les passages sont affectés à un scénario.

B. Conception détaillée

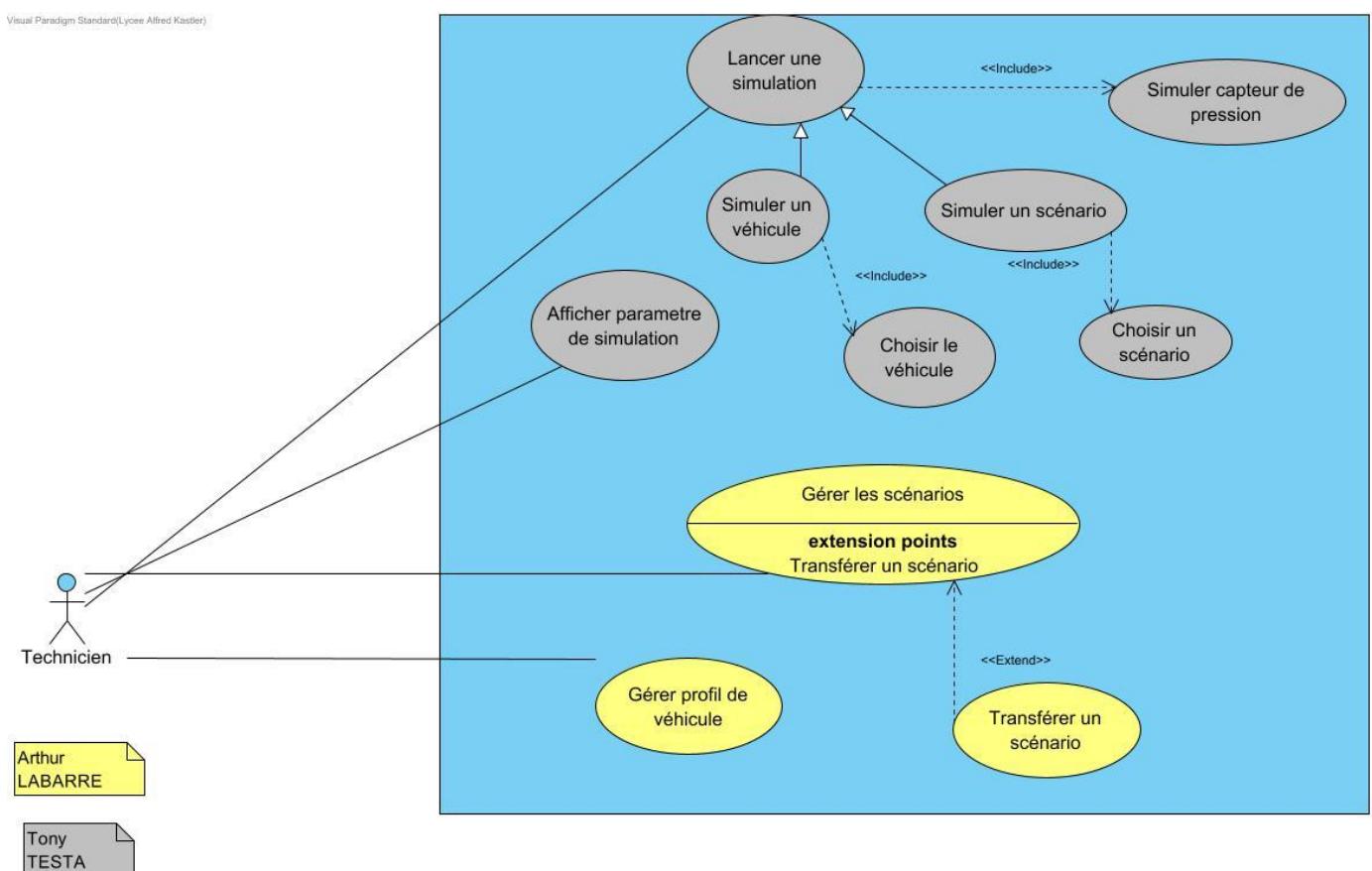
1) Diagramme de déploiement



2) Diagramme d'exigence



3) Cas utilisation



Scenario (Main)

Steps	Procedures	Expected Results
1. Choisir un scenario sur le logiciel ou le créer, le valider		

Scenario (choisir scenario app desktop)

Steps	Procedures	Expected Results
1. Choisir un scenario sur le logiciel ou le créer, le valider	Lancer le logiciel et choisir/créer son scénario	Le scénario est enregistré et prêt à l'envoi

➊ 1. Gérer les scénarios

ID: UC53

Consiste à définir une liste de véhicule avec pour chacun une vitesse et un temps inter-véhicule. Un scénario permet de simuler le trafic de véhicule.

➋ 2. Gérer les véhicules

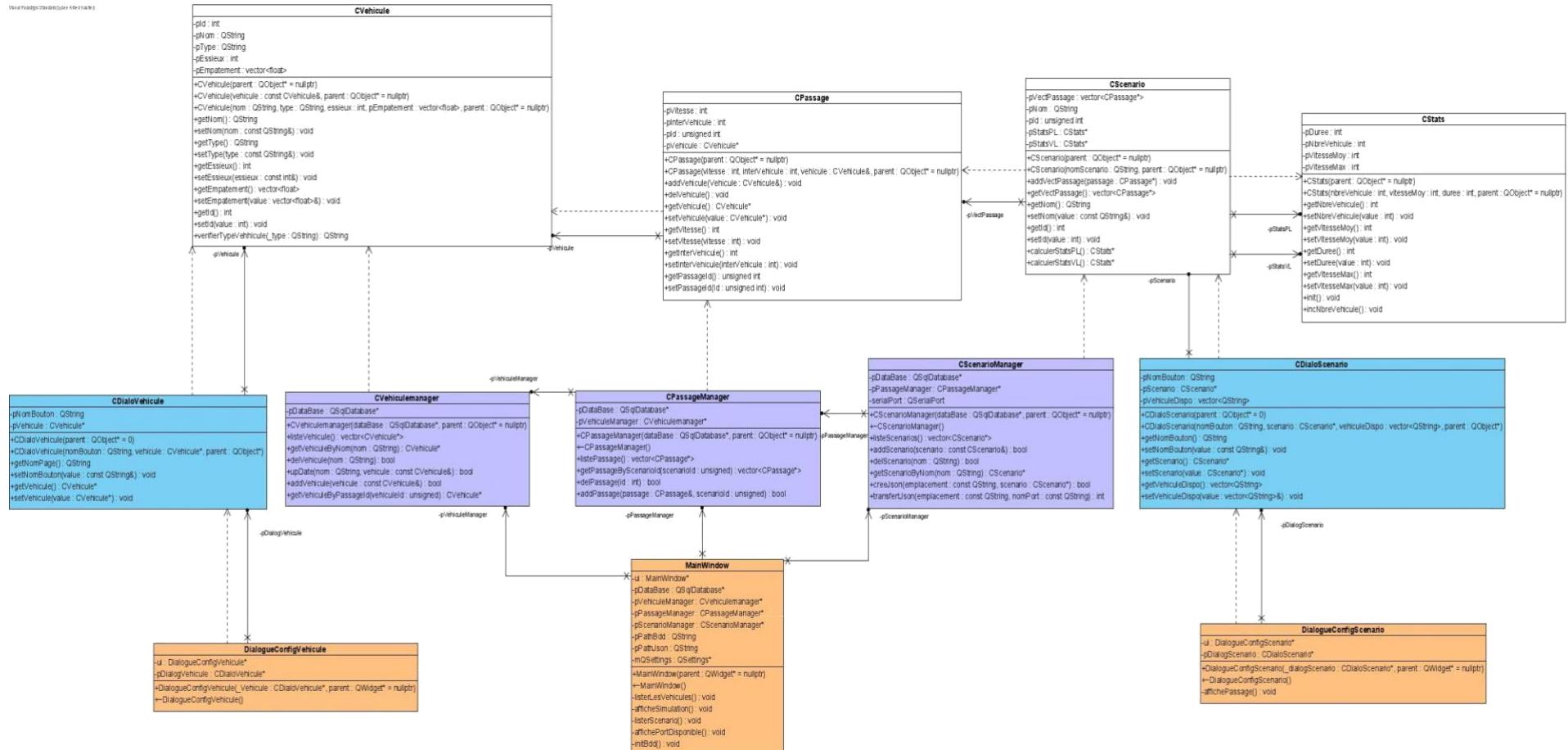
Consiste à configurer le nombre d'essieu, l'empattement entre essieux et le nom du véhicule

➌ 3. Transférer un scénario

ID: UC61

Transférer le scénario sur la partie embarqué du sous-système de simulation

4) Diagramme de classe de conception



Classes conteneurs

Classe permettant d'interagir avec l'extérieur
(base de données, JSON, liaison série)

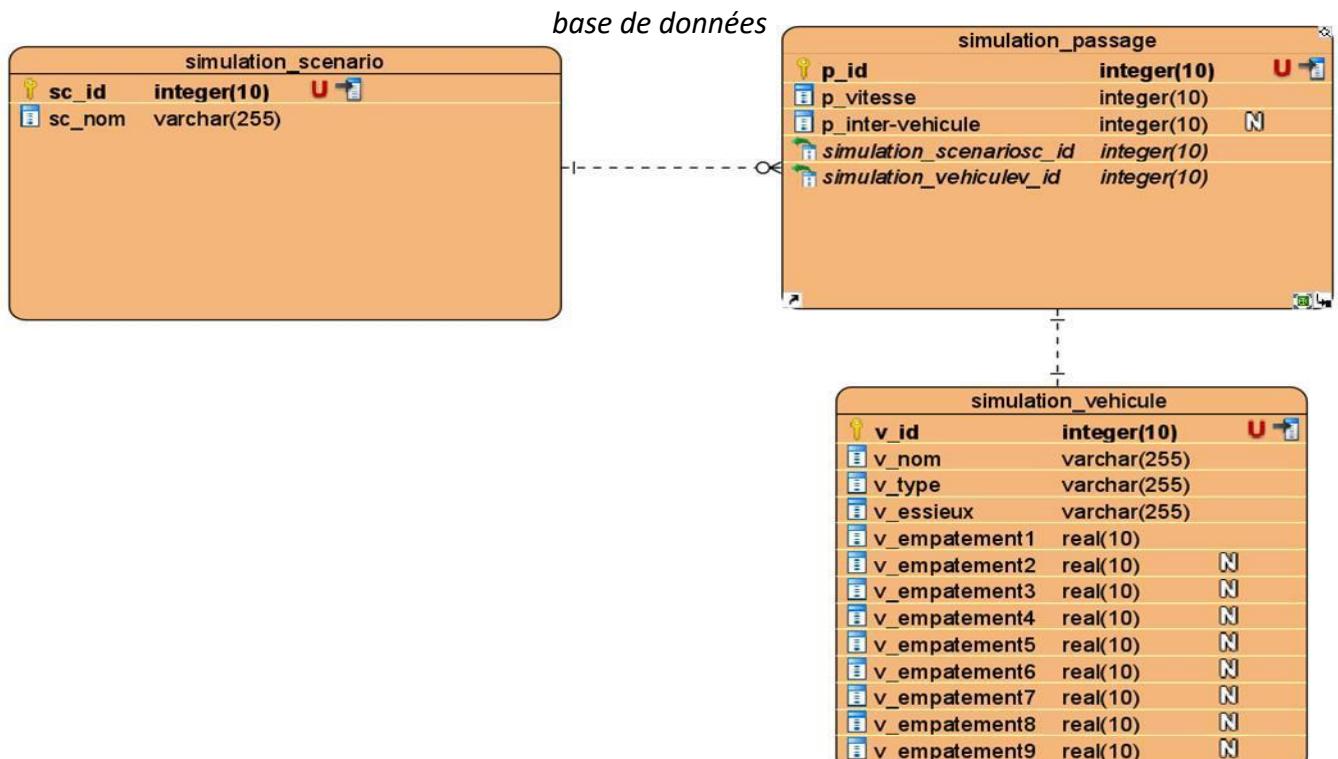
Classes permettant de remplir les interfaces
hommes-machines

Classes conteneurs pour les boîtes de
dialogues

Nom de la classe	Description
CVehicule	Classe conteneur des véhicules
CPassage	Classe conteneur des passages
CScenario	Classe conteneur des scenarios, et remplit classe conteneur cstats
CStats	Classe conteneur des stats
CVehiculeManager	Classe permettant d'interroger ou d'envoyer des requêtes dans la base de données en rapport avec les véhicules
CPassageManager	Classe permettant d'interroger ou d'envoyer des requêtes dans la base de données en rapport avec les passages
CScenarioManager	Classe permettant d'interroger ou d'envoyer des requêtes dans la base de données en rapport avec les scénarios, de remplir le JSON et l'envoyer en liaison série
CDialoVehicule	Classe conteneur pour l'IHM de dialogueConfigVehicule
CDialoScenario	Classe conteneur pour l'IHM dialogueConfigScenario
DialogueConfigVehicule	Classe pour la boîte de dialogue « créer ou modifier un véhicule »
DialogueConfigScenario	Classe pour la boîte de dialogue « créer ou modifier un scénario »
MainWindows	Classe permettant de simuler un trafic et connaître ces statistiques pour le projet KOLANTR

C. Implémentation

A partir de l'application, l'utilisateur interagie avec la base de données pour créer, modifier ou supprimer des véhicules, des passages et des scénarios. Ci-dessous, le Schéma entités-relations :



Créer une application sous Qt Creator

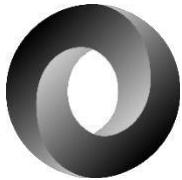


Qt est un Framework de développement multi-plateforme.

Qt Designer est un module de Qt Creator qui permet de créer des IHM via une interface.

– Les fichiers d'interface graphique sont formatés en XML et portent l'extension .ui.

– Lors de la compilation, un fichier d'interface graphique est converti en classe C++ par l'utilitaire uic.



Remplir un fichier JSON

JSON : JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML



Envoyer des données en liaison série

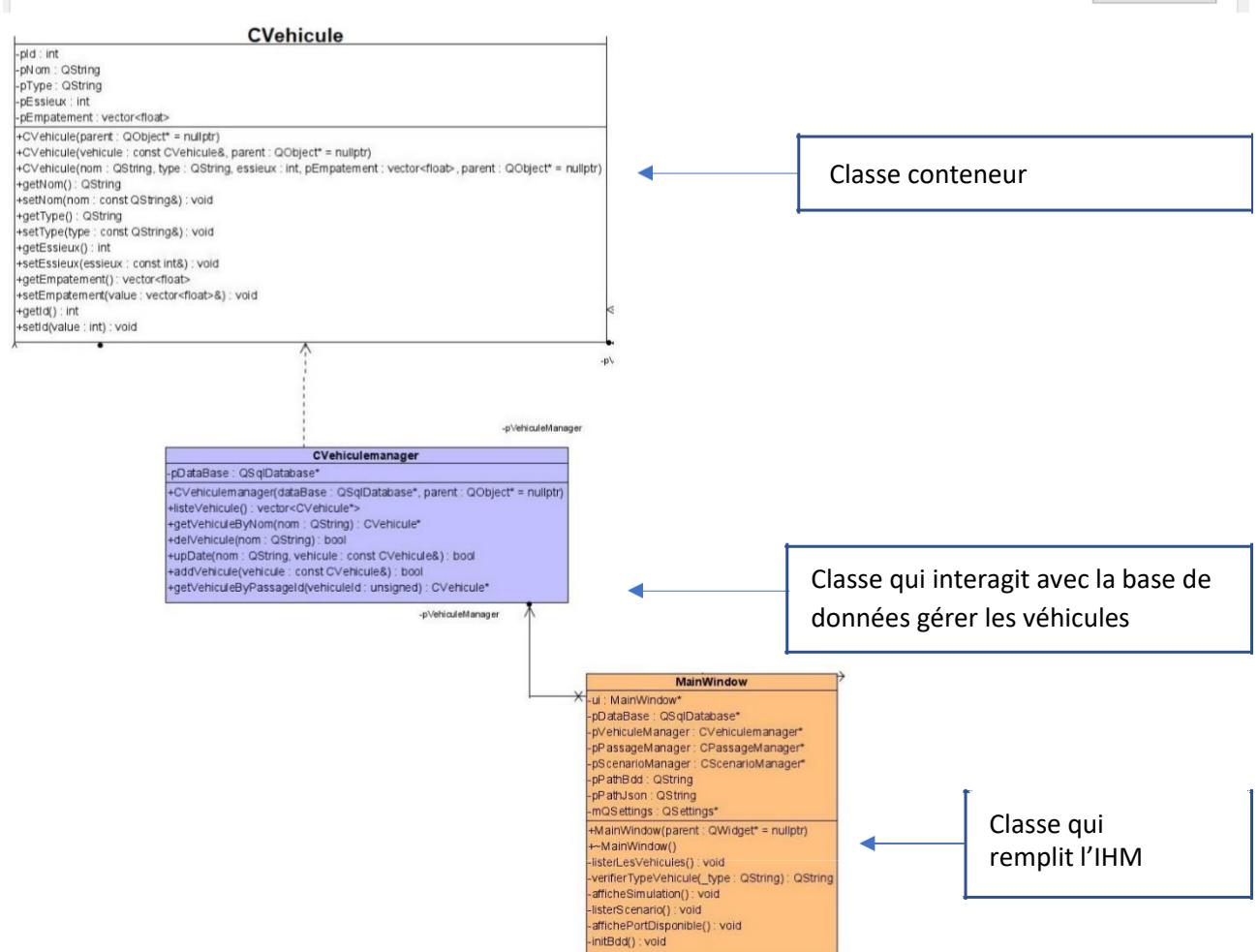
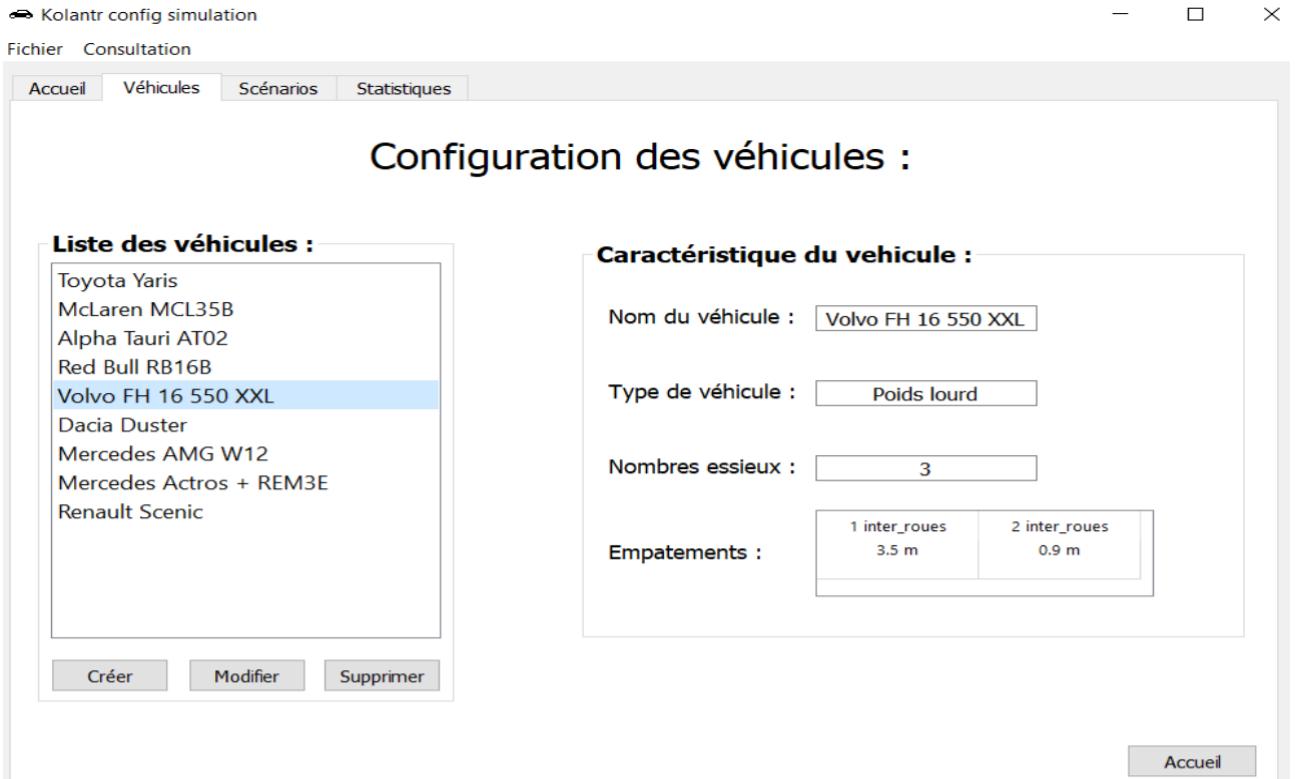
La communication avec l'application PC du sous-système de simulation se fera via une liaison série RS232

L'application développée pour ce projet permet de créer différents véhicules et des scénarios de passages de véhicules. Les scénarios sont transférés en liaison série vers la partie embarquée du sous-système de simulation. Elle a 4 onglets :

Le premier onglet est la page d'accueil permettant de diriger vers les autres onglets.

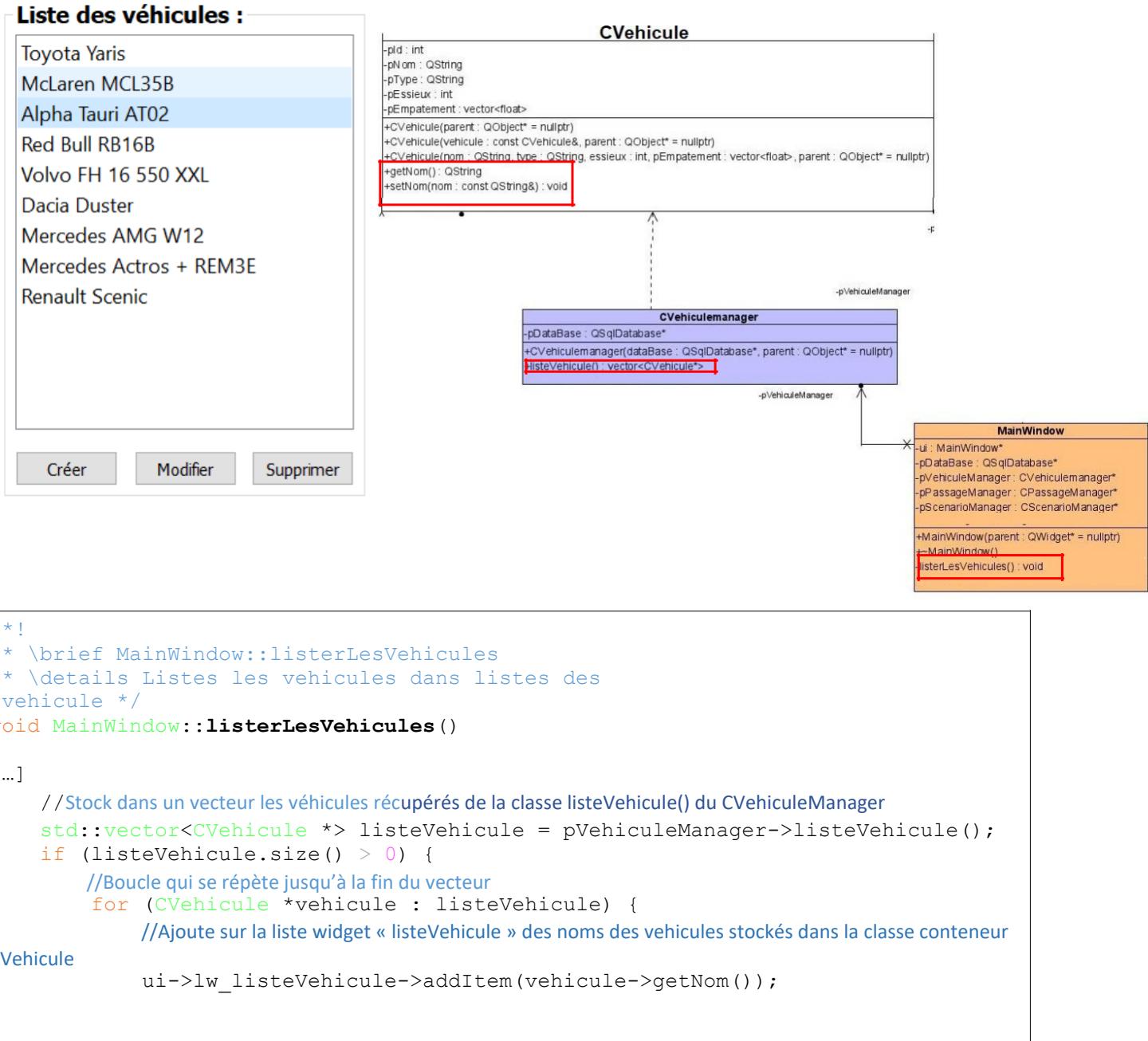


Le deuxième onglet est consacré pour la configuration des véhicules.



Partie du diagramme de conception pour les configurations des véhicules

Exemple pour lister les véhicules dans la liste widget « listeVehicule »:



Extrait code **listerLesVehicules()** de la classe MainWindow

```

/*!
 * \brief CVehiculemanager::listeVehicule
 * \return cvehicule
 * \details Listes des objets
 CVehicules */
std::vector<CVehicule*> CVehiculemanager::listeVehicule()
{
    [...]
    QSqlQuery *sqlQuery = new QSqlQuery(pDataBase->database());
    //prépare la requete pour réquerer les valeurs des vehicules dans la base de
    données
    sqlQuery->prepare("SELECT * FROM
simulation_vehicule"); sqlQuery->exec();

    CVehicule *vehicule = nullptr;
    while (sqlQuery->next()) {
        //Création d'un objet vehicule
        vehicule = new CVehicule();
        //envoi du nom du vehicule récupérer de la base dans la classe conteneur
        CVehicule
        vehicule->setNom(sqlQuery-
        >value("v_nom").toString()); //Ajout de l'objet crée
        dans vectBadges vectVehicule.push_back(vehicule);
    }
}
return vectVehicule;

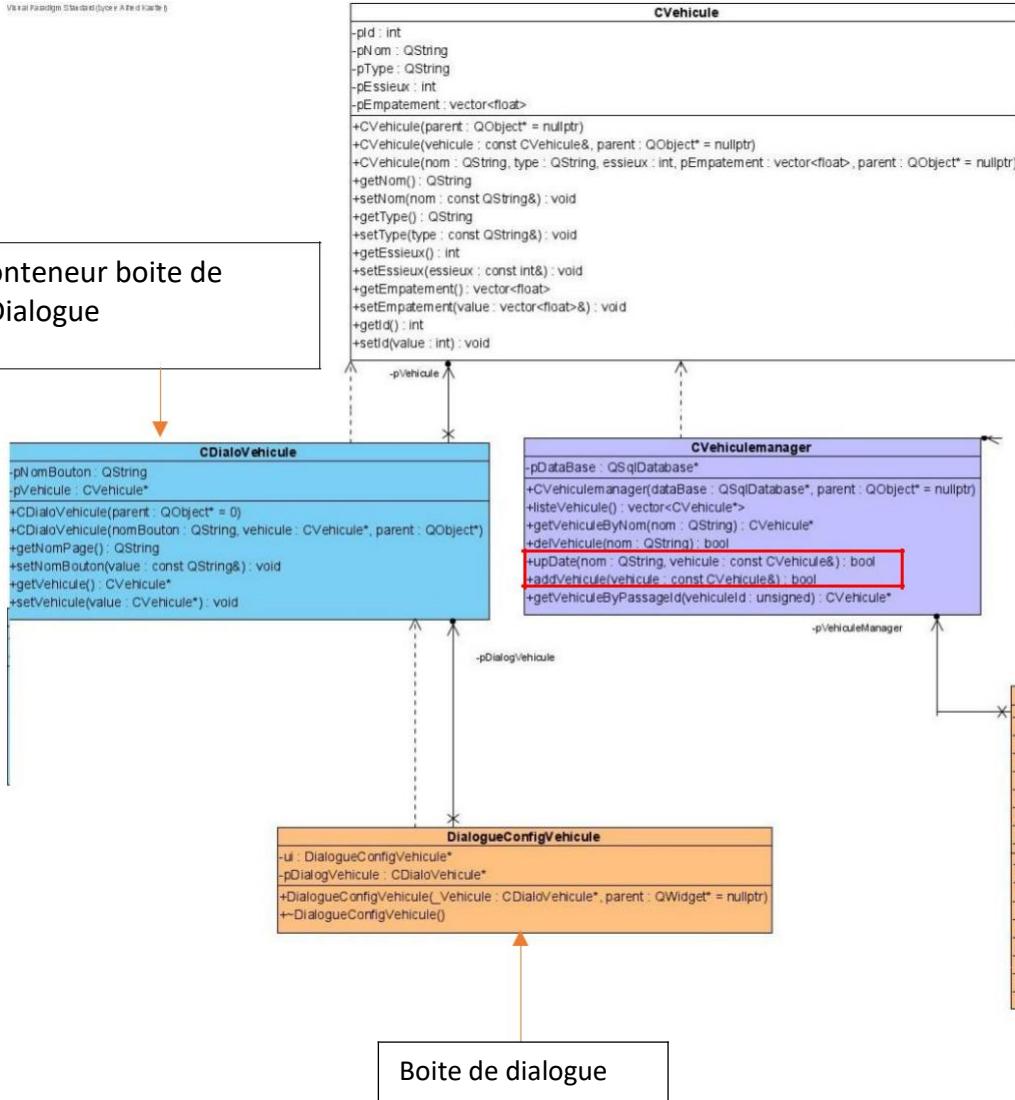
```

Extrait code listeScenarios() de la classe CVehiculemanager

Boites de dialogues pour créer ou modifier un véhicule.

The image shows two dialog boxes side-by-side. Both dialogs have a title bar with a car icon and the text 'Créer un véhicule' or 'Modifier véhicule : Toyota Yaris'. The left dialog is titled 'Créer un véhicule' and contains fields for 'Nom véhicule' (with input 'Vehicule_'), 'Type de véhicule' (dropdown menu showing 'Véhicule léger'), 'Nombres essieux' (spin box showing '2'), and 'Empatements (m)' (table with rows '1' and '2.6'). The right dialog is titled 'Modifier véhicule : Toyota Yaris' and contains identical fields: 'Nom véhicule' (input 'Toyota Yaris'), 'Type de véhicule' (dropdown menu showing 'Véhicule léger'), 'Nombres essieux' (spin box showing '2'), and 'Empatements (m)' (table with rows '1' and '5.2'). Both dialogs have 'Annuler' and 'Valider' buttons at the bottom.

Conteneur boite de Dialogue



```


/*
 * \brief MainWindow::on_pb_creerVehicule_released
 * \details avec le bouton, ouvre la fenêtre dialogue vehicule pour creer le vehicule
 */
void MainWindow::on_pb_creerVehicule_released()
{
    CDialoVehicule *vehiculeDialog = new CDialoVehicule(this);
    vehiculeDialog->setNomBouton(ui->pb_creerVehicule->text());
    //crée l'object vehicule
    CVehicule *vehicule = new CVehicule(this);
    vehiculeDialog->setVehicule(vehicule);
    DialogueConfigVehicule *dialogueConfVehicule = new DialogueConfigVehicule(vehiculeDialog, this);
    //Lance la boite de dialogue
    int ret = dialogueConfVehicule->exec();

    if (ret == QDialog::Accepted) {
        //récupération tous les parametre du vehicule_dialog a la methode
        CVehicule CVehicule *newVehicule = vehiculeDialog->getVehicule();
        //ajout dans la bdd
        pVehiculeManager->addVehicule(*newVehicule);
        //maj list widget
        listerLesVehicules();
        delete newVehicule;
    }
}
  

```

Extrait code `on_pb_creerVehicule_released()` de la classe `MainWindow`

```

/*!
 * \brief DialogueConfigVehicule::on_pb_valider_released
 * \details pb validation et envoie des paramètres du véhicule à la classe CDialogueVehicule
 */
void DialogueConfigVehicule::on_pb_valider_released()
{
    CVehicule *vehicule = new CVehicule(this);
    vehicule->setNom(ui->le_nomVehicule->text());

    //maj diminutif vl et pl
    if (ui->cb_typeVehicule->currentText() == "Véhicule léger")

        vehicule->setType("vl");
    else
        vehicule->setType("pl");

    vehicule->setEssieux(ui->sp_nombreEssieux-
    >value()); std::vector<float> vectEmpattement;
    //maj vector empattement avec boucle for
    for (int i = 0; i < vehicule->getEssieux() - 1 ; i++) { //boucle qui tourne pendant
    x fois le nombre essieux -1
        vectEmpattement.push_back(ui->tw_empattement->item(0, i)->text().toFloat());
    }
    for (int i = vehicule->getEssieux() ; i <= 9 ; i++)
        { vectEmpattement.push_back(0.0);
    }
    vehicule->setEmpattement(vectEmpattement);
    pDialogVehicule->setVehicule(vehicule);

}

```

La troisième page pour la configuration et transfert des passages et des scénarios.

Extrait code on_pb_valider_released() de la classe DialogueConfigVehicule

```

/*!
 * \brief CVehiculemanager::addVehicule
 * \param vehicule
 * \return sqlQuery->isValid()
 * \details ajoute une voiture dans la base de données
 */
bool CVehiculemanager::addVehicule(const CVehicule &vehicule)
{
    QSqlQuery *sqlQuery = new QSqlQuery (pDataBase->database());
    //prépare la requête pour envoyer les valeurs des véhicules dans la base de données
    sqlQuery->prepare(
        "INSERT "
        "INTO`simulation_vehicule`(`v_nom`, `v_type`, `v_essieux`, `v_empattement1`, `v_"
        "empattement2`, `v_empattement3`, `v_empattement4`, `v_empattement5`, `v_empattement6`, `v_"
        "empattement7`, `v_empattement8`, `v_empattement9`)VALUES "
        "(:nom,:type,:essieux,:empattement1,:empattement2,:empattement3,:empattement4,:"
        ":empattement5,:empattement6,:empattement7,:empattement8,:empattement9)");

    sqlQuery->bindValue(":nom", vehicule.getNom()); sqlQuery-
    >bindValue(":type", vehicule.getType()); sqlQuery-
    >bindValue(":essieux", vehicule.getEssieux()); sqlQuery-
    >bindValue(":empattement1", vehicule.getEmpattement()[0]); sqlQuery-
    >bindValue(":empattement2", vehicule.getEmpattement()[1]); sqlQuery-
    >bindValue(":empattement3", vehicule.getEmpattement()[2]); sqlQuery-
    >bindValue(":empattement4", vehicule.getEmpattement()[3]); sqlQuery-
    >bindValue(":empattement5", vehicule.getEmpattement()[4]); sqlQuery-
    >bindValue(":empattement6", vehicule.getEmpattement()[5]); sqlQuery-
    >bindValue(":empattement7", vehicule.getEmpattement()[6]); sqlQuery-
    >bindValue(":empattement8", vehicule.getEmpattement()[7]); sqlQuery-
    >bindValue(":empattement9", vehicule.getEmpattement()[8]);

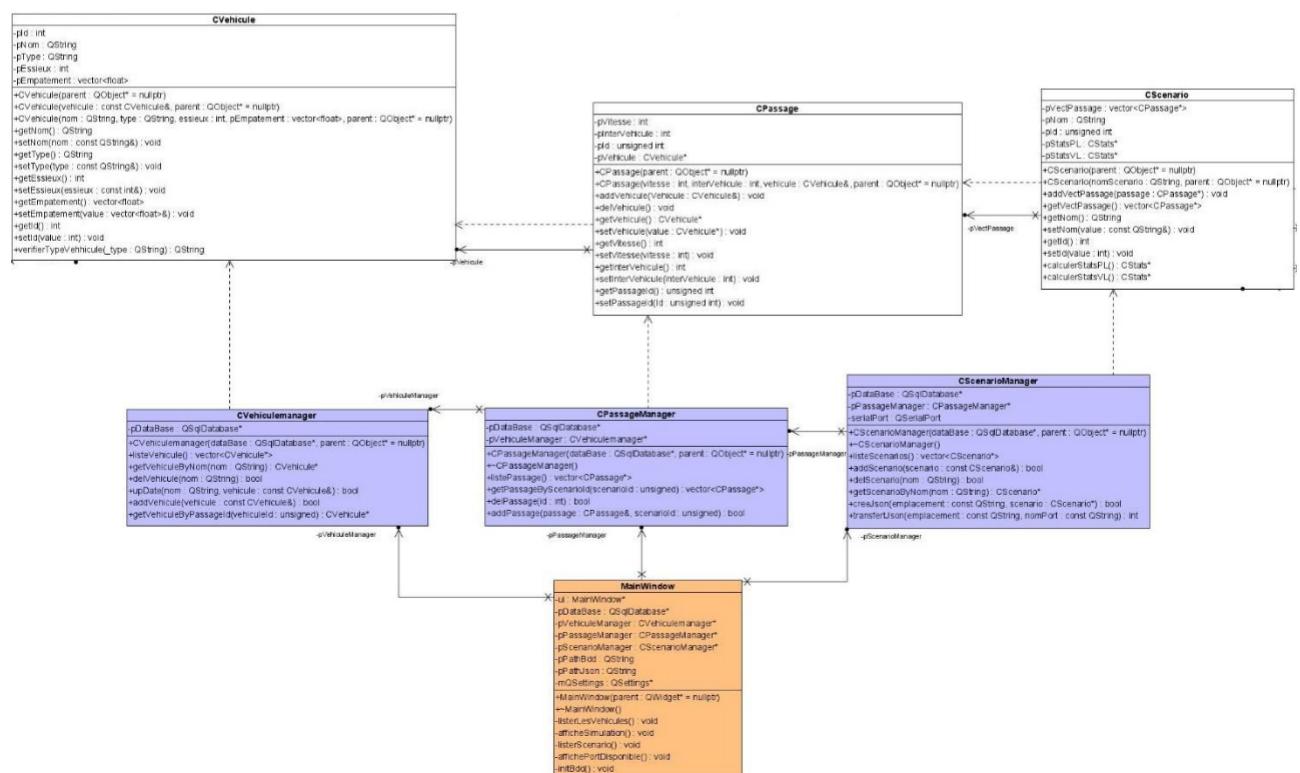
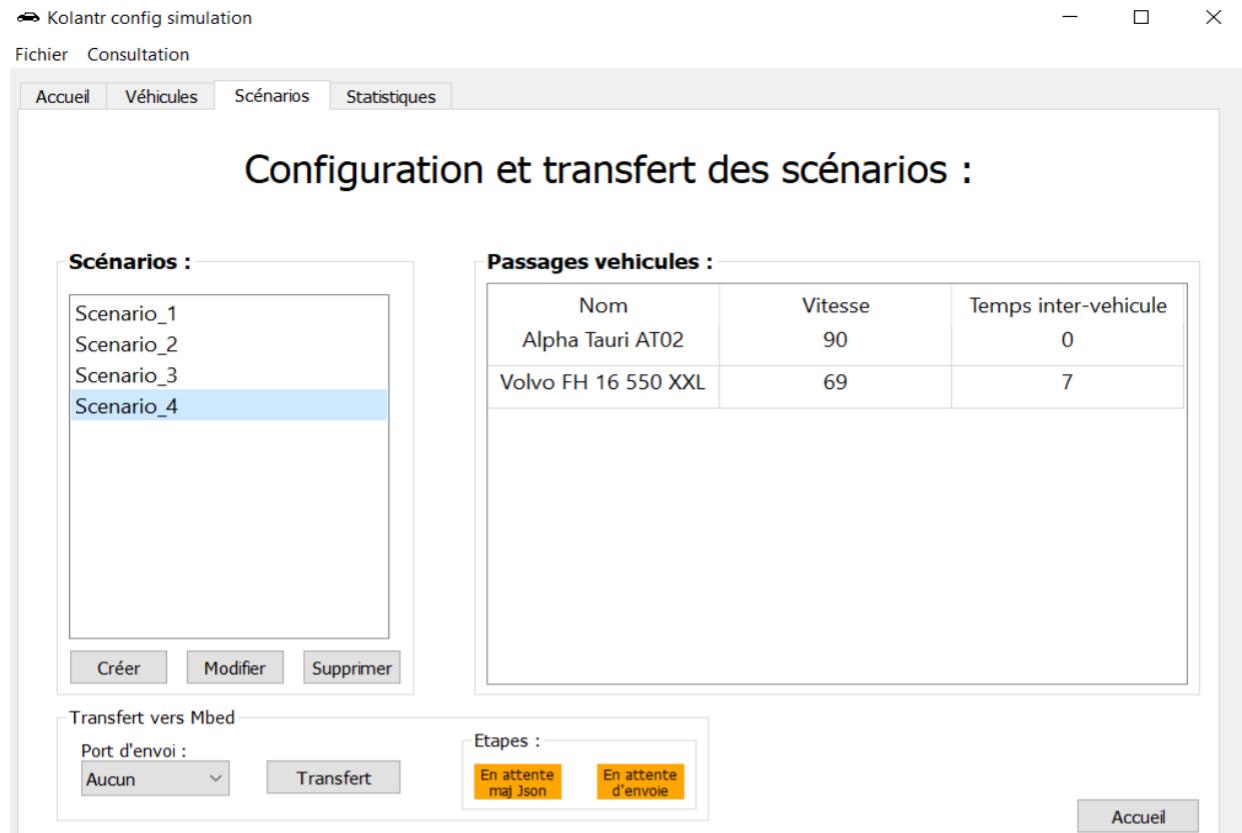
    sqlQuery->exec();
    return sqlQuery->isValid();
}

```

Extrait code addVehicule() de la classe CVehiculeManager

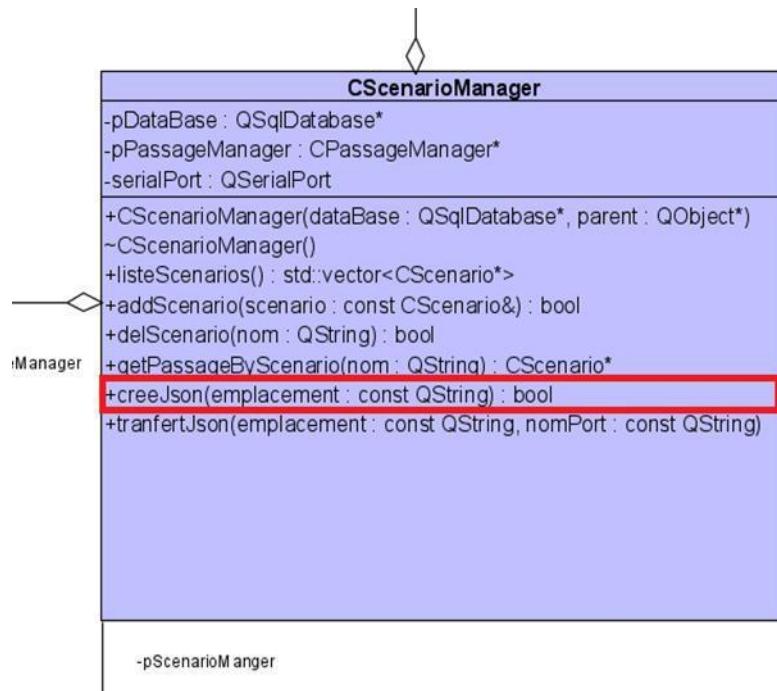
Le troisième onglet est pour la configuration et transfert des passages et des scénarios

Pour configurer les scénarios, cela fonctionne pareil que les véhicules mais avec en plus les classes passages et scénarios.



Extrait diagramme qui est utilisé pour la partie scénarios

L'onglet scénario sert aussi pour transférer des scénarios. L'application complète un fichier JSON avec les valeurs du scénario sélectionné. Par la suite, l'application envoie le fichier JSON en liaison série sur le port sélectionné.



Méthode `creeJson()` de la classe `CScenarioManager` qui remplit le JSON

```

/*!
 * \brief CScenarioManager::creeJson
 * \param emplacement
 * \param scenario
 * \return bool
 * \details remplissage fichier
Json */

bool CScenarioManager::creeJson(const QString emplacement, CScenario *scenario)
{
    QJsonObject objectJSON;
    QJsonObject passage;
    QJsonArray tabPassage;

    // creation object Json : scenario
    objectJSON.insert("nomScenario", scenario->getNom());
    // remplissage des passages
    for (long unsigned i = 0; i < scenario->getVectPassage().size(); i++)
    {
        //remplissage vehicule
        //passage.insert("nomVehicule", scenario->getVectPassage()[i]->getVehicule()->getNom());
        passage.insert("nbreEssieux", scenario->getVectPassage()[i]->getVehicule()->getEssieux()); passage.insert("typeVehicule", scenario->getVectPassage()[i]->getVehicule()->getType()); //tableau empatement
        QJsonArray tabEmpatement;
        for (int i2 = 0; i2 < scenario->getVectPassage()[i]->getVehicule()->getEssieux() - 1; i2++)
            tabEmpatement.push_back(round(scenario->getVectPassage()[i]->getVehicule()->getEmpatement()[i2]*100)/100); //ROund : arondi
        decimal
        passage.insert("empatement",tabEmpatement);
        //remplissage passage du vehicule
        passage.insert("vitesseVehicule", scenario->getVectPassage()[i]->getVitesse());
        passage.insert("tempsInterVehicule", scenario->getVectPassage()[i]->getInterVehicule()); //tableau du passage
        tabPassage.push_back(passage);
    }
    objectJSON.insert("passages",tabPassage);
    QJsonDocument document;
    document.setObject(objectJSON);

    QByteArray bytes = document.toJson(/*QJsonDocument::Compact*/QJsonDocument::Indented);
    QFile file(emplacement);
    if( file.open( QIODevice::WriteOnly | QIODevice::Text | QIODevice::Truncate ) )
    {
        QTextStream iStream( &file );
        iStream.setCodec( "utf-8" );
        iStream << bytes;
        file.close();
        return true;
    }
}

```

*Extrait code **creeJson(const QString emplacement, CScenario *scenario)** de la classe **CScenarioManager***

Kolantr config simulation

Fichier Consultation

Accueil Véhicules Scénarios Statistiques

Configuration et transfert des scénarios :

Scénarios :

- Scenario_1
- Scenario_2
- Scenario_3
- Scenario_4

Passages véhicules :

Nom	Vitesse	Temps inter-vehicule
Alpha Tauri AT02	90	0
Volvo FH 16 550 XXL	69	7

Créer **Modifier** **Supprimer**

Transfert vers Mbed

Port d'envoi : Aucun ▾ **Transfert**

Etapes :

Json à jour
Port d'envoi

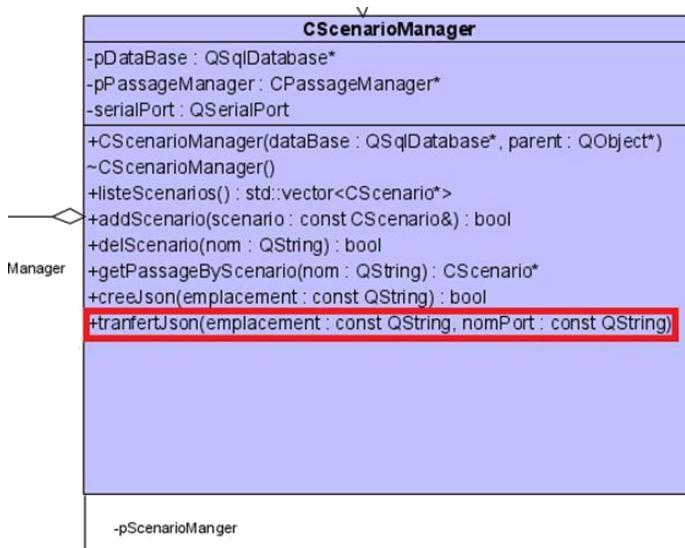
Accueil

scenario - Bloc-notes

Fichier Edition Format Affichage Aide

```
{
    "nomScenario": "Scenario_4",
    "passages": [
        {
            "emplacement": [
                3.2
            ],
            "nbreEssieux": 2,
            "tempsInterVehicule": 0,
            "typeVehicule": "v1",
            "vitesseVehicule": 90
        },
        {
            "emplacement": [
                3.5,
                0.9
            ],
            "nbreEssieux": 3,
            "tempsInterVehicule": 7,
            "typeVehicule": "p1",
            "vitesseVehicule": 69
        }
    ]
}
```

Fichier JSON remplit



Méthode transfertJson() de la classe CScenarioManager pour transferer le fichier .json sur un port serie

```

/*
 * \brief CScenarioManager::transfertJson
 * \param emplacement
 * \param nomPort
 * \return reponse
 * \details envoie sur la liaison
 * serie */
int CScenarioManager::transfertJson(const QString emplacement, const QString nomPort)
{
    serialPort.close();
    int reponse = 0;
    QString portWindows = nomPort;
    portWindows.truncate(3);
    if (portWindows == "COM") {
        serialPort.setPortName(nomPort);
        serialPort.setBaudRate(9600);
    } else {
        serialPort.setPortName("/dev/" + nomPort);
        serialPort.setBaudRate(9600);
    }
    QFile dataFile(emplacement);

    QByteArray writeData(dataFile.readAll());
    dataFile.close();

    qint64 bytesWritten = serialPort.write(writeData);
}

```

Extrait code transfertJson(const QString emplacement, const QString nomPort) de la classe //Caractère de fin

```

char fin = 0x04; CScenarioManager
bytesWritten = serialPort.write(&fin, 1);

```

}

Extrait code transfertJson(const QString emplacement, const QString nomPort) de la classe CScenarioManager

Kolantr config simulation

Fichier Consultation

Accueil Véhicules Scénarios Statistiques

Configuration et transfert des scénarios :

Scénarios :

- Scenario_1
- Scenario_2
- Scenario_3
- Scenario_4

Passages véhicules :

Nom	Vitesse	Temps inter-vehicule
Alpha Tauri AT02	90	0
Volvo FH 16 550 XXL	69	7

Transfert vers Mbed

Port d'envoi : **COM4**

Etapes :

- Json à jour
- Json envoyé

Accueil

Sélection du port d'envoi et envoie du fichier .json

YAT - [[Terminal1 *] - COM5 - Open - Connected]

File Terminal Send Receive Log View

Monitor

```
{
  "nomScenario": "Scenario_4",
  "passages": [
    {
      "emplacement": [
        3.2
      ],
      "nbreEssieux": 2,
      "tempsInterVehicule": 0,
      "typeVehicule": "vl",
      "vitesseVehicule": 90
    },
    {
      "emplacement": [
        3.5,
        0.9
      ],
      "nbreEssieux": 3,
      "tempsInterVehicule": 7,
      "typeVehicule": "pl",
      "vitesseVehicule": 69
    }
  ]
}<EOT>
```

Réception sur le port COM5

Boite de dialogue pour créer un scénario

Créer un scenario ? X

Veuillez remplir le formulaire

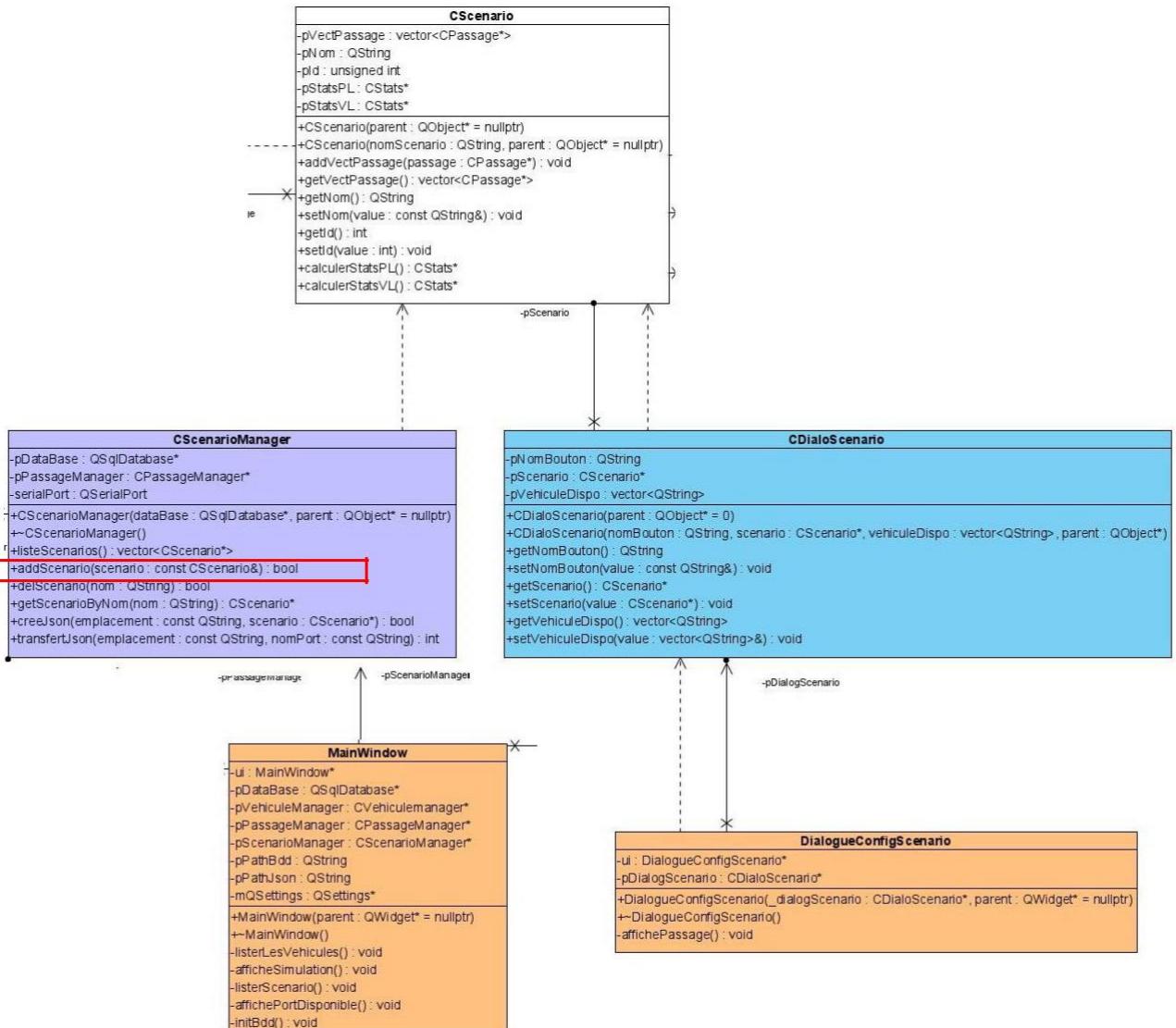
Nom du scénario : Scenario_		
Liste véhicules disponible :	Véhicule sélectionné :	Passage véhicule :
Toyota Yaris McLaren MCL35B Alpha Tauri AT02 Red Bull RB16B Volvo FH 16 550 XXL Dacia Duster Mercedes AMG W12 Mercedes Actros + REM3E Renault Scenic	Nom véhicule : Toyota Yaris Vitesse en km/h : 50 Temps inter-véhicule : 2	Passage véhicule :
< >	-->	Monter Descendre Supprimer
Annuler Valider		

Boite de dialogue pour modifier un scénario

Modifier un scénario : Scenario_3 ? X

Veuillez remplir le formulaire

Nom du scénario : Scenario_3																				
Liste véhicules disponible :	Véhicule sélectionné :	Passage véhicule :																		
Toyota Yaris McLaren MCL35B Alpha Tauri AT02 Red Bull RB16B Volvo FH 16 550 XXL Dacia Duster Mercedes AMG W12 Mercedes Actros + REM3E Renault Scenic	Nom véhicule : Volvo FH 16 550 XXL Vitesse en km/h : 50 Temps inter-véhicule : 2	Passage véhicule : <table border="1"> <thead> <tr> <th>Nom</th> <th>Vitesse</th> <th>Temps inter-vehicule</th> </tr> </thead> <tbody> <tr> <td>Mercedes AMG W12</td> <td>47</td> <td>0</td> </tr> <tr> <td>Red Bull RB16B</td> <td>67</td> <td>10</td> </tr> <tr> <td>Toyota Yaris</td> <td>59</td> <td>15</td> </tr> <tr> <td>Renault Scenic</td> <td>55</td> <td>8</td> </tr> <tr> <td>Dacia Duster</td> <td>55</td> <td>11</td> </tr> </tbody> </table>	Nom	Vitesse	Temps inter-vehicule	Mercedes AMG W12	47	0	Red Bull RB16B	67	10	Toyota Yaris	59	15	Renault Scenic	55	8	Dacia Duster	55	11
Nom	Vitesse	Temps inter-vehicule																		
Mercedes AMG W12	47	0																		
Red Bull RB16B	67	10																		
Toyota Yaris	59	15																		
Renault Scenic	55	8																		
Dacia Duster	55	11																		
< >	-->	Monter Descendre Supprimer																		
Annuler Valider																				



Extrait diagramme qui est utilisé pour les boites de dialogue des scénarios

La simulation se termine avec le dernier onglet pour l'analyse des statistiques d'un scénario.

The screenshot shows the application's main window with several tabs at the top: Accueil, Véhicules, Scénarios, and Statistiques. The Statistiques tab is currently active, displaying statistical data for a selected scenario. On the left, a list box titled 'Liste des scénarios' contains items: Scenario_1, Scenario_2, and Scenario_3. The main content area is divided into sections: 'Statistiques du scénario', 'Véhicules légers', and 'Poids lourds'. The 'Statistiques du scénario' section shows: Durée du scénario : 34 secondes, Nombres de véhicules passés : 7, and Vitesse moyenne : 54 km/h. The 'Véhicules légers' section shows: Nombres passés : 5, Vitesse moyenne : 58 km/h, and Vitesse max : 71 km/h. The 'Poids lourds' section shows: Nombres passés : 2, Vitesse moyenne : 51 km/h, and Vitesse max : 52 km/h. At the bottom right, there is a 'Accueil' button.

```

/*
 * \brief CScenario::calculerStatsVL
 * \return pStatsVL
 * \details calcule statistique vehicule leger
 */
CStats *CSenario::calculerStatsVL()
{
    // initialise les attributs à 0
    pStatsVL->init();
    int nbreVehicule = 0;
    int vitesseMoy = 0;
    int duree = 0;
    int vitesseMax = 0;
    int vitesseTotal = 0;
    // parcourt le vecteur des passages
    for (long unsigned i = 0; i < pVectPassage.size(); i++) {
        if (pVectPassage[i]->getVehicule()->getType() == "vl") {
            // calcule nombre de véhicules
            nbreVehicule += 1;
            // calcule la durée
            duree = (int unsigned) ((pVectPassage[i]->getInterVehicule() +
            duree)); // calcule la vitesse total
            vitesseTotal += (int unsigned) (pVectPassage[i]-
            >getVitesse()); // calcule la vitesse max
            if (vitesseMax < pVectPassage[i]->getVitesse())
                vitesseMax = (int unsigned) (pVectPassage[i]->getVitesse());
        }
    }
    // calcule la vitesse moyenne
    if (nbreVehicule > 0)
        vitesseMoy = vitesseTotal / nbreVehicule;

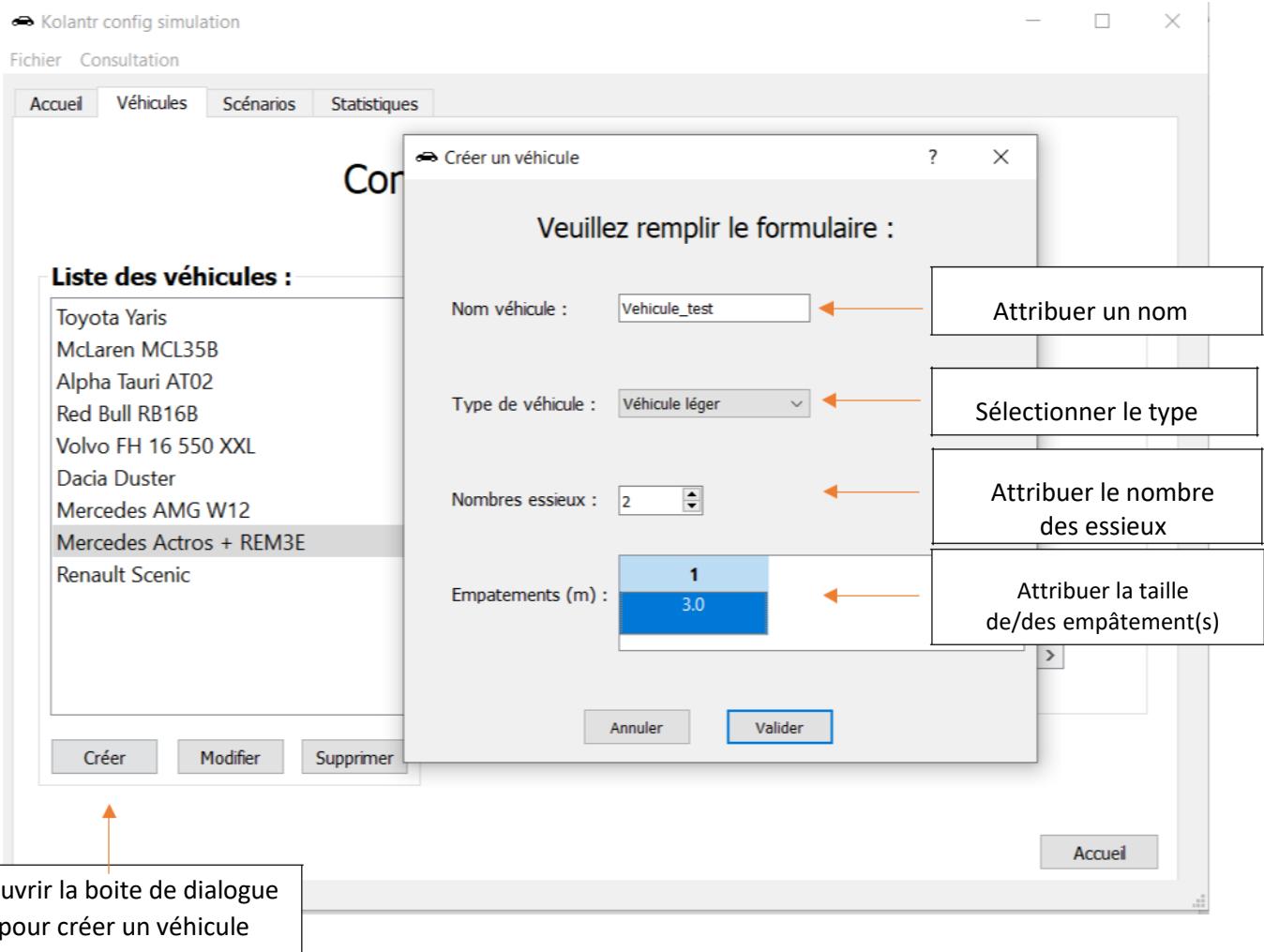
    pStatsVL->setNbreVehicule(nbreVehicule);
    pStatsVL->setVitesseMoy(vitesseMoy);
    pStatsVL->setDuree(duree);
    pStatsVL->setVitesseMax(vitesseMax);
    return pStatsVL;
}

```

Extrait calculerStatsVL() de la classe CScenario

D. Test

Test unitaire : ajouter véhicule



Une fois créé, le véhicule s'ajoute dans la base de données

Table : simulation_vehicule

vehicule_id	v_nom	v_type	v_essieu	v_emplacement1	v_emplacement2	v_emplacement3
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
10	82 Vehicule_test	vl	2	3.0	0.0	0.0

Extrait de la base de données de la table « simulation_vehicule »

Test unitaire : modifier un scénario

Kolantr config simulation

Fichier Consultation

Accueil Véhicules Scénarios Statistiques

Configuration et transfert des scénarios :

Scénarios :

- Scenario_1
- Scenario_2
- Scenario_3
- Scenario_4

Passages véhicules :

Nom	Vitesse	Temps inter-véhicule
Alpha Tauri AT02	90	0
Volvo FH 16 550 XXL	69	7

Modifier un scénario : Scenario_4

Veuillez remplir le formulaire

Nom du scénario : Scenario_4

Liste véhicules disponibles :

- Toyota Yaris
- McLaren MCL35B
- Alpha Tauri AT02
- Red Bull RB16B
- Volvo FH 16 550 XXL
- Dacia Duster
- Mercedes AMG W12
- Mercedes Actros + REM3E
- Renault Scenic
- Vehicule_test

Véhicule sélectionné :

Nom véhicule : Vehicule_test

Vitesse en km/h : 55

Temps inter-véhicule : 2

Passage véhicule :

Nom	Vitesse	Temps inter-véhicule
Alpha Tauri AT02	90	0
Vehicule_test	55	2
Volvo FH 16 550 XXL	69	7

Attribuer une vitesse à la voiture sélectionnée

Attribuer un temps inter-véhicule à la voiture sélectionnée

Ajouter le passage au scénario

Paramétriser l'ordre des passages au scénario

Une fois modifié, le scénario se met à jour dans la base de données

Table : simulation_scenario	
sc_id	sc_nom
4	232 Scenario_4

Extrait de la base de données de la table « simulation_scenario »

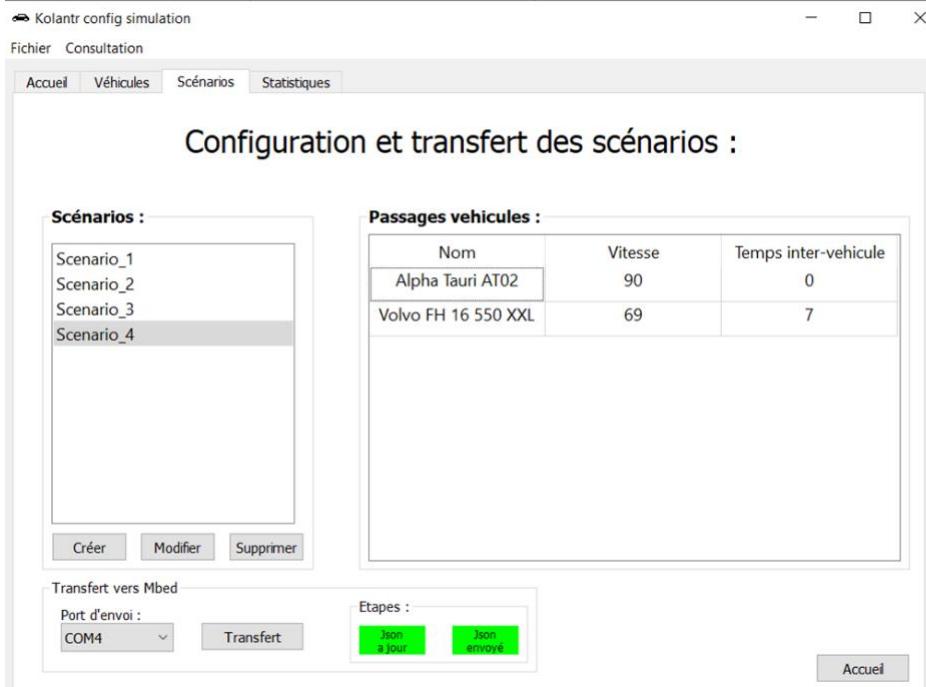
Table : simulation_passage					
	p_id	p_vitesse	p_inter-vehicule	id_scenario	id_vehicule
19	556	90	0	232	15
20	557	55	2	232	82
21	558	69	7	232	17

Extrait de la base de données de la table « simulation_passage »

Table : simulation_vehicule								
	vehicule_id	v_nom	v_type	v_essieux	v_empatement1	v_empatement2	v_empatement3	v_empatement4
3	15	Alpha Tauri AT02	vl	2	3.20000004768372	0.0	0.0	0.0
5	17	Volvo FH 16 550 XXL	pl	3	3.5	0.899999976158142	0.0	0.0
10	82	Vehicule_test	vl	2	3.0	0.0	0.0	0.0

Extrait de la base de données de la table « simulation_vehicule »

Tests d'intégration : transfert d'un scénario vers Mbed



Trame envoyée du côté de l'application

The screenshot shows the 'WATCH' window of a debugger. It lists several memory locations with their addresses and values. The data corresponds to the JSON message sent from the application. The window also shows a 'CALL STACK' and a 'PAUSED ON BREAKPOINT' status.

Address	Value
445	116 't'
446	121 'y'
447	112 'p'
448	101 'e'
449	86 'V'
450	101 'e'
451	104 'h'
452	105 'i'
453	99 'c'
454	117 'u'
455	108 'l'
456	101 'e'
457	34 '''
458	58 ':'
459	32 ''
460	34 '''
461	118 'v'
462	108 'l'

La trame est reçue du côté Mbed

E. Bilan personnel technique

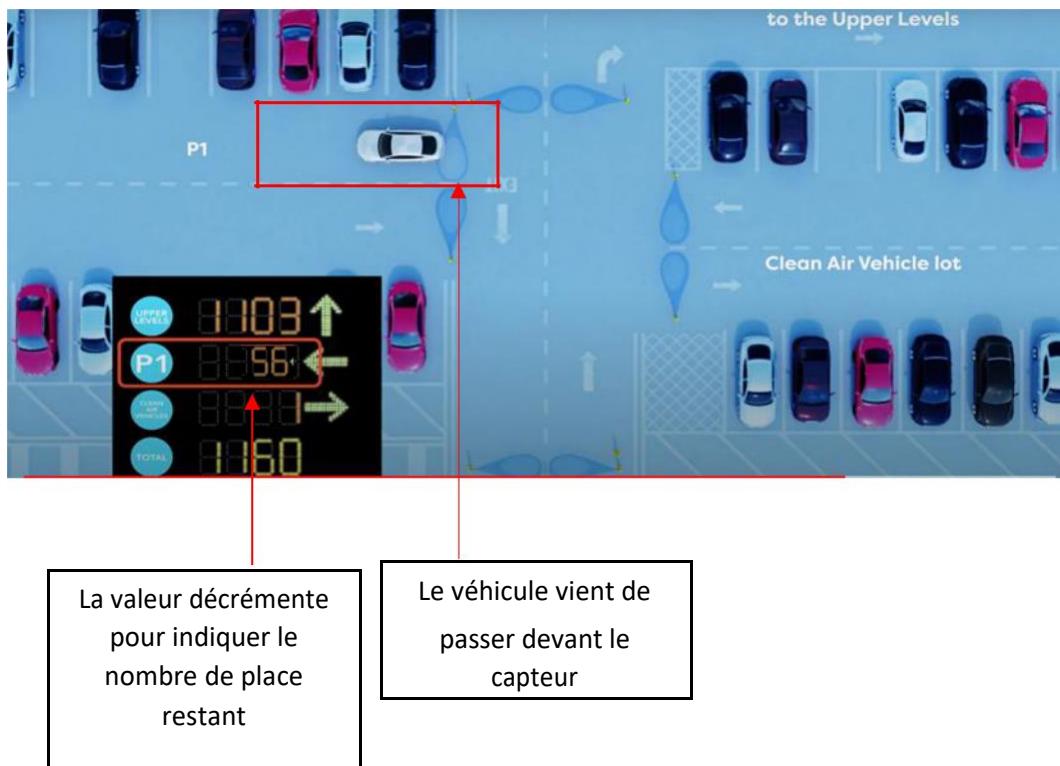
Tache à réaliser :

- * Générer les modèles de véhicule : réalisé
- * Configurer véhicule : réalisé
- * Générer les scénarios de simulations : réalisé
- * Configurer un scénario : réalisé
- * Calculer stats scénario : réalisé
- * Transférer un scénario : réalisé

F. Partie physique

Autre capteur qui aurait pu être utilisé pour le détecter le passage de véhicules : OVS-01CC

Le OVS-01CC d'OPTEX est conçu pour détecter et permet ainsi de compter de façon fiable les véhicules en mouvement jusqu'à 60km/h. Il est souvent utilisé pour compter le nombre de véhicules dans un parking. Il élimine les contraintes d'installation de boucle au sol. Plus besoin de creuser. L'OVS-01CC peut être posé à 50 cm du sol et peut détecter les petits, les gros véhicules ainsi que des carrosseries plastiques

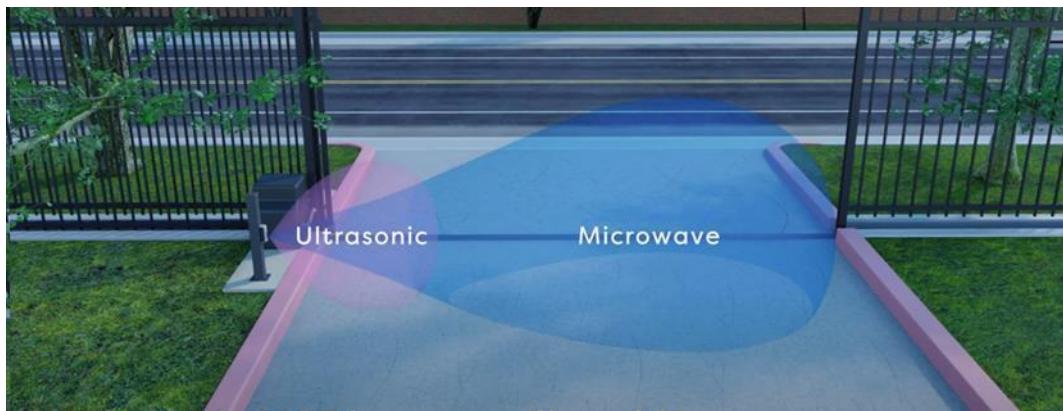


OVS-01CC Caractéristiques

Méthode de détection	Hyperfréquence (changement Doppler et FMCW)
Fréquence	24GHz
Temps de réponse	300msec
Alimentation	12-24VDC
Consommation	Max 190mA (à 24VDC) avec chauffage ON Max 70mA (à 24VDC) quand le chauffage est OFF
Sortie	Sortie relais DC30V, 0.3A (NO/NC switchable)
Entrée	Entrée NO/NC
Distance de détection	Distance maximum programmable de 1 à 8m
Vitesse du véhicule détecté	De 2 à 60km/h
Paramètres	Sensibilité Sorties Distance max.
Indicateur	Niveau de 1 à 5 2 x (N.O. / N.C) Contacts 1m, 2m, 3m, 4m, 5m, 6m, 7m, 8m. Fonctionnement normal Attente : vert. Détection: rouge, notification environnement défavorable : clignotant lent vert. Contrôle de zone détection Pas de détection : clignotant vert Détection : clignotant rouge Calibration Préparation : Clignotant Lent Bleu, Calibration : Clignotant Rapide Bleu Réinitialisation Reset complet: Clignotant jaune pendant 2 secondes
Température de fonctionnement	-30°C à 50°C
Taux d'humidité	95% max. (sans condensation)
Indice international de protection	IP65
Fonctionnement	Intérieur/Extérieur
Hauteur de pose	Installation hauteur à 500mm.
Angle d'ajustement horizontal	Horizontallement +/-30 deg. (par 5 deg)
Poids	480g (Incluant les accessoires)
Accessoires	4 vis et manuel d'installation

Caractéristique OVS-01CC

Le capteur utilise des fréquences 24ghz ce sont des hyperfréquences car les fréquences sont entre 300MHz et 300GHz.



- Le détecteur compte les véhicules qui se déplacent grâce à l'utilisation de deux technologies : le changement Doppler et les FMCW

L'effet Doppler :

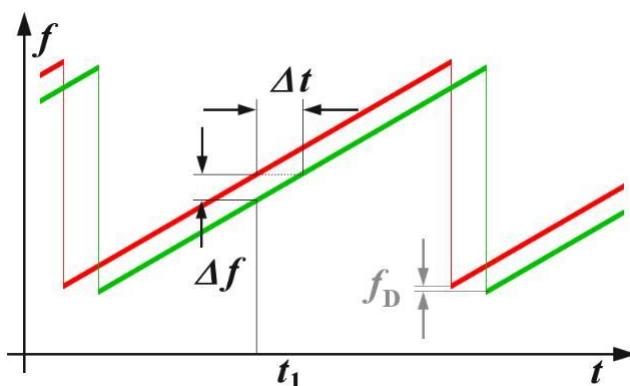
L'effet Doppler est un phénomène qui caractérise le changement de fréquence de l'onde reçue par le récepteur lorsque l'émetteur et le récepteur se déplacent l'un par rapport à l'autre.

Si le capteur est devant la voiture et que la voiture se rapproche : les fronts d'onde sont serrés. Par conséquent, la fréquence des bips reçus est plus grande que la fréquence émise par la voiture. Par contre, si le capteur est derrière la voiture et donc la voiture s'éloigne : les fronts d'onde sont desserrés. Alors, la fréquence des bips reçus est plus petite que celle émise par la voiture.

Onde continue modulée en fréquence (FMCW) :

Le changement permanent du signal émis à proximité de la fréquence de référence est utilisé pour détecter des cibles fixes. Lorsque le radar reçoit l'écho, il peut mesurer la fréquence du signal réfléchi par la cible. En se référant au moment où la même valeur de fréquence est envoyée, le temps entre l'envoi et la réception de la fréquence peut être mesuré, de sorte que la distance radar-cible peut être mesurée, tout comme un radar à impulsions.

Le radar à modulation de fréquence en onde continue (FMCW) change progressivement la fréquence de son signal à une vitesse de rampes montantes et descendantes.



Mesure de distance par un système à onde continue modulée en fréquence (FMCW)

VI. Etudiant Théo SERNOT

A. Description partie personnelle

Le système KolAnTr doit permettre de collecter, d'envoyer périodiquement et de permettre à la consultation des données via un site web d'une campagne de mesures.

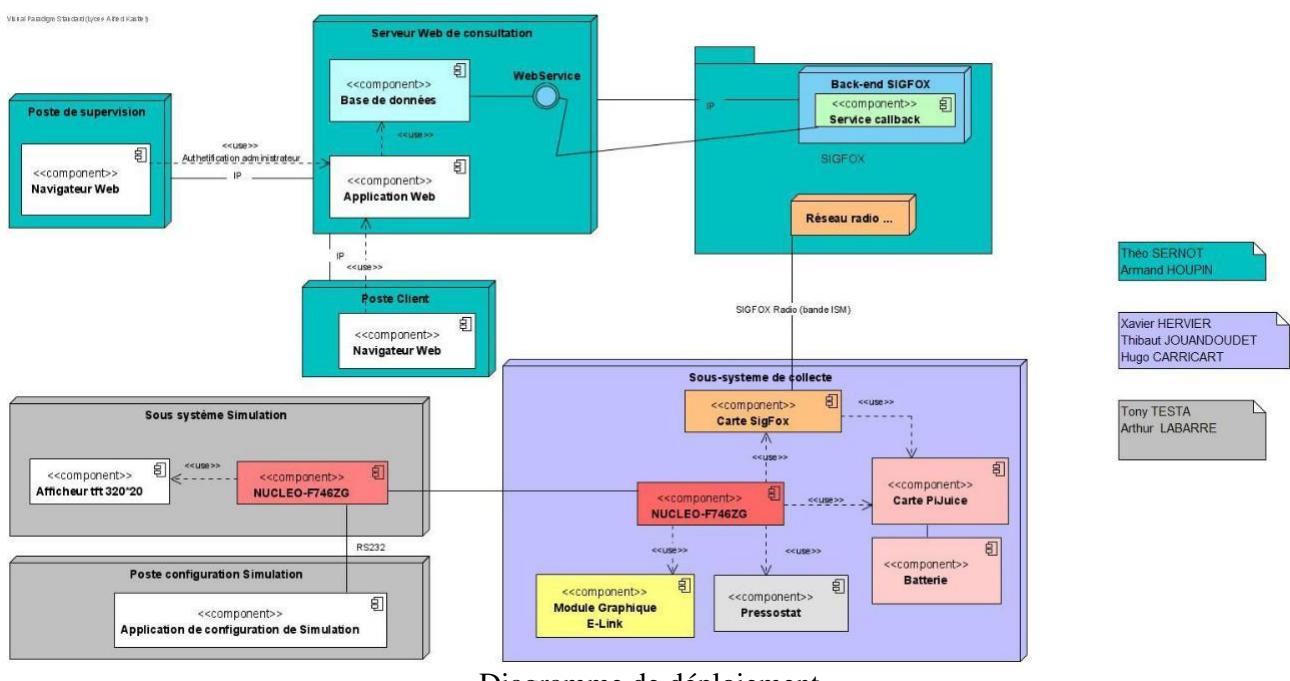
Une campagne de mesure consiste à détecter le passage de véhicule sur une chaussée et d'établir des statistiques de trafic routier comme le nombre de véhicule par heure par exemple.

L'objectif de détection d'une campagne de mesures est multiple il peut permettre de dimension les chaussées en fonction du trafic des poids lourds, d'optimiser l'infrastructure à partir des niveaux de trafic, d'optimiser les opérations d'entretien du réseaux de trafic à partir des trafics tous véhicules et du trafic des poids lourds, d'avoir une connaissance précise des vitesses pratiquées, de définir l'impact des mesures de sécurité sur le comportement des automobilistes.

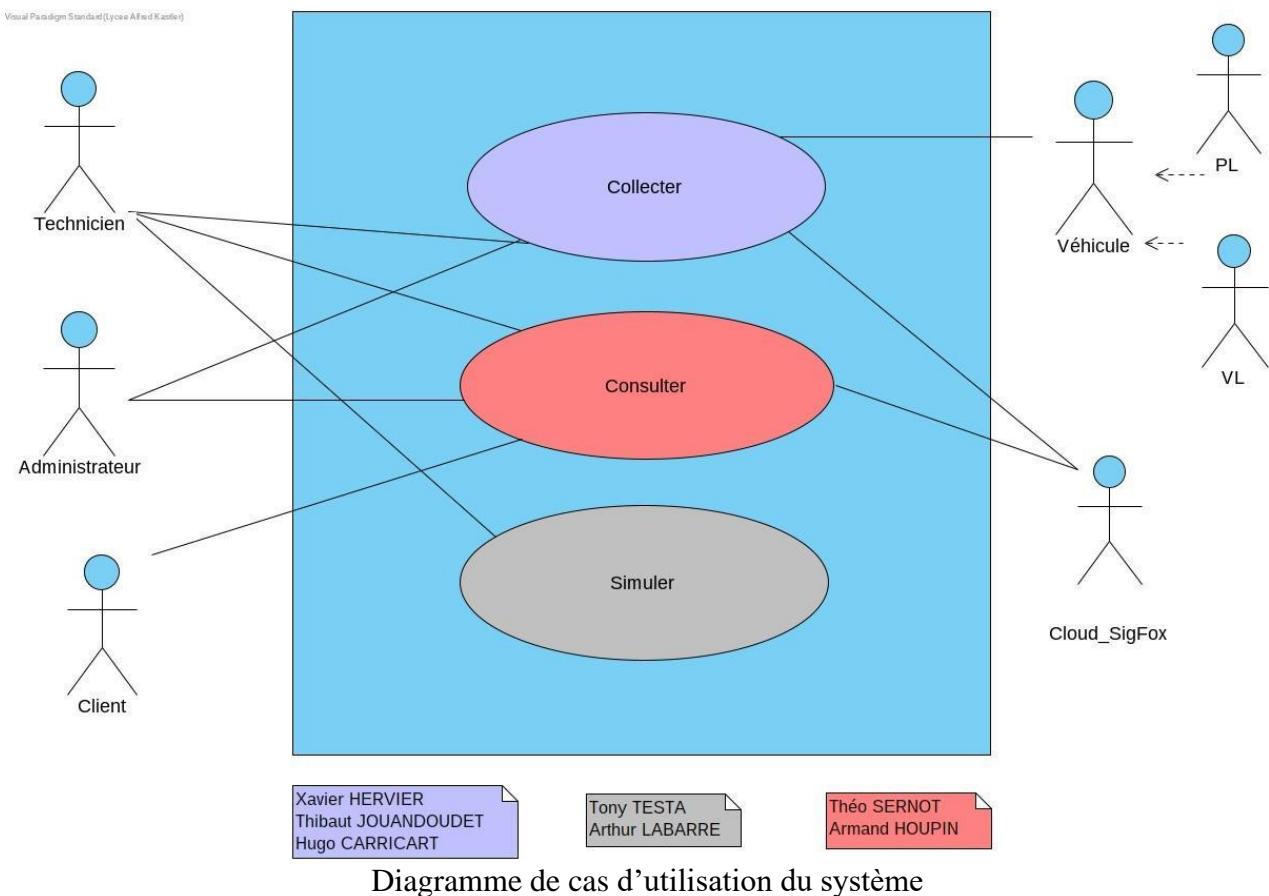
Le sous système consultation de KolAnTr est un site Web qui permet à l'exploitant de gérer une campagne de mesure (BREAD : Browse, Read, Edit, Add, Delete) ; importer les données issues d'un sous système de collecte ; consulter les données d'une campagne de collecte issues de la télémétrie ou d'une importation des données ; exporter les données d'une campagne dans un format exploitable par un tableur.

Les exigences de ma partie était de collecter les statistiques des campagnes de collecte, collecter via les boîtiers de collecte, collecter via Sigfox, consulter une campagne de mesure, exporter les statistiques et consulter les statistiques.

Ces différentes exigences ont été réalisé avec le développement d'un web service ainsi qu'une base de données.



B. Conception détaillé



Ce diagramme est un diagramme UML (Unified Modeling Language, Langage de Modélisation Unifié) de cas d'utilisation. Il est utilisé pour représenter le comportement fonctionnel du système logiciel de KolAnTr. On y voit les différents acteurs du projet. En effet il y a le Technicien qui agit sur les 3 sous-systèmes qui sont la collecte, la consultation et la simulation, l'Administrateur qui agit lui sur la partie collecte et la partie consultation, le Client qui agit seulement sur la partie consulter, le Véhicule qui peut être de type poids lourd (PL) ou véhicule léger (VL) agit seulement sur consulter et enfin le Cloud Sigfox qui agit sur les sous systèmes de collecte et de consultation.

Au final ce diagramme de cas d'utilisation permet de décrire les interactions entre les acteurs qui sont donc Technicien, Administrateur, Client, Véhicule ou bien le Cloud Sigfox et le système que l'on voit ici en sous systèmes. On peut donc voir que chaque utilisateur du système agissent sur des sous systèmes bien précis. Sur le diagramme, on y voit donc les interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. On y voit donc, représenté par un ligne, la mise en relation par association d'un acteur et d'un cas d'utilisation. Les acteurs humains sont donc le Technicien, l'Administrateur et le client. On y voit aussi des relations entre acteurs car en effet, les acteur poids lourd (PL) et véhicule léger (VL) sont des sous-types de l'acteur Véhicule.

Sur le diagramme de cas d'utilisation du système, on y trouve aussi les différentes personnes qui ont travaillé sur les différentes parties, comme Xavier HERVIER, Thibaut JOUANDOUDET et HUGO CARRICART pour le sous-système de Collecte, Tony TESTA et Arthur LABARRE pour le sous-système de Simulation et enfin Théo SERNOT (donc moi-même) et Armand HOUPIN pour le sous-système de Consultation.

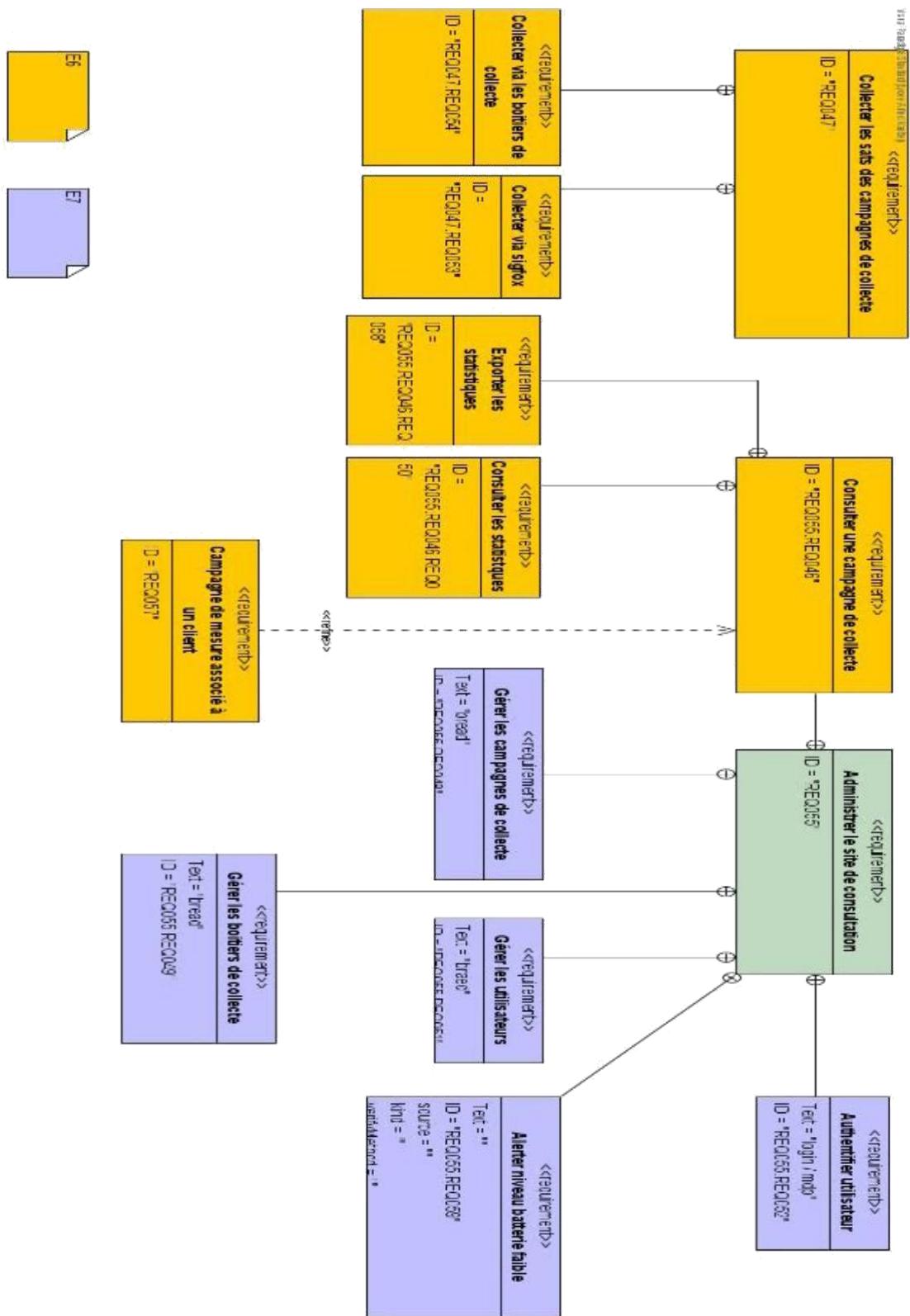


Diagramme d'exigence de la partie consultation

Une exigence spécifie un besoin ou une règle qui doit être satisfaite. Une exigence peut spécifier une fonction qu'un système doit exécuter ou des critères de performance à atteindre.

Ce diagramme est un diagramme d'exigence, il permet de comprendre efficacement les besoins fonctionnels, de performances et d'interface.

Il décrit graphiquement les capacités et les contraintes qui doivent être satisfaite par le système. Il interprète donc le cahier des charges et est relié aux aspects structurels et comportementaux du système. Il reprends donc aussi les contraintes du cahier des charges.

Il est utilisé pour collecter toutes les exigences techniques, légales, physiques, commerciales normatives ou autres du projet. Ce diagramme est transversal à l'intégralité du système et permet en plus de hiérarchiser les spécifications.

Il contient des blocs d'exigences fonctionnelles qui comporte une entête avec l'indication <<requirement>>, son titre et un corps avec un numéro d'identification (ID) et un définition précise. Certaines exigences se décompose en sous-exigences, les liens hiérarchiques entre exigences et sous-exigences sont matérialisés par une liaison avec un cercle du coté de l'exigence de plus haut niveau. La numérotation (ID) reprends aussi dans sa forme la hiérarchie des blocs.

Étant étudiant E6, on peut observer que pour ma partie, collecter les stats des campagnes de collecte exige que ce soit fait via les boîtiers de collecte et via Sigfox. On voit aussi les sous-exigences exporter les statistiques, consulter les statistiques et campagne de mesure associé à un client reliés à consulter une campagne de collecte.

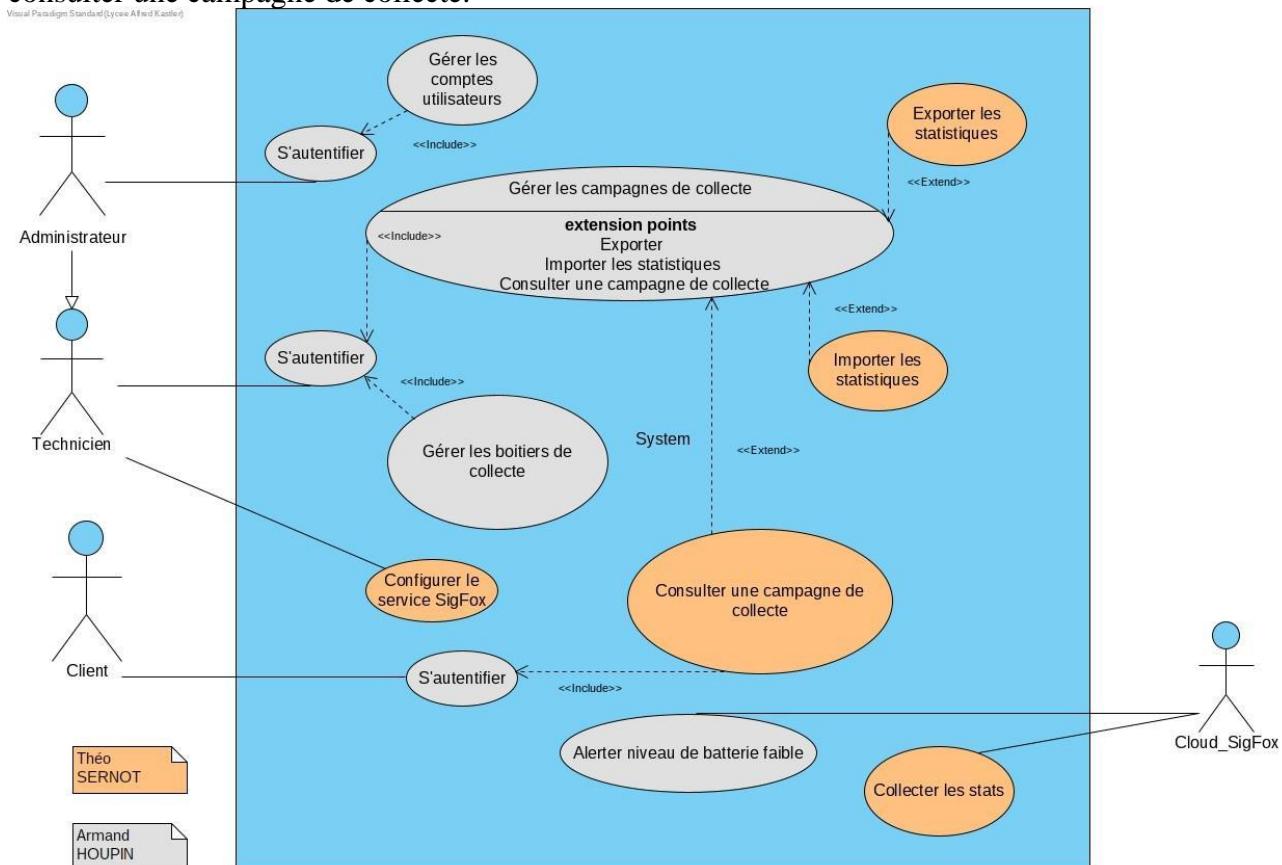


Diagramme de cas d'utilisation de la partie consultation

Dans ce diagramme de cas d'utilisation de la partie consultation, on peut voir, pour ma partie que le Technicien qui hérite d'Administrateur s'occupe de configurer le services Sigfox, le Cloud Sigfox collecte les stats, le Client peut consulter une campagne de collecte. On peut voir aussi qu'on peut importer et exporter les statistiques depuis la gestion des campagnes de collecte.

C. Implémentation

Pour réaliser ce projet, différentes technologies ont été utilisées.

Premièrement, nous avons utilisé Apache, un serveur HTTP. Un serveur web, c'est le logiciel qui réponds et renvoie les pages HTML au navigateur quand on consulte un site internet. Il est gratuit et open-source et permet, comme son nom l'indique, de servir du contenu sur le web.

Quand quelqu'un souhaite visiter un site web, il saisit le nom de domaine dans la barre d'adresse de son navigateur. Puis le serveur web fournit les fichiers demandés en agissant comme un livreur virtuel.

Bien qu'appelé serveur web, Apache n'est pas un serveur physique mais plutôt un logiciel qui s'exécute sur un serveur. Son travail consiste à établir une connexion entre un serveur et les navigateurs des visiteurs du site web (Firefox, Google Chrome, Safari, etc.) tout en délivrant des fichiers entre eux (structure client-serveur).

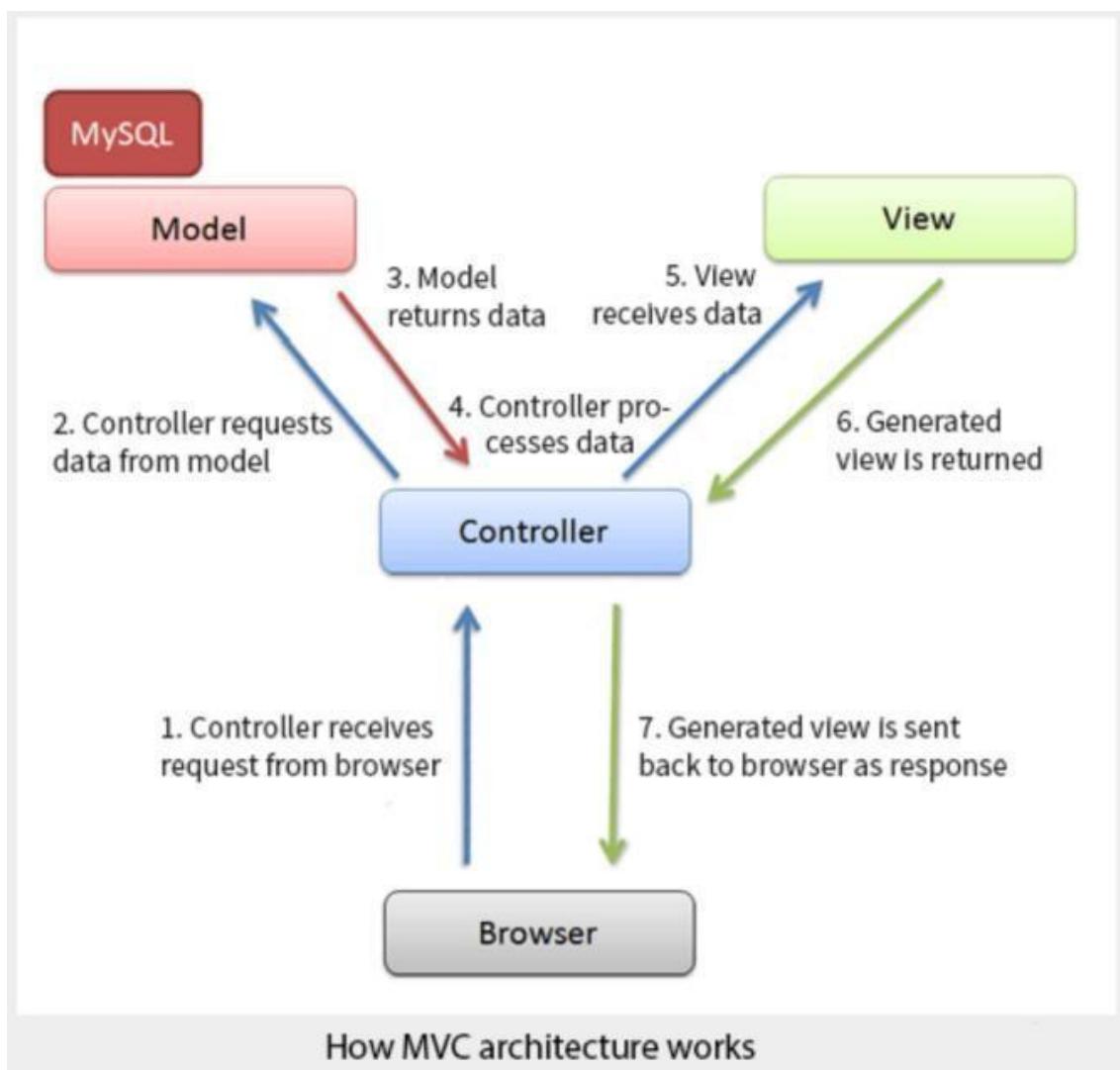
Apache est un logiciel multiplateforme, il fonctionne donc à la fois sur les serveurs Unix et Windows. Lorsqu'un visiteur souhaite charger une page sur un site web, son navigateur envoie une requête au serveur et Apache renvoie une réponse avec tous les fichiers demandés (texte, images, etc.). Le serveur et le client communiquent via le protocole http et Apache est responsable de la communication fluide et sécurisée entre les deux machines.



Ensuite, nous nous sommes servis de Laravel, un framework web open-source écrit en PHP respectant le principe modèle-vue-contrôleur (mvc).

Le modèle-vue-contrôleur est un motif d'architecture logicielle destiné aux interfaces graphiques et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs. Un modèle contient les données à afficher. Une vue contient la présentation de l'interface graphique. Un contrôleur contient la logique concernant les actions effectuées par l'utilisateur.

En résumé, lorsqu'un client envoie une requête à l'application : la requête envoyée depuis la vue est analysée par le contrôleur (via par exemple une callback), le contrôleur demande au modèle approprié d'effectuer les traitements et informe à la vue que la requête est traitée, la vue notifiée fait une requête au modèle pour se mettre à jour (par exemple affiche le résultat du traitement via le modèle). Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose.



Nous avons utilisé aussi Artisan, un composant qui permet d'effectuer un certain nombre d'actions, comme par exemple la génération de fichiers, comme les Contrôleurs, les Models ou encore les fichiers de tests. Il est très pratique pour la gestion des migrations en base de données.



Puis nous avons exploité Eloquent, un ORM (object-relational mapping) élégant et efficace. Son utilité essentielle se trouve dans le traitement de données relationnelles. Comme relations possibles, il y a **OneToOne** (1:1), la plus simple mais la moins utile, dans deux tables A et B de relation 1:1, un attribut de la table A se rapporte seulement à un attribut de la table B ; **hasOne**, chaque enregistrement de la table principale (celle contenant la clé primaire) peut avoir au plus un enregistrement dans la table secondaire (celle contenant la clé étrangère). Inversement, chaque enregistrement de la table secondaire est relié à un et exactement un enregistrement dans la table principale ; **belongsTo** qui est l'inverse.



Composer a aussi été utilisé, c'est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet de déclarer et d'installer les bibliothèques dont le projet principal a besoin.



Nous avons aussi exploiter Voyager, c'est tout simplement le back office manquant de Laravel. En effet, il est bien fait et permet de gagner du temps. Avec Voyager on a accès à une interface d'administration pour notre application Laravel, on peut facilement ajouter, modifier ou supprimer des données dans l'application, construire un menu ou encore mettre en place les BREAD (Browse, Read, Edit, Add, Delete).



Enfin, Sigfox sert de serveur Web et permet d'envoyer des call-back grâce à son back-end, dans mon projet, les trames envoyés par la platine Mbed sont relayés par le simulateur de réseau Sigfox, le back-end génère une call-back



Par la suite, j'ai donc établi ensuite le schéma entités-relations de la base de données. Il permet d'illustrer comment les « entités » sont liées les unes aux autres au sein du système. Ce diagramme a été utilisé pour concevoir la base de données relationnelle de KolAnTr. Il a permis de définir la logique et les règles à utiliser ou les technologies spécifiques à employer. C'est l'étape initiale pour déterminer les prérequis du projet. On l'a aussi utilisé plus tard pour modéliser notre base de données.

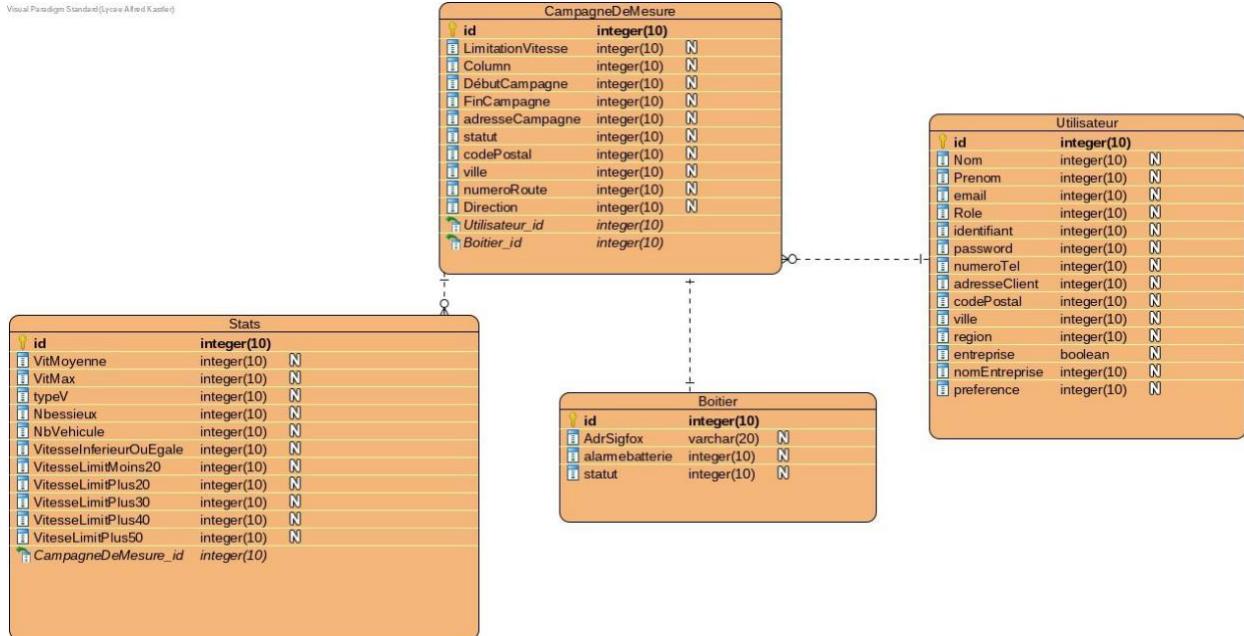


Schéma entités-relations base de données

On peut y voir qu'une campagne de mesure est associée à plusieurs statistiques mais que plusieurs statistiques sont associés qu'à une seule campagne, un boîtier est associé à une campagne de mesure et inversement et enfin une campagne de mesure peut avoir plusieurs utilisateurs mais un utilisateur est associé à une seule campagne de mesure.

On peut voir que dans la table Stats on a comme attributs : VitMoyenne, VitMax, typeV, NbVehicule, VitesseInferieurOuEgale, VitesseLimitMoins20, VitesseLimitPlus20, VitesseLimitPlus30, VitesseLimitPlus40, VitesseLimitPlus50, CampagneDeMesure_id.

Dans la table CampagneDeMesure on a comme attributs : LimitationVitesse, Column, DébutCampagne, FinCampagne, adresseCampagne, statut, codePostal, ville, numeroRoute, Direction, Utilisateur_id, Boitier_id, Utilisateurid.

Dans la table Boîtier : AdrSigfox, alarmebatterie, statut

Puis pour finir, dans la table Utilisateur : Nom, Prenom, email, Role, identifiant, password, numeroTel, adresseClient, codePostal, ville, region, entreprise, nomEntreprise, preference

The screenshot shows the Database Voyager dashboard with a sidebar containing icons for database management. The main area lists various tables with their names in blue: CompteAdmin, CompteClient, boitiers, campagnedemesures, categories, failed_jobs, menus, pages, permissions, posts, roles, sigfoxmessages, statistiques, translations, user_roles, and users. Each table entry includes a 'Parcourir le BREAD' button (orange), an 'Editer le BREAD' button (blue), and a 'Supprimer le BREAD' button (red). To the right of the table list is a section titled 'Actions sur le tableau' (Actions on the table) with three rows of buttons for 'Vue' (View), 'Editer' (Edit), and 'Supprimer' (Delete).

Tableau de bord → Database Voyager

On peut voir ici la base de données qui a été créée avec les différentes tables qui sont : CompteAdmin, CompteClient, boitiers, campagnedemesures, categories, failed_jobs, menus, pages, permissions, posts, roles, sigfoxmessages, statistiques, translations, user_roles et users.

On y voit aussi que les BREAD ont été créés, en effet on voit qu'on peut par exemple Parcourir le BREAD, Editer le BREAD ou encore Supprimer le BREAD.

The screenshot shows the Database Voyager interface for managing relations between tables. It displays four relation entries:

- boitiers**: Type: Relation. Options: Parcourir, Lire, Éditer, Ajouter, Supprimer. Relation: boitiers. Action: Ouvrir Détails de la relation.
- stat_id**: Type: integer. Clé: Obligatoire: Non. Options: Naviguer, Lire, Éditer, Ajouter, Supprimer. Relation: stat_id. Action: Ouvrir Détails de la relation.
- users**: Type: Relation. Options: Parcourir, Lire, Éditer, Ajouter, Supprimer. Relation: users. Action: Ouvrir Détails de la relation.
- statistiques**: Type: Relation. Options: Parcourir, Lire, Éditer, Ajouter, Supprimer. Relation: statistiques. Action: Ouvrir Détails de la relation.

At the bottom left is a red button labeled 'Créer une relation' (Create a relation).

On peut aussi ajouter ou modifier les relations entre les différentes tables.

Concernant le cheminement des informations, le sous-système de collecte envoie les trames à l'aide du modem branché sur une carte mbed.



Carte Mbed avec le Modem

Puis les trames sont reçues par le simulateur Sigfox à l'aide du dongle Sigfox. Le back-end génère les callbacks HTTP à destination du Webservice qui permet au PC développeur de recevoir les données. Les données des callbacks sont ensuite décodées par le webservice et elles sont enregistrées dans la base de données MySQL. Pour finir le simulateur de requête permet de tester le webservice en jouant le rôle du back-end Sigfox



Dongle Sigfox permettant au simulateur de recevoir les trames

Sigfox Network Emulator



CONFIGURATION MESSAGES ABOUT Authentication enabled 

RADIO DEVICES CALLBACKS CONNECTION

Devices configuration

- Device 1 Identifier (hex!) Name
- Device 2 Identifier (hex!) Name
- Device 3 Identifier (hex!) Name
- Device 4 Identifier (hex!) Name
- Device 5 Identifier (hex!) Name

SAVE **CANCEL**

Sigfox Network Emulator



CONFIGURATION MESSAGES ABOUT Authentication enabled 

RADIO DEVICES CALLBACKS CONNECTION

Send duplicate

Custom payload:

URL syntax: <http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...>
Available variables: device, time, duplicate, snr, station, data, avgSnr, rssi, seqNumber, LQI, ack
Custom variables:
Url pattern

Use HTTP Method

Headers **ADD**

Content type

Body

Callbacks - SERVICE ACKNOWLEDGE

Channel

Send duplicate

URL syntax: <http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...>
Available variables: device, time, duplicate, snr, rssi, station, avgSnr, infocode
Url pattern

Use HTTP Method

Headers **ADD**

Callbacks - SERVICE STATUS

Configuration du simulateur Sigfox avec la création de la callback

2ED518 27 mai 2021 08:55:36 408 355388283830002609044809  

Trame envoyé avec test de callback validé

Les trames sont faites ainsi :

T	Nº Véhicule	Nb essieux	Vit Moy	Vit Max	Vit OK	Vit LM20	Vit LP20	Vit LP30	Vit LP40	Vit LP50
0	3	5	0	7	1	0	5	0	1	3
0	110101010101	00111000100000	01010000000000	01110000000000	01110000000000	01110000000000	01110000000000	01110000000000	01110000000000	01110000000000
0011010101010011	11000100000000	00101000000000	00101000000000	00101000000000	00101000000000	00101000000000	00101000000000	00101000000000	00101000000000	00101000000000
3	5	5	3	8	8	2	8	3	8	3
						0	0	0	2	6
						0	0	0	4	4
						9	9	9	8	0
						0	0	0	4	9

Pour décoder la trame, examinons de plus près le contrôleur SigfoxMessageController.

```
namespace App\Http\Controllers;

use App\Models\sigfoxmessage;
use Illuminate\Http\Request;
use App\Models\campagnemesure;
use App\Models\boitier;
use App\Models\Statistique;
```

Au début du programme on voit « namespace » c'est en quelque sorte un moyen d'encapsuler des éléments. Puis on y voit plusieurs fois l'opérateur use qui permet d'importer ici par exemple les modèles.

```
class SigfoxMessageController extends Controller
```

La classe SigfoxMessageController est créée et comme la classe est étendu, elle hérite de toutes les méthodes de Controller. L'héritage est très utile pour définir et abstraire certaines fonctionnalités communes à plusieurs classes, tout en permettant la mise en place de fonctionnalités supplémentaires dans les classes enfants, sans avoir à réimplémenter en leur sein toutes les fonctionnalités communes. Par contre les méthodes privées de Controller ne sont pas accessible à la classe SigfoxMessageController. Par conséquent, la classe SigfoxMessageController peut réimplémenter une méthode privé elle-même sans se soucier des règles d'héritage normale.

```
public function index()
{
    return sigfoxmessage::all();
```

Ici, on a une déclaration d'une fonction publique nommée index, elle pourra donc être appelée partout. Elle doit donc retourner tout les messages sigfox.

```
public function store(Request $request)
```

On a ensuite la création de notre fonction principale nommée store avec request comme paramètre donc des requêtes. Cette fonction à pour but de stocker une ressource nouvellement créée dans le stockage.

```
if (!$request->input('device')) {
    return response()->json('ERREUR : champ device vide!', 500);
}
```

Ici, si on a pas rempli le champ « device » qui est en fait l'identifiant de l'appareil, on nous retourne une réponse d'erreur 500 qui signifie que le champs « device » est vide.

```

$sigfoxMessage = sigfoxmessage::create([
    'device' => $request->input('device'),
    'data'    => $request->input('data'),
    'time'   => $request->input('time'),
]);

```

Ensuite on a la création d'un message sigfox qui est composé de : « device » (identifiant de l'appareil), « data » (qui est la trame des données), et « time » (qui correspond à l'heure et à la date d'envoi)

```

$binData = "";

$info = $sigfoxMessage->data;

```

On passe ici à l'initialisation de binData puis on met dans info le sigfoxMessage qui correspond en réalité au data.

```

if(strlen($sigfoxMessage->data) == 1)
{
    $boitier = boitier::where('adrSigfox', $sigfoxMessage->device)->first();

    if($sigfoxMessage->data == 1)
    {
        $info2 = $boitier->adrSigfox;

        $boitier->alarmeBatterie = $sigfoxMessage->data;
        $boitier->update();

        $info = "Problèmes Batterie";

        // laisser place Armand envoie mail quand batterie est faible
        $users = User::all();

        foreach($users as $users)
        {
            if($users->role_id == 1 || $users->role_id == 2)
            {
                Mail::to($users)->bcc("kolantr2021snir@gmail.com")
                    ->queue(new MessageGoogle('Une batterie est faible : ' . $info2));
            }
        }
    }

    return (array(['info' => $info, 'info2' => $info2]));
}

}elseif($sigfoxMessage->data == 0)
{
    $info2 = $boitier->adrSigfox;

    $boitier->alarmeBatterie = NULL;
    $boitier->update();

    $info = "Batterie rempli";
    return (array(['info' => $info, 'info2' => $info2]));
}
} else {

```

Cette partie concerne le boîtier, on y voit le calcul de la taille de la trame avec l'utilisation de strlen. Si la valeur est de 1 alors il y a un problème de batterie, sinon, si la valeur est de 0 alors la batterie est remplie.

```
if( strlen($sigfoxMessage->data) > 10){
    for ($i = 0; $i < strlen($sigfoxMessage->data); $i++) {

        $quartet = base_convert($sigfoxMessage->data[$i], 16, 2);
        $quartet = str_pad($quartet, 4, "0", STR_PAD_LEFT);
        $binData .= $quartet;
    }

    $info2 = $binData;
```

Toujours dans le SigfoxMessageController.php, si la trame est supérieur à 10 octets, on convertit chaque chiffre hexadécimal en un quartet binaire, on complète la chaîne jusqu'à un format fixe en 4 chiffres binaires, puis on concatène au résultat. Pour la dernière ligne on met donc les données binaires dans info2.

```
$typeV = extraitChampBinaire($binData, 0, 1);
$NbVehicule = extraitChampBinaire($binData, 1, 11);
$NbEssieu = extraitChampBinaire($binData, 12, 13);
$VitMoyenne = extraitChampBinaire($binData, 25, 8);
$VitMax = extraitChampBinaire($binData, 33, 8);
$VitesseInferieureOuEgale = extraitChampBinaire($binData, 41, 11);
$VitesseLimitMoins20 = extraitChampBinaire($binData, 52, 11);
$VitesseLimitPlus20 = extraitChampBinaire($binData, 63, 10);
$VitesseLimitPlus30 = extraitChampBinaire($binData, 73, 9);
$VitesseLimitPlus40 = extraitChampBinaire($binData, 82, 7);
$VitesseLimitPlus50 = extraitChampBinaire($binData, 89, 7); //96
```

Ici, on procède à l'extraction des données binaires donc on peut voir que par exemple pour le typeV, il utilise 1 bit, il commence à 0, que pour le NbVehicule il utilise 11 bits et commence à 1, etc. jusqu'à 96 bits qui correspondant donc à la longueur totale de la trame data.

```
$typeV = dechex($typeV);
$NbVehicule = dechex($NbVehicule);
$NbEssieu = dechex($NbEssieu);
$VitMoyenne = dechex($VitMoyenne);
$VitMax = dechex($VitMax);
$VitesseInferieureOuEgale = dechex($VitesseInferieureOuEgale);
$VitesseLimitMoins20 = dechex($VitesseLimitMoins20);
$VitesseLimitPlus20 = dechex($VitesseLimitPlus20);
$VitesseLimitPlus30 = dechex($VitesseLimitPlus30);
$VitesseLimitPlus40 = dechex($VitesseLimitPlus40);
$VitesseLimitPlus50 = dechex($VitesseLimitPlus50);
```

La fonction dechex permet la conversion décimal en hexadécimal. On convertit les données hexadécimal en données décimales.

```
$boitier = boitier::where('adrSigfox', $sigfoxMessage->device)->first();
```

On va chercher dans la base de données le boîtier qui a l'adresse SigfoxMessage->device

```
if(empty($boitier))
{
    return response()->json('ERREUR : boitier inexistant', 500);
}
$campagne = campagnemesure::where('id_boitier', $boitier->id)->first();
if(empty($campagne))
{
    return response()->json('ERREUR : campagne inexistante', 500);
}

if($typeV == 0)
{
    $type_vehicule = "voiture";
} else {
    $type_vehicule = "camion";
}
```

Là on dit que si la valeur boîtier est vide alors on retourne une erreur de boîtier inexistant et pareil si la valeur campagne est vide alors on retourne une erreur de campagne inexistante.

On dit aussi que si la valeur de typeV est égale à 0 alors c'est une voiture sinon c'est un camion, donc camion à la valeur de 1.

```
$stat = Statistique::create([
    'campagneId' => $campagne->id,
    'typeV' => $type_vehicule,
    'timestamp' => date('Y-m-d H:i:s', $sigfoxMessage->time),
    'VitMax' => $VitMax,
    'NbEssieux' => $NbEssieu,
    'VitMoyenne' => $VitMoyenne,
    'NbVehicule' => $NbVehicule,
    'VitesseInferieureOuEgale' => $VitesseInferieureOuEgale,
    'VitesseLimitMoins20' => $VitesseLimitMoins20,
    'VitesseLimitPlus20' => $VitesseLimitPlus20,
    'VitesseLimitPlus30' => $VitesseLimitPlus30,
    'VitesseLimitPlus40' => $VitesseLimitPlus40,
    'VitesseLimitPlus50' => $VitesseLimitPlus50,
]);

return (array(['info' => $info, 'info2' => $info2]));
```

Ici on peut voir la création du relevé puis que ça retourne les informations qui sont donc celles du boîtier et celles de la trame des données.

```

for ($i = 0; $i < strlen($sigfoxMessage->data); $i++) {

    $quartet = base_convert($sigfoxMessage->data[$i], 16, 2);
    $quartet = str_pad($quartet, 4, "0", STR_PAD_LEFT);
    $binData .= $quartet;
}

$info = $binData;

```

Donc ici, comme précédemment, mais cette fois-ci avec le boîtier, on calcul la longueur de la trame, on convertit chaque chiffre hexadécimal en un quartet binaire, on crée un format fixe en 4 chiffres binaires, on concatène le résultat puis on met les données dans la variable info.

```

$NbEssieu = extraitChampBinaire($binData, 0, 13);

$NbEssieu = dechex($NbEssieu);

$boitier = boitier::where('adrSigfox', $sigfoxMessage->device)->first();

if(empty($boitier))
{
    return response()->json('ERREUR : boitier inexistant', 500);
}
$campagne = campagnemesure::where('id_boitier', $boitier->id)->first();
if(empty($campagne))
{
    return response()->json('ERREUR : campagne inexisteante', 500);
}

$stat = Statistique::create([
    'campagneId' => $campagne->id,
    'timestamp' => date('Y-m-d H:i:s', $sigfoxMessage->time),
    'NbEssieux' => $NbEssieu,
]);

return (array(['info' => $info]));

```

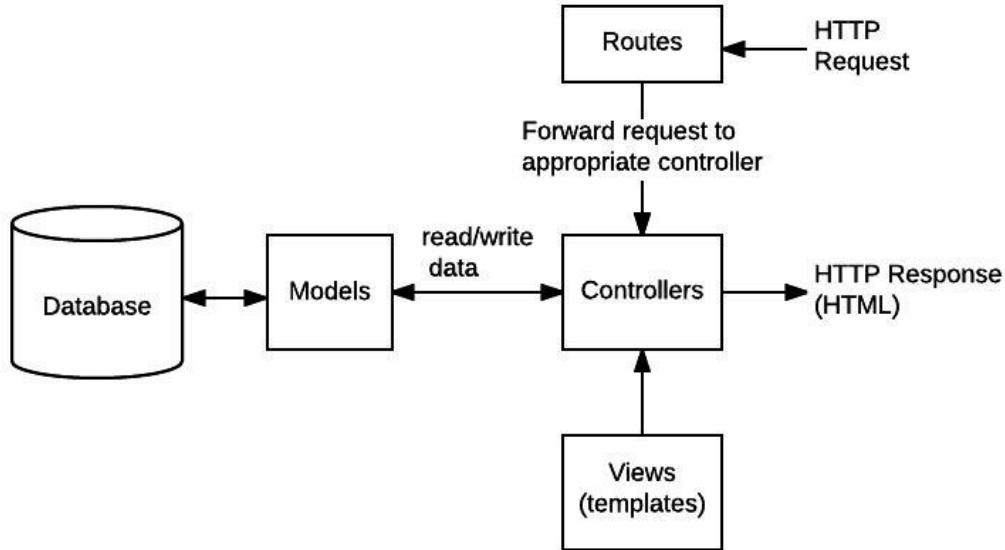
Ici, c'est pour la variable NbEssieu, donc on extrait les données binaires, on fait ensuite une conversion comme précédemment. On finit ensuite par une création de relevés et on retourne la valeur de la variable info.

```
function extraitChampBinaire($chaine, $index, $taille)
{
    $bin = substr($chaine, $index, $taille);
    if (strpos($bin, "0") === false) {

        $val = null;
    } else {
        $val = base_convert($bin, 2, 10);
    }
    return $val;
}
```

En bas de la page, la fonction extraitChampBinaire est créée et donc on y voit par exemple que si aucun zéro n'est trouvé alors tout les bits passent à 1 donc NULL.

Concernant la route, elle est associé à un autre contrôleur. Voici le fonctionnement général des routes :



Il est parfois utile de nommer une route, par exemple pour générer une URL ou pour effectuer une redirection. La syntaxe pour nommer une route est celle-ci :

```
Route::get('/Statistiques', [
    'as'=>'Statistiques',
    'uses'=>'App\Http\Controllers\CollecteSigfoxController@index'
]);
```

```
class CollecteSigfoxController extends Controller
{
    public function index()
    {
        $Stats = Statistique::all();
        return view('pages/Statistiques')->with('Stats', $Stats);
    }
}
```

La première capture d'écran viens du fichier web.php et la seconde du CollecteSigfoxController.php. Sur les deux on y retrouve la route (/Statistiques) et le contrôleur (CollecteSigfoxController). Le contrôleur traite la demande de l'utilisateur et la page Statistiques est donc retourné à l'utilisateur.

D. Tests

Au niveau des tests, des tests d'intégrations ont permis de valider les tests unitaires puisqu'en effet, pour ma partie, il est nécessaire d'envoyer une trame afin de déterminer si mon programme est valide. Ci-dessous on retrouve un de ces tests que je vais détailler.

```
envoi message Sigfox : 355388283830002609044809
commande modem : AT$SF=355388283830002609044809
réponse modem reçue : OK

Message Sigfox envoyé
```

Ici on peut voir l'envoie d'une trame de la partie collecte, en effet Hugo Carricart envoie la trame depuis le modem Sigfox. La trame est en hexadécimal afin de pouvoir faire passer toutes les informations et aussi parce que le modem ne peut envoyer que de l'hexadécimal.



Ensuite, comme le montre cette capture, la trame est bien envoyée au simulateur Sigfox. On y voit l'id du device (2ED518), la date et l'heure (27 mai 2021 8h55min36s), la trame des données (355388283830002609044809) qui est encore en hexadécimal, le niveau de connexion qui est excellent (le logo du signal en vert l'indique) ainsi que la callback (logo avec la flèche qui pointe vers le haut et qui est vert puisque la callback est fonctionnelle).

Une fois la trame envoyé au simulateur Sigfox, elle passe par le contrôleur puis est stockée dans une base de données comme le montre l'image ci-dessous.

VitesseLimitPlus30	VitesseLimitPlus40	VitesseLimitPlus50	campagnId	created_at	updated_at
11	10	9	4	2021-05-27 06:55:37	2021-05-27 06:55:37

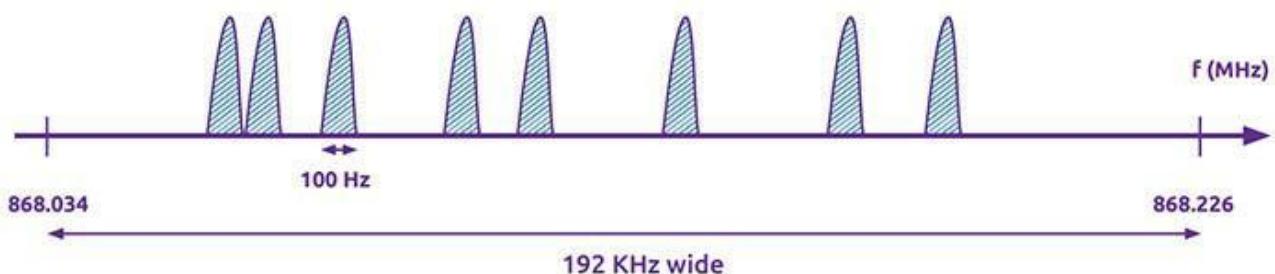
On peut voir que le contrôleur a agit sur la trame puisqu'on y retrouve les différentes informations qui sont donc les données de la trames qui ont été convertis en décimales. Pour le type de véhicule on voit que le contrôleur a bien affecté la bonne valeur puisque on voit écrit « voiture » et non 0 puisque en decimal on aurait trouvé 0 ce qui correspond à voiture.

E. Partie physique

Sigfox utilise la modulation radio à bande ultra-étroite (UNB, Ultra Narrow Band). La bande ultra-étroite offre une grande capacité de canaux, de longues distances de transmission et une haute résistance au bruit.

Sigfox utilise la technologie radio UNB à 868 MHz en Europe et au Moyen-Orient, et 902 MHz en Amérique du Nord. Ces bandes sont disponibles publiquement. Sa bande passante est de 192 kHz. L'UNB génère un signal radio relativement étroit. Par conséquent, un plus grand nombre de signaux peuvent être diffusés simultanément. Chaque message utilise une fréquence de 100 Hz et est transmis à 100 ou 600 bits par seconde, en fonction de la région. Les choix de modulation et de protocole de Sigfox ne sont pas optimisés pour une vitesse élevée, mais ils sont très efficaces en matière de bilan de liaison. Le bilan de liaison élevé de Sigfox offre une longue distance de communication. L'UNB utilise également de plus petites trames radio, ce qui permet d'augmenter la capacité de données et de réduire la consommation énergétique par rapport à un protocole classique avec un débit plus élevé. Cette technologie favorise les longues distances et fait preuve d'une haute résistance au bruit. Avec la modulation D-BPSK (modulation par déplacement de phase binaire), elle transmet seulement 1 Hz à 1 bps. Un signal unique occupe donc une part minimale de la bande de communication.

La modulation D-BPSK est utilisée pour trois raisons : Elle est facile à mettre en place ; le faible débit permet d'utiliser des éléments peu coûteux ; la puissance du signal de Sigfox étant concentrée sur une bande passante étroite, les stations de base peuvent être très sensibles et peuvent démoduler les signaux à proximité du bruit de fond ; cette technologie améliore la résistance aux interférences. La sensibilité de la station de base peut être de -142 dBm à 100 bps et de -134 dBm à 600 bps.



Spectre de fréquences UNB de Sigfox

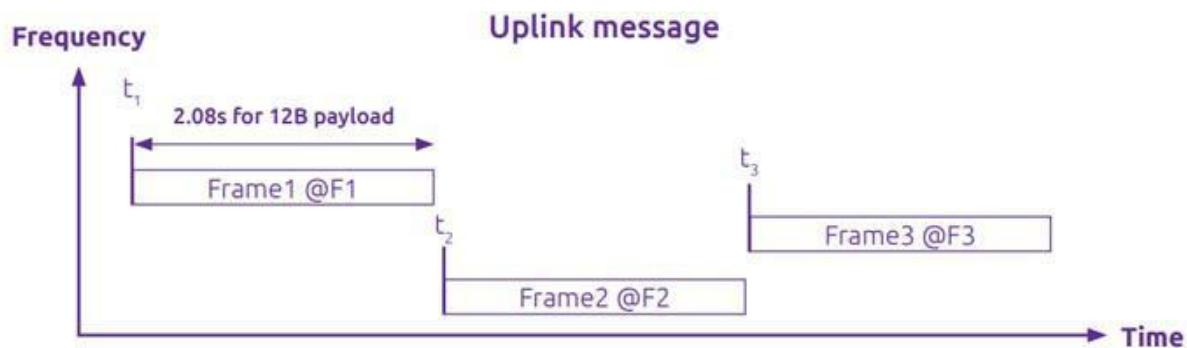
Sigfox est conçu pour envoyer des messages peu conséquents avec un débit minimal. Il réduit ainsi la consommation énergétique et allonge la durée de vie de la batterie. Le protocole de systèmes classiques est optimisé pour transporter un grand volume de données, mais il s'avère généralement inefficace. Une trame Sigfox transporte 12 octets de données et un maximum de 26 octets, surdébit compris.

Ce protocole léger offre deux principaux avantages puisque chaque octet de données et de surdébit consomme de l'énergie, l'utilisation de ces trames efficaces optimise la durée de vie de la batterie des dispositifs finaux et une réduction du surdébit signifie une plus grande capacité pour les données utilisateur.

Un message en liaison montante contient jusqu'à 12 octets de charge utile et prend en moyenne 2 s pour atteindre les stations de base qui contrôlent le spectre cherchant à démoduler des signaux UNB. Pour une charge utile de 12 octets, une trame Sigfox utilise 26 octets au total.

Un dispositif peut transmettre jusqu'à 140 messages par jour. La charge utile autorisée avec des messages en liaison descendante est de 8 octets, avec 4 messages autorisés par jour.

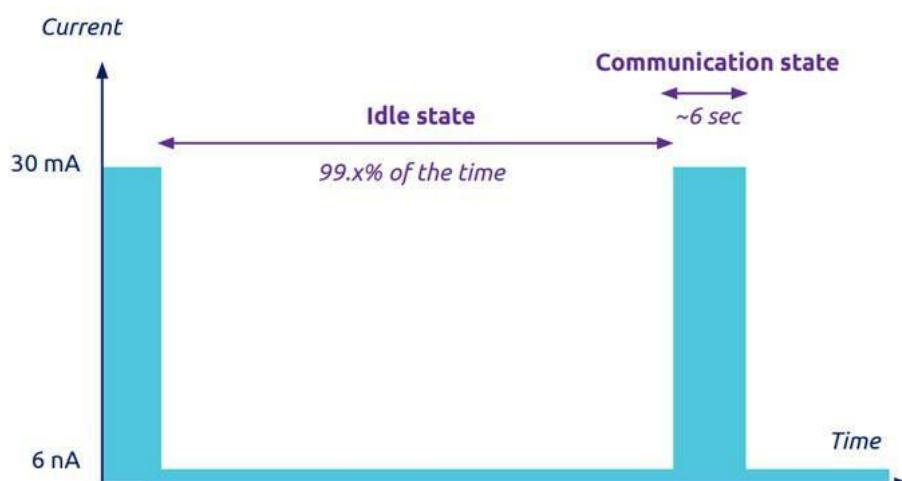
Avec l'accès aléatoire, il n'est plus utile de dépenser du débit supplémentaire et de gaspiller de l'énergie en synchronisant les dispositifs avec le réseau. Le dispositif transmet un message sur une fréquence aléatoire, puis envoie deux répliques sur d'autres fréquences à des moments différents : un mode appelé « diversité temporelle et fréquentielle ».



Transmission en liaison ascendante et saut de fréquence

Réception coopérative : contrairement aux protocoles cellulaires, les dispositifs ne sont pas liés à des stations de base spécifiques. Le message émis est reçu par n'importe quelles stations de base (trois en moyenne) à proximité. C'est ce que l'on appelle la « diversité spatiale ».

Faible courant de ralenti : les dispositifs sont inactifs plus de 99 % du temps, et le courant de ralenti est très faible, souvent de quelques nanoampères seulement.



Un faible courant de ralenti signifie une augmentation de la durée de vie de la batterie

Le Cloud Sigfox prend en charge tous les services Sigfox. Pour récupérer les messages, mais également gérer tous les objets et pièces utilisateur, les clients et partenaires de Sigfox interagissent directement avec le Cloud Sigfox. L'accès aux données, la facturation, la gestion des utilisateurs et des dispositifs, les cartes de service et les autres fonctions sur le Cloud Sigfox sont accessibles par le biais de trois interfaces : le portail Web, disponible via un simple navigateur, qui permet aux utilisateurs finaux d'accéder à toutes les fonctions du Cloud ; l'API, qui, à l'aide de scripts, permet un accès automatisé à toutes les fonctions des services disponibles sur le portail ; les rappels, qui permettent de recevoir automatiquement les nouveaux événements (cette fonctionnalité supprime le surdébit coûteux qui cherche en permanence à recevoir des messages).

L'accès aux services du Cloud Sigfox et à sa suite d'outils dépend du profil de l'utilisateur : un administrateur peut gérer les utilisateurs et l'ensemble des dispositifs, consulter les cartes de service et vérifier les contrats ; un distributeur peut créer des contrats ; un opérateur peut accéder à l'outil de planification radio et à l'ensemble de la surveillance du réseau ; un utilisateur peut être limité à la visualisation des messages uniquement.

Sigfox peut couvrir des distances allant de 30 à 50 km dans les environnements ruraux, et de 3 à 10 km dans les zones urbaines. La longue portée de Sigfox est due au faible débit de données, à la puissance de sortie des dispositifs et à la sensibilité des stations de base. L'utilisation d'une bande sub-GHz offre également une bonne couverture en intérieur comparée aux réseaux de 2,4 GHz.

Sigfox couvre près de 40 pays à l'aide de son propre réseau ou, plus souvent, de réseaux partenaires.

Les stations de base sont plus simples à déployer et à intégrer avec le reste du réseau actuel à faible consommation et n'impliquent aucune difficulté de configuration. Cette méthode est très différente de l'installation d'une cellule mobile traditionnelle.

Sigfox met en place des mesures de sécurité au niveau de l'écosystème et des dispositifs.

L'écosystème met en pratique la sécurité par défaut :

- Authentification + intégrité + anti-relecture des messages propagés sur le réseau
- Cryptographie
- Chiffrement de la charge utile en option pour assurer la confidentialité des données

Chaque partie du réseau est isolée et les risques sont évalués, de sorte qu'en cas de piratage, seule une petite partie du réseau est affectée

Pour les dispositifs, Sigfox a défini trois niveaux de sécurité. Le fournisseur d'application peut choisir le niveau le plus adapté au projet et à sa sensibilité.

- Niveau moyen : les informations d'identification de sécurité sont stockées dans le dispositif
- Niveau élevé : les informations d'identification de sécurité sont stockées dans une zone protégée basée sur logiciel
- Niveau très élevé : les informations d'identification de sécurité sont stockées dans un environnement sécurisé

Les mesures de sécurité permettent également de chiffrer les données transférées sur le réseau.

Seuls le dispositif et le client final connaissent le code secret. L'algorithme n'a pas d'incidence sur le volume de la charge utile. Si le message est chiffré, la charge utile fera toujours 12 octets.

F. Bilan personnel technique

Tâches accomplies :

- Mise en œuvre d'un web service
- Création d'un contrôleur
- Création de 2 méthodes
- Élaboration du modèle de données
- Création des tables
- Création des modèles
- Ajout des BREAD
- Ajout des relations

Tâches restantes :

- Être avertis si une trame est fausse
- Être avertis du niveau de batterie

Intégration :

Concernant l'intégration, j'arrive à recevoir les trames de Hugo qui fait partie du sous-système de collecte et les callbacks sont fonctionnelles, je reçois bien les informations dans ma base de données.

VII. Étudiant Armand HOUPIN

A. Description partie personnelle

Sur la sous-partie de consultation, l'utilisateur doit pouvoir accéder au site web du projet. Il y a 3 acteurs qui peuvent interagir avec le site web, l'administrateur, le technicien et le client.

L'administrateur : Il peut gérer les comptes du site web (créer, modifier, supprimer), il peut aussi gérer les campagnes de mesures et gérer les boîtiers.

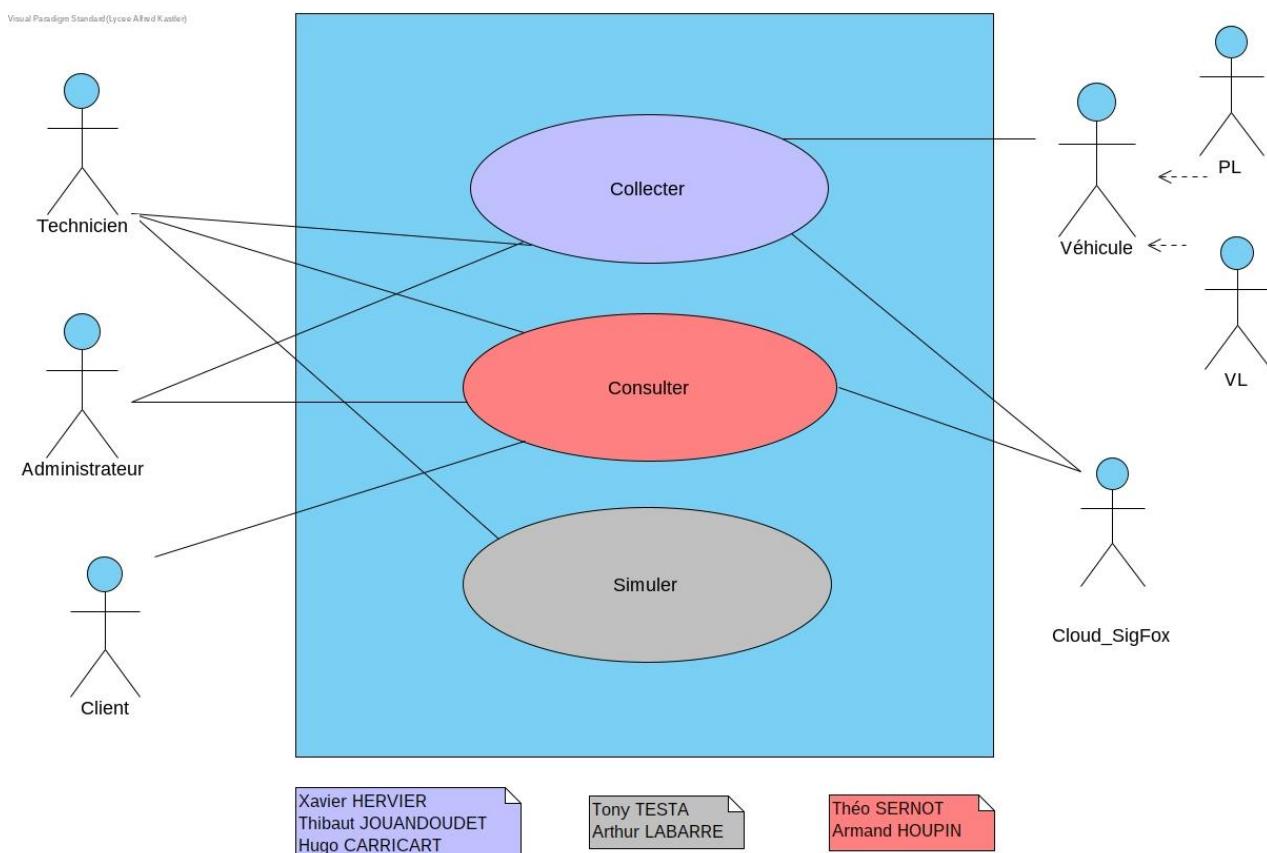
Le technicien : Il peut gérer les campagnes de mesures et gérer les boîtiers.

Le client : Il peut visualiser sa campagne de mesure, avec les statistiques toutes les une heures, visualisation en graphique et il peut télécharger les statistiques en .CSV (format tableur)

En plus, les administrateurs et les techniciens reçoivent des e-mails quand une batterie d'un système de collecte est vide.

Le client reçoit un e-mail quand son compte est créé.

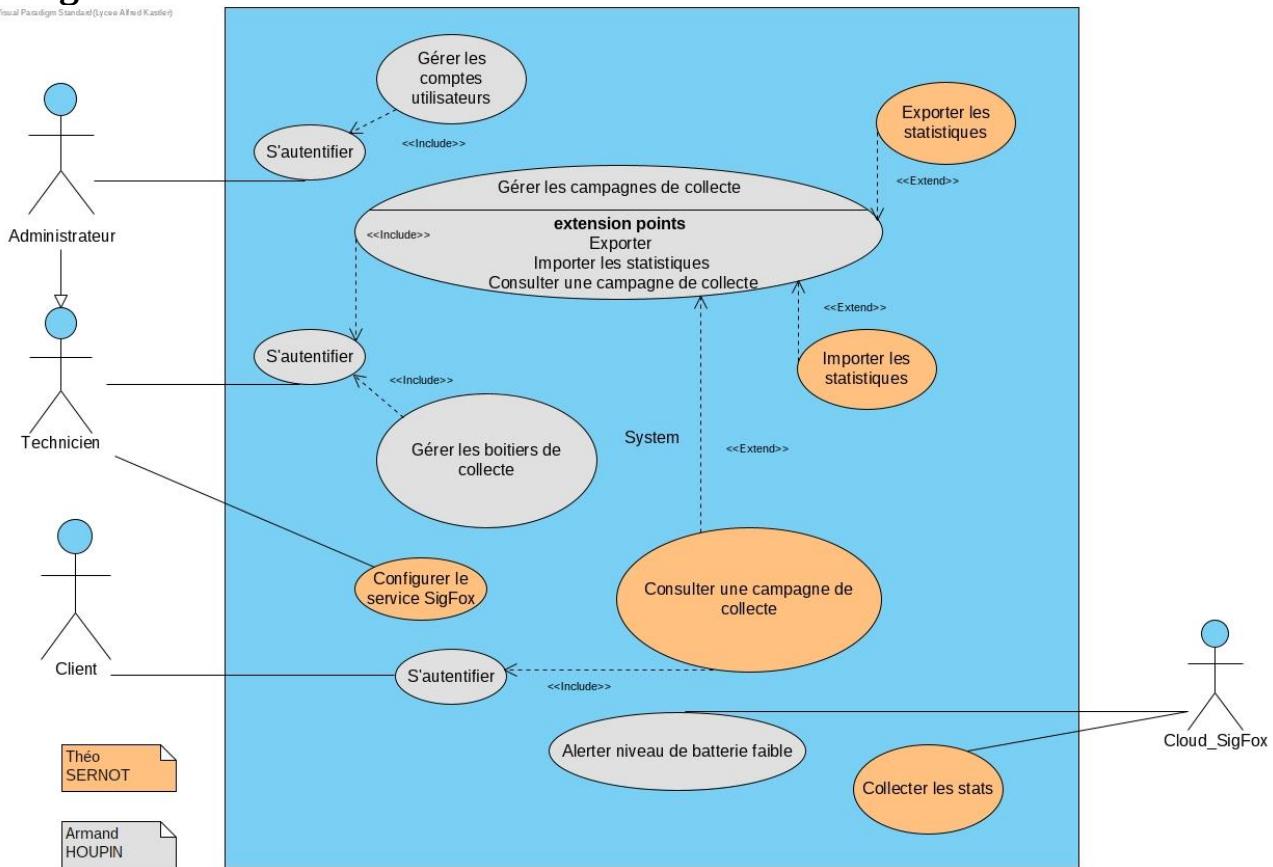
B. Conception détaillée



Ma partie est celle en Rouge « Consulter »

Diagramme de cas d'utilisation

Visual Paradigm Standard (Lycée Alfred Kastler)



Gérer les boîtiers de collecte

Scenario (Créer boitier)

Steps	Procedures	Expected Results
1. Remplir le formulaire		
2. Cliquez sur [creer un boitier] ou [modifier un boitier]	Cliquez sur [créer boitier]	Créer le boitier et renvoie sur la liste des boitiers

Scenario (modifier boitier)

Steps	Procedures	Expected Results
1. Remplir le formulaire		
2. Cliquez sur [creer un boitier] ou [modifier un boitier]	Cliquez sur [modifier boitier]	Modifie le boitier et renvoie sur la liste des boitiers

Gérer les campagnes de collecte

Scenario (Créer une campagne)

Steps	Procedures	Expected Results
1. Remplir le formulaire	Remplir le formulaire	

Steps	Procedures	Expected Results
2. Cliquez sur [Créer une campagne] pour créer la campagne	Cliquez sur [Créer une campagne de mesure]	Crée la campagne et renvoie sur la liste de la campagne

Scenario (Modifier une campagne)

Steps	Procedures	Expected Results
1. Remplir le formulaire		
2. Cliquez sur [Créer une campagne] pour créer la campagne	Cliquez sur [modifier la campagne]	Modifie la campagne et renvoie sur la liste des campagnes

Gérer les comptes utilisateurs

Scenario (créer compte client)

Steps	Procedures	Expected Results
1. Remplir le formulaire	Remplir le formulaire	
2. Cliquez sur [créer le compte] pour modifier le compte client	Cliquez sur [Crée un compte]	Créer un compte client avec ces informations et renvoie sur la page qui liste les comptes clients

Scenario (modifier compte client)

Steps	Procedures	Expected Results
1. Remplir le formulaire		
2. Cliquez sur [créer le compte] pour modifier le compte client	Cliquez sur [modifier le compte]	Modifie le compte avec les nouvelles informations et renvoie sur la liste des comptes clients

Scenario (adresse mail déjà existante)

Steps	Procedures	Expected Results
1. Remplir le formulaire	remplir une adresse mail déjà existante dans la bdd	
2. Cliquez sur [créer le compte] pour modifier le compte client	Cliquez sur [ajouter un compte] ou [modifier le compte]	renvoie une erreur qui nous indique "Adresse mail déjà existante" et nous renvoie sur la page d'ajout ou de modification du client

S'autentifier

Scenario (login avec un identifiant et un mot de passe correct)

Steps	Procedures	Expected Results
1. Entrer identifiant	Remplir adminKolantr2021 comme identifiant	
2. Entrer Mot de passe	Remplir password comme mot de passe	Huit caractères astérisqu (c'est-à-dire *) sont affichés
3. Cliquez sur [connecter] pour ouvrir la page d'Accueil	Cliquez sur [connecter]	La page d'accueil de l'utilisateur s'affiche

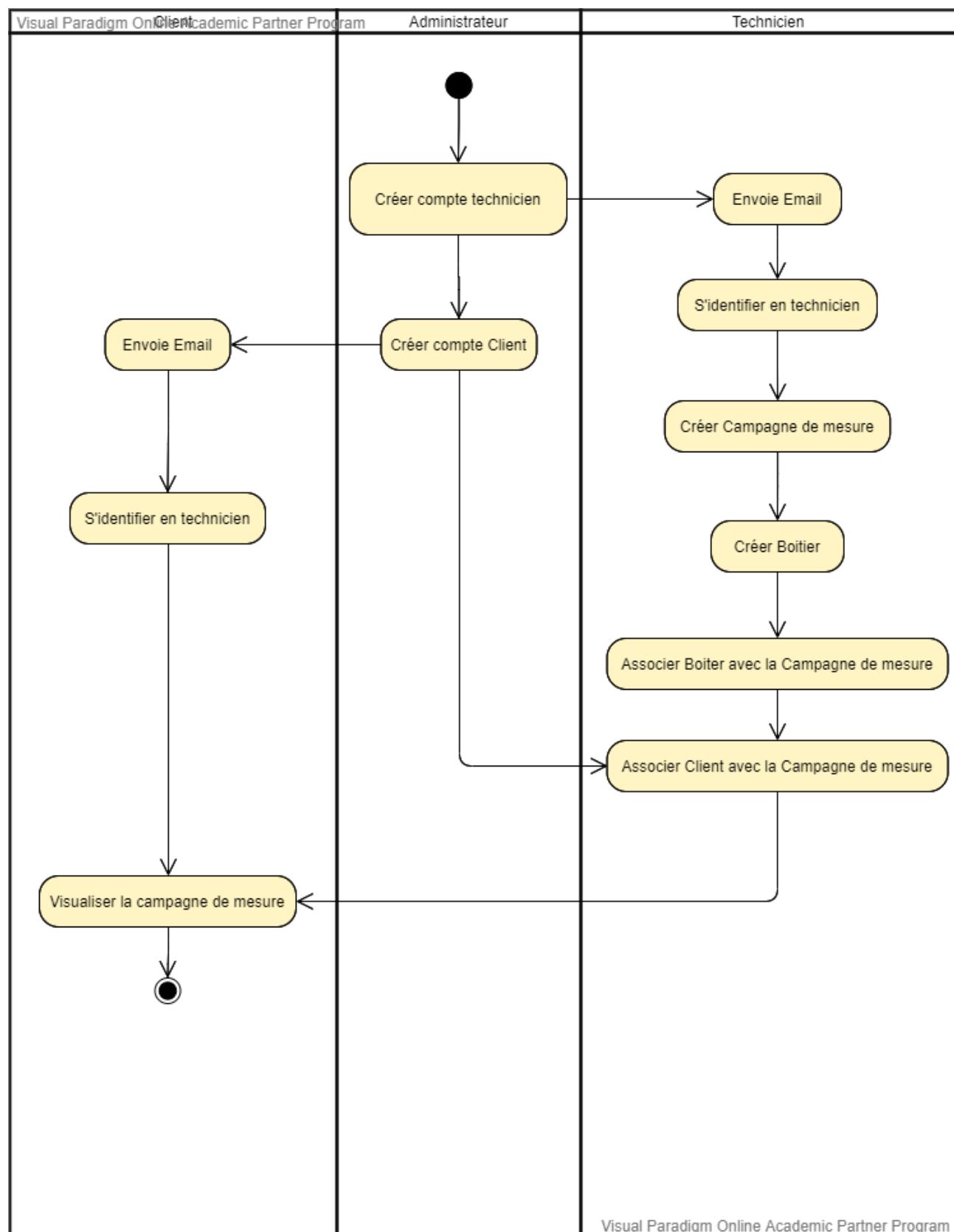
Scenario (connexion sans identifiant et mot de passe)

Steps	Procedures	Expected Results
1. Entrer identifiant	N'entrez rien	
2. Entrer Mot de passe	N'entrez rien	
3. Cliquez sur [connecter] pour ouvrir la page d'Accueil	Cliquez sur [connecter]	Attention "Veuillez spécifier l'identifiant et le mot de passe" apparaît. Connexion abandonnée

Scenario (connexion avec identifiant et mot de passe incorrect)

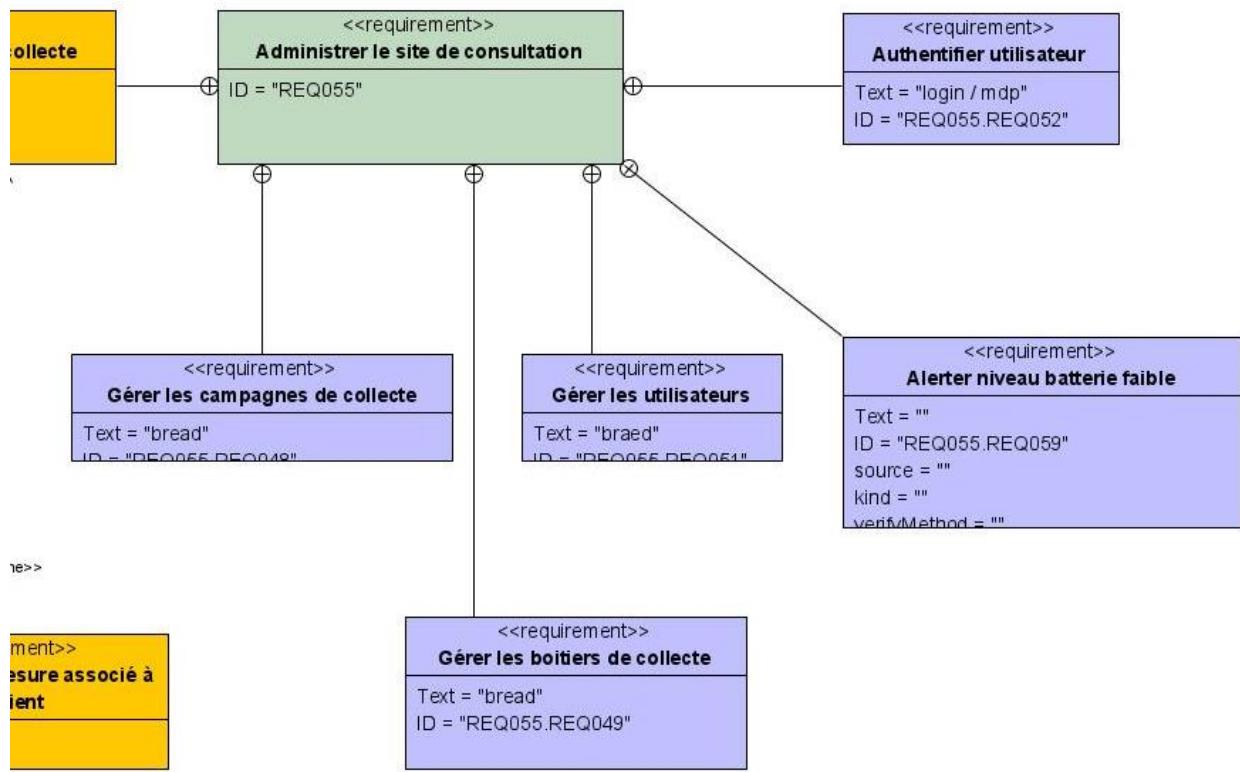
Steps	Procedures	Expected Results
1. Entrer identifiant	Entrez un identifiant incorrect	
2. Entrer Mot de passe	Entrez un mot de passe incorrect	Caractères astérisques sont affichés
3. Cliquez sur [connecter] pour ouvrir la page d'Accueil	Cliquez sur [connecter]	Attention "L'identifiant et le mot de passe est incorrect" apparaît. Connexion abandonnée

Diagramme d'activité : De la création d'un compte technicien à la consultation d'une campagne de mesure pour le client



Visual Paradigm Online Academic Partner Program

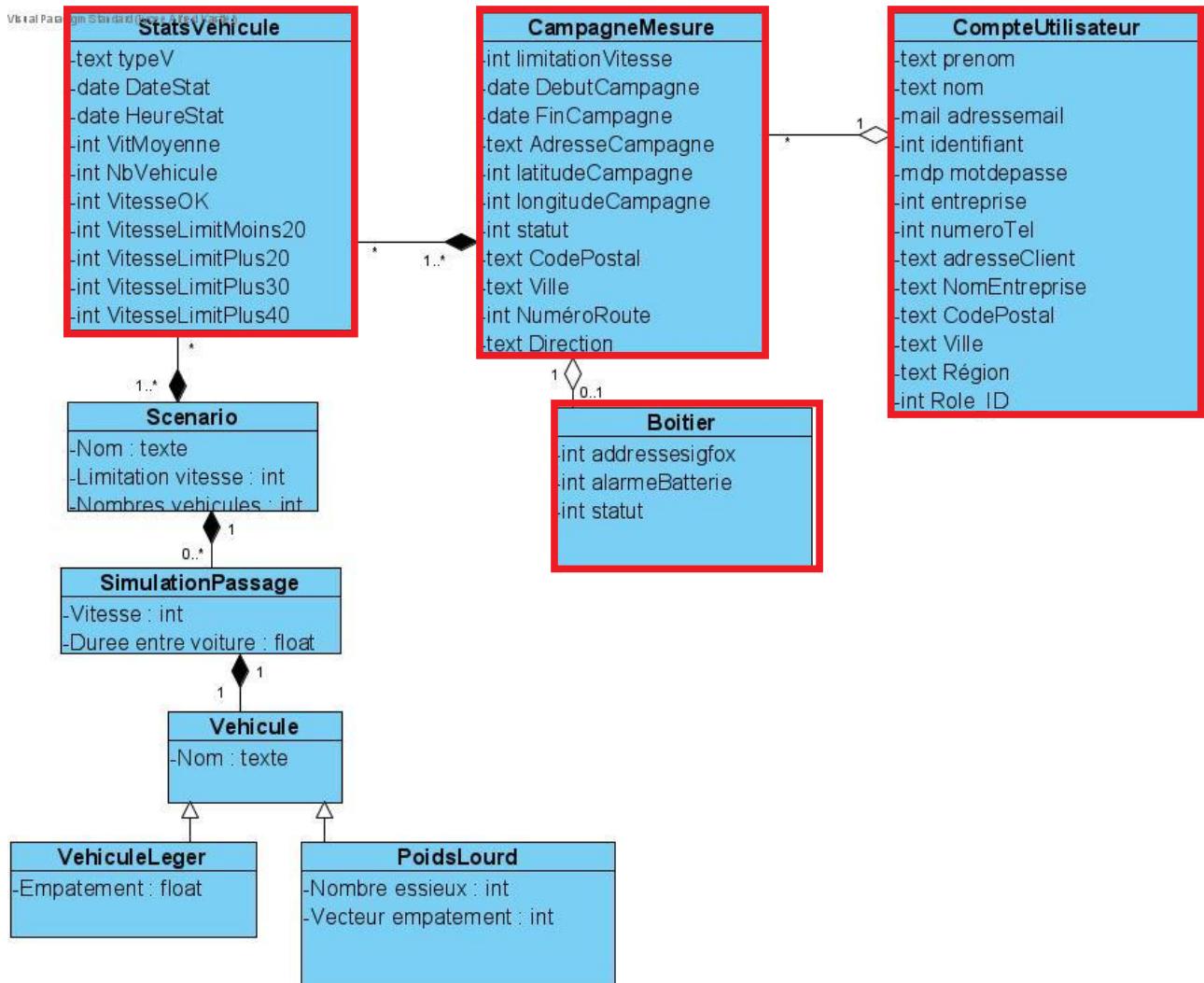
Diagramme d'exigence



Exigence liée à ma partie :

- Gérer les campagnes de collecte
- Gérer les utilisateurs
- Gérer les boîtiers de collecte
- Alerter niveau batterie faible
- Authentifier utilisateur
- Administrer le site de consultation

Modèle de données



Ma partie consultation

C. Implémentation

Logiciels utilisés pour le projet

Pour commencer le projet, il faut s'assurer d'avoir les prérequis d'installer sur l'ordinateur

Prérequis (installer sur l'ordinateur) :

- SQL
- Apache
- PHP

Présentation des logiciels utilisés :

LARAVEL

Laravel est un framework web open-source écrit en PHP respectant le principe modèle-vue-contrôleur et entièrement développé en programmation orientée objet. Laravel est distribué sous licence MIT, avec ses sources hébergées sur GitHub



MYSQL

MySQL est un système de gestion de bases de données relationnelles. Il est distribué sous une double licence GPL et propriétaire



VOYAGER

Voyager est un package d'administration Laravel qui comprend les opérations BREAD (CRUD), un gestionnaire de médias, un générateur de menu et bien plus encore.



VOYAGER

SQL

SQL est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.



APACHE

Le logiciel libre Apache HTTP Server est un serveur HTTP créé et maintenu au sein de la fondation Apache.



COMPOSER

Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.



ARTISAN

Artisan est l'interface de ligne de commande incluse avec Laravel. Artisan existe à la racine de votre application en tant que script artisan et fournit un certain nombre de commandes utiles qui peuvent vous aider lors de la création de votre application.

CREATION DE LA BASE DE DONNEES AVEC MYSQL (en ligne de commande)

Se connecter à MYSQL depuis un terminal

```
mysql -u root -p
```

Création de la base de données

```
CREATE DATABASE kolantr;
```

Création d'un nouveau User dans MYSQL

```
CREATE USER kolantr@localhost IDENTIFIED by 'mdpKOLANTR&2021';
```

Accorder tous les droits à l'utilisateur pour une base de données

```
GRANT ALL PRIVILEGES ON kolantr.* to kolantr@localhost;
```

CREATION DU PROJET LARAVEL (en ligne de commande dans un terminal)

CREER LE DOSSIER DU PROJET

```
composer create-project laravel/laravel kolantr
```

LANCER L'APPLICATION

```
cd kolantr  
php artisan serve
```

Pour accéder à l'application, taper dans votre navigateur

http://127.0.0.1:8000/

Pour associer la base de donnée au projet Laravel, il faut rentrer les informations dans le fichier « .env ». Ceci est le fichier de configuration de l'application

```
1 DB_CONNECTION=mysql  
2 DB_HOST=127.0.0.1  
3 DB_PORT=3306  
4 DB_DATABASE=kolantr  
5 DB_USERNAME=kolantr  
6 DB_PASSWORD=mdpKOLANTR&2021|
```

L'exemple ci-dessus montre où il faut rentrer les informations de la base de données.

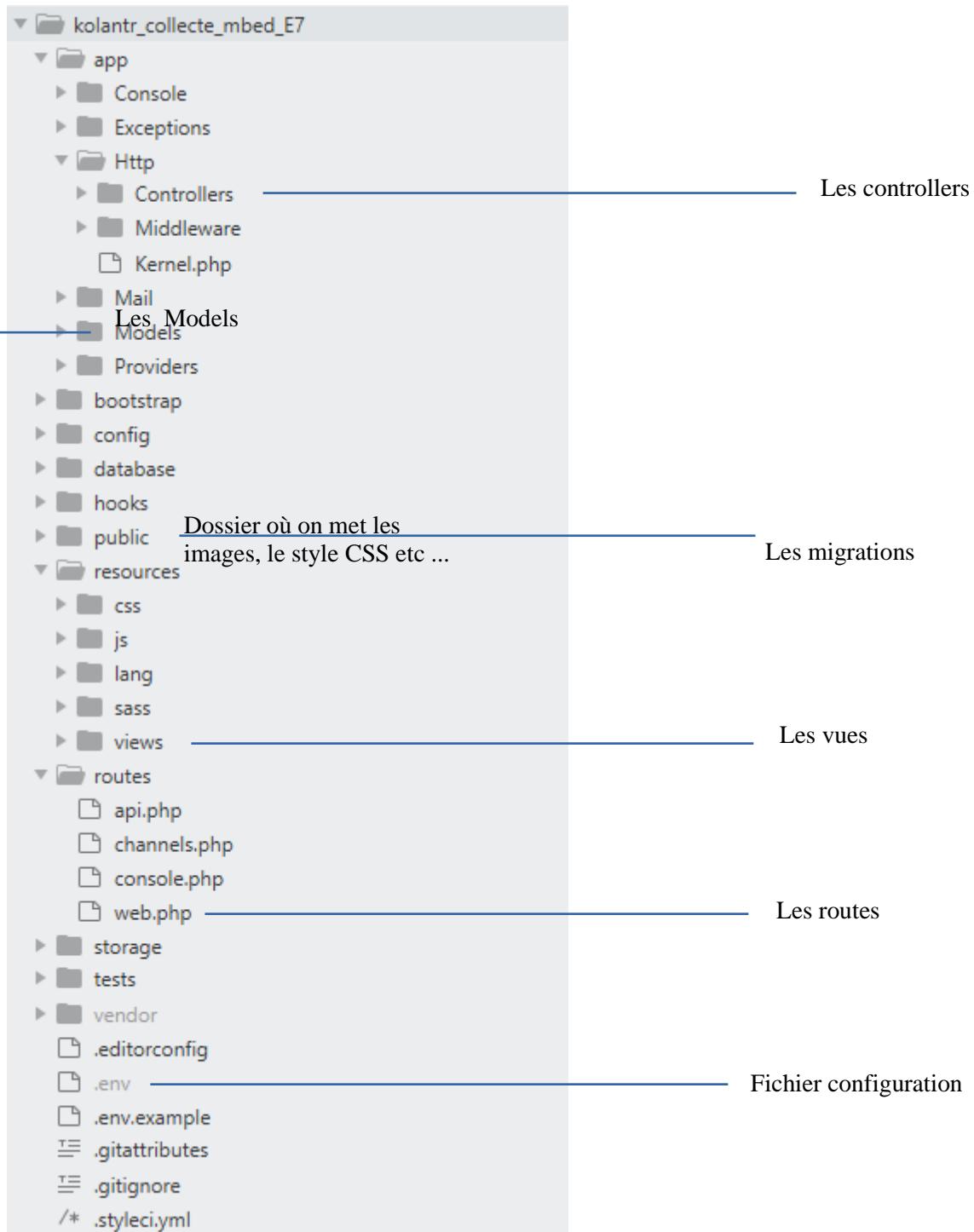
DB_DATABASE est le nom de la base

DB_USERNAME est le nom de la personne (doit rester secret)

DB_PASSWORD est le mot de passe pour se connecter (doit rester secret)

COMMENT FONCTIONNE LARAVEL ?

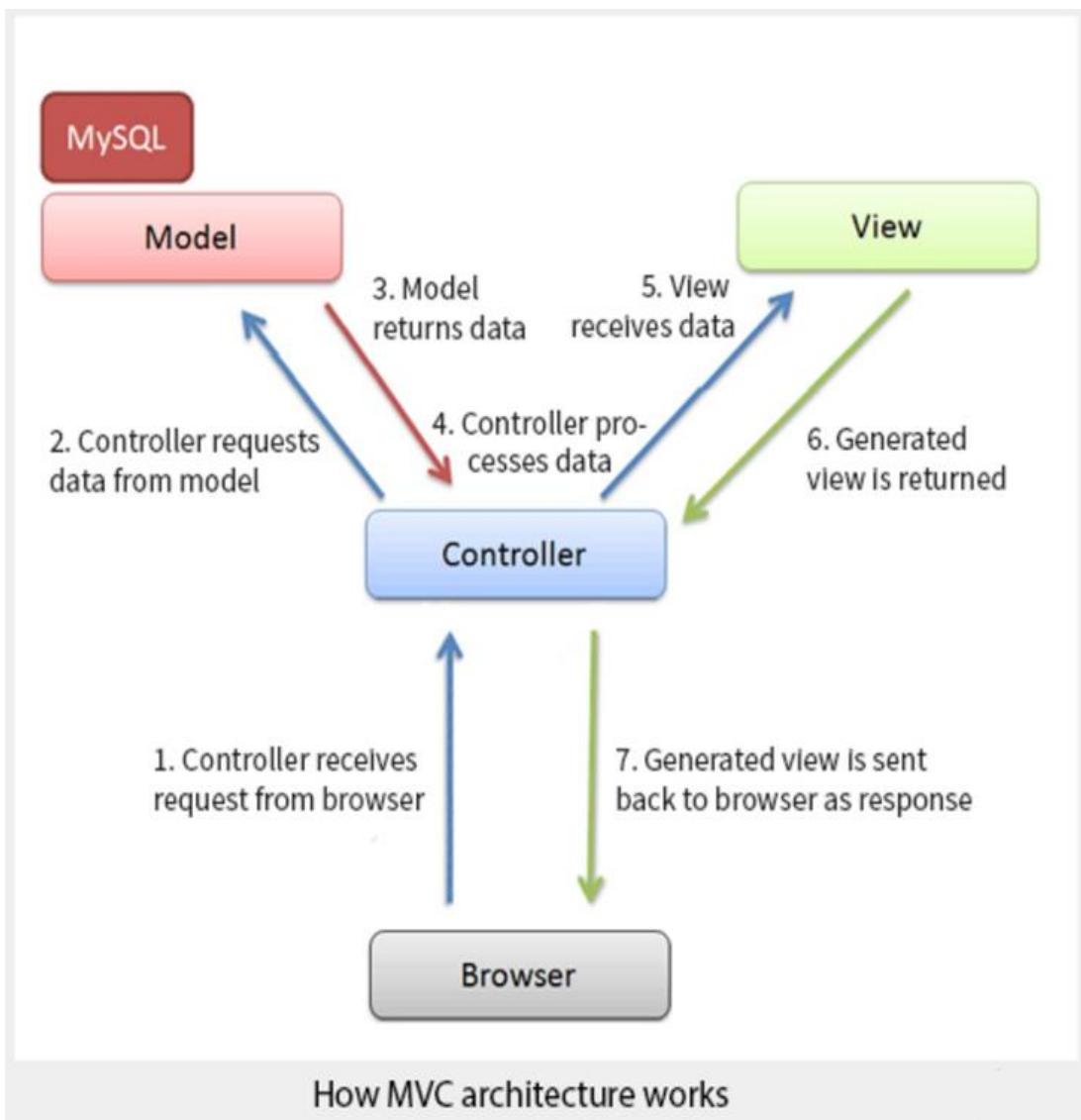
Explication de Laravel (arborescence)



Le fonctionnement de Laravel MVC (Models – Vues – Controllers)

Laravel est basée sur le système MVC, c'est à dire Models Vues Controllers.

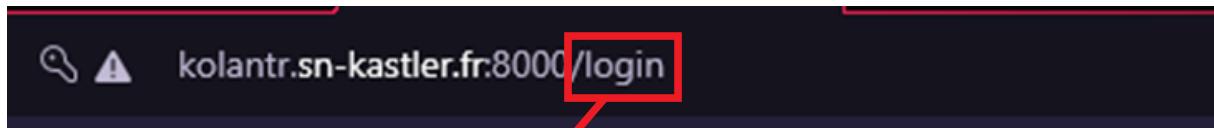
L'utilisateur arrive sur la page web, il se trouve au niveau du « Browser » sur le schéma au-dessus. Le « **browser** » ou plus précisément **L'URL**, est liée à une **route**. Cette **route** est associé à un **controller** et une **méthode**. Le controller peut être lié à un **models** (une table de la base de données) et à une **vues**. La **vue** c'est ce que l'utilisateur voit sur son écran.



Route et le controller de la route « /login »

Cela permet à un utilisateur de s'authentifier

Pour démontrer comment cela fonctionne, j'ai utilisé un exemple venant du projet



L'utilisateur à taper l'url :
http://kolantr.sn-kastler.fr:8000/login

Les routes se trouvent dans le fichier « web.php »

La page web est liée à la route « login »

```
Route::get('/login', [App\Http\Controllers\PagesController::class, 'login'])->name('login');
```

On voit que la route « /login » est liée à un controller, le controller « PagesController » (cadre vert) et à une méthode, la méthode « login » (cadre violet)

Dans PagesController,
on retrouve la
méthode « login »

```
class PagesController extends Controller
{
    public function login()
    {
        if (auth()->check()) {
            return redirect()->route('accueil');
        } else {
            return view('auth.login');
        }
    }
}
```

Un Controller sert à effectuer le traitement avant d'envoyer la vue à l'utilisateur.

Par exemple, dans cette méthode, il vérifie si l'utilisateur est connecté. Si il est connecté, il est redirigé sur la route « accueil » sinon il affiche la vue « auth.login » c'est à dire le fichier login dans le dossier auth.

Utilisation du Model « boitier » dans le controller

« gestionboitierController.php »

Ce controller permet gérer la liste des boîtiers, l'ajout d'un boîtier dans la base de données, de le modifier, de le supprimer. Dans cette exemple, on se trouve dans le controller « gestionboitierController.php »

Dans les controllers, on retrouve à chaque début du fichier, des liens vers des méthodes ou comme dans notre cas, un lien vers les models (cadre rouge)

Le Models sert à lier les attributs d'une table, celle en question, la table boîtier, pour faciliter le traitement avec les attributs pour ensuite les utiliser plus facilement lorsqu'on veut récupérer les données de la table

```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
use App\Models\User;  
use App\Models\campagnemesure;  
use App\Models\boitier;  
use App\Models\demande;
```

```
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
  
class boitier extends Model  
{  
    protected $fillable = [  
        'adrSigfox',  
        'alarmebatterie',  
        'statut'  
    ];  
}
```

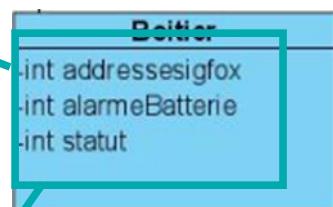
Dans la méthode « ajouterBoitier » on voit qu'on ajoute un nouveau « boitier » dans la table. On retrouve les attributs du Models

Quand le traitement est fini, il renvoie sur une vue ou sur une route pour effectuer à notre traitement. On retourne aussi des données à travers L'URL et à la vue (cadre vert)

Dans le Models « boitier » on retrouve les attributs :

- adrSigfox
- alarmebatterie
- statut

On retrouve les mêmes attributs dans modèle de données



```
public function ajouterBoitier()  
{  
    if (! auth()->check()) {  
        return redirect()->route('login');  
    }  
  
    $boitier = new boitier;  
    $boitier->adrSigfox = request('sigfox');  
    $boitier->statut = request('statut');  
    $boitier->alarmeBatterie = NULL;  
  
    $boitier->save();  
  
    return redirect()->route('listeboitieralert_path', ['alert' => 1]);  
}
```

En temps normal, pour ajouter une donnée dans une table, on utilise la commande
INSERT INTO nom_table VALUES ...

Avec Laravel on simplifie le travail

On ajoute une données dans une table en faisant la méthode → **save()** ;

Pour Modifier c'est → **update()** ; et pour supprimer c'est → **delete()** ;

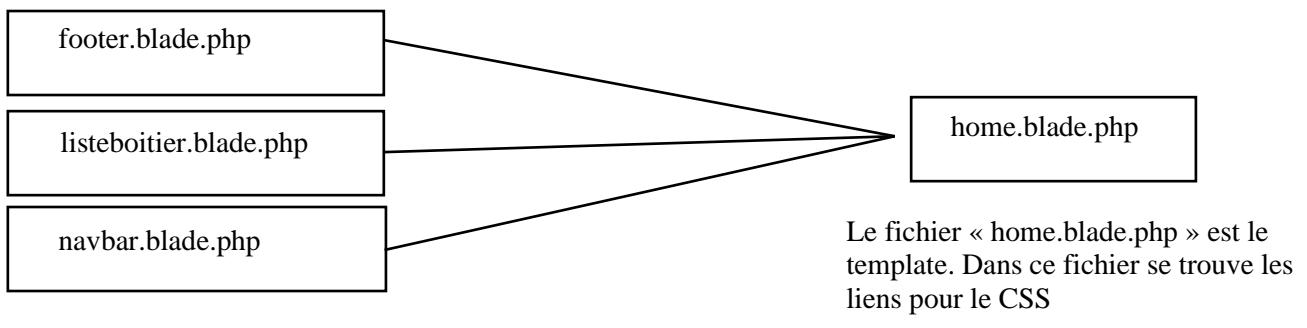
Pour récupérer une données dans une table, on utilise la méthode **::all()** au lieu de taper la requête SQL **SELECT * FROM nom_table**

Utilisation de la Vue « listeboitier.blade.php »

Toutes les vues dans la Laravel doivent obligatoirement finir par .blade.php dans son nom de fichier. Laravel utilise le langage « blade » pour aussi faciliter l'utilise du php dans les Vues.

Le principe de Laravel est d'avoir une Vues principale, qu'on appelle plus particulièrement « Template » puis on incrémente d'autre pages dans le template. Ce template sert à éviter de répéter le code qu'on pourrait plusieurs fois. Par exemple, si on veut une barre de navigation sur chaque pages, on le code dans le template puis on incrémente 1 nouveau code pour chaque pages. Cela permet d'éviter de retaper le code de la barre de navigation à chaque fois.

Schéma



Comment fonctionne les vues ?

Dans le fichier « home.blade.php » on y trouve « @yield('') » et dans les fichiers à incrémenter, on y trouve « @section('') ». Laravel fait le lien entre le @yield et le @section. On y trouve aussi des @include pour intégrer directement le code du fichier à implémenter.

Pour expliquer le fonctionnement, je vais utiliser la page « listeboitier.php » comme exemple.

Voici la vue « listeboitier.blade.php »

The screenshot shows a web application interface. At the top, there is a dark header bar with the text "KOLANTR Accueil | Gestion Compte Client Gestion Campagne Mesure Gestion Boitier Liste" and a user profile "HOUPIN Armand" with a gear icon. Below the header is a title "Gérer Boitier" with a magnifying glass icon and a search bar labeled "Rechercher". A red box highlights the main content area, which displays a table of device information. The columns are "Adresse @Sigfox", "Niveau Batterie", "Statut", and "Edit Delete". Two rows are shown: one with address "123456", battery level "Batterie Correct", status "utilisé", and edit/delete icons; another with address "2ED518", battery level "Batterie Correct", status "utilisé", and edit/delete icons. At the bottom, a blue box contains the copyright notice "© 2020 – 2021 Lycée Alfred Kastler Talence".

Sur cette vue, on y voir une barre de navigation (cadre en violet), la liste boitier (cadre en rouge) et le footer (cadre en bleu)

Ce fichier est composé du fichier principale « home.blade.php » et 3 autres fichiers « listeboitier.blade.php », « navbar.blade.php » et « footer.blade.php »

```
@yield('style')
</head>

<body class="@if(Auth::user()->preference == 'theme_dark') bg-dark @else bg-light @endif">
@include('layouts.partials.navbar')

@if(Auth::user()->hasRole('admin') or Auth::user()->hasRole('technicien'))

@if(!empty($alarme_popup))
| @include('alertbatterie')
@endif

@endif

@yield('content')

<script src="{{ asset('/js/bootstrap.bundle.min.js')}}"></script>
<script src="{{ asset('/js/bootstrap.bundle.js')}}"></script>

@yield('scriptAlert')
@yield('script')

</body>

</html>
```

La page
« home.blade.php »

On y trouve un
{@yield('style')}
{@yield('content')}

Et un
@include('layouts.
partials.navbar')

L'intégration du code de la
barre de navigation se situe
donc dans le @include (
cadre violet)

Le code pour la liste des
boitiers se trouve dans le
{@yield('content')} (cadre
rouge)

L'extrait du code de la barre de navigation qui se trouve dans le fichier « navbar.blade.php »

```
<header>
<nav class="navbar navbar-expand-lg navbar-dark fixed-top bg-violet" aria-label="Eighth navbar example">
  <div class="container">
    <a class="navbar-brand" href="{{ route('login') }}>KOLANTR</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarsExample07">
      <span class="navbar-toggler-icon"></span>
    </button>

    @guest
      <div class="collapse navbar-collapse" id="navbarsExample07">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item active">
            <a class="nav-link" href="#">Home
          </li>
        </ul>
        <ul class="navbar-nav me_auto mb-2 mb-lg-0">
          <li>
            <button class="btn btn-outline-success" type="submit" name="ville"> <a class="nav-linkArmand" href="#">Armand</a>
          </button>
        </ul>
      </div>
    @endguest
  </div>
</nav>
```

L'extrait du code du @yield('content') (la liste des boîtiers)

```
@extends('layouts.home', ['title' => 'Gérer Boitier Mesure'])

@section('style')
  <link href="{{ asset('/bootstrap/form-validation.css') }}" rel="stylesheet">
@stop

@section('content')

@if(!empty($alert))
  <div class="alert success text-center" id="alert">
    <span class="closebtn">&times;</span>
    <strong>Succés !</strong>
    <hr>
    {!! $messagealert !!}
  </div>
@endif



<main>

    <div class="py-5 text-center">
      <h2 class="mx-auto mb-1 py-5 featurette-heading"><span class="text-muted"> <i class="fas fa-cubes" width="75" height="75"></i> </span> Gérer Boitier Mesure</h2>
    </div>

    <div class="row g-3 shadow p-3">
      <div class="col-md-7 col-lg-12 mx-auto">
        <div class="row g-3">
          <div class="col-md-8">
            <a href="{{ route('creationBoitier_path') }}><svg xmlns="http://www.w3.org/2000/svg" width="30" height="35">
              <path d="M6 8a3 3 0 1 0 0-6 3 3 0 0 0 6zm2-3a2 2 0 1 1-4 0 2 2 0 0 1 4 0zm4 8c0 1-1 1-1 1H1s-1 0-1-1 1-4 6
              <path fill-rule="evenodd" d="M13.5 5a.5.5 0 0 1 .5.5V7h1.5a.5.5 0 0 1 0 1H14v1.5a.5.5 0 0 1-1 0V8h-1.5a.5.5 0 0 1 0-1z" />
            </a>
          </div>
        </div>
      </div>
    </div>
  </main>


```

On y retrouve le « content » (cadre rouge).Le footer (cadre bleu sur la première image) se trouve à la fin du fichier « listeboiter.blade.php »

```
@if(Auth::user()->hasRole('admin') or Auth::user()->hasRole('technicien') or $title != "Home" )

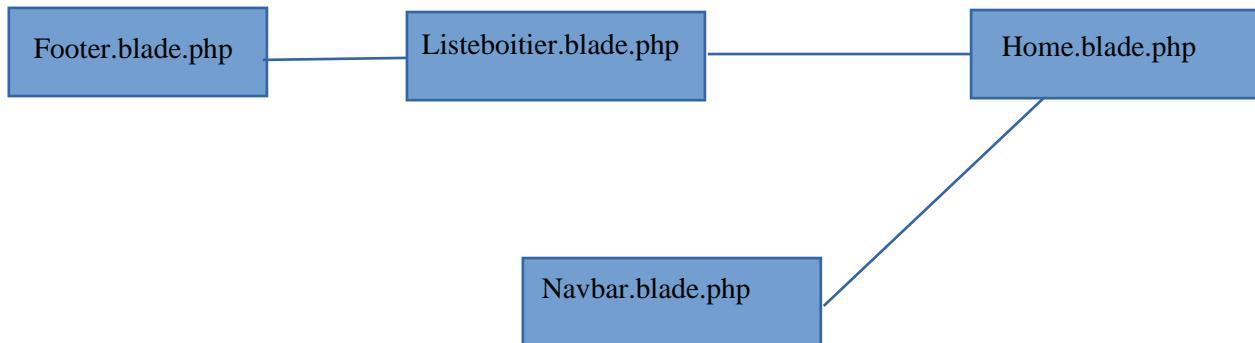
  @include('layouts.partials.footer')

@endif
```

On appelle le fichier « footer.blade.php » à s'intégrer dans le fichier « listeboitier.blade.php »

```
<footer class="my-5 pt-5 text-muted text-center text-small">
    <p class="mb-1">&copy; 2020 - 2021 Lycée Alfred Kastler Talence</p>
</footer>
```

Si on devait le définir avec un schéma



Les migrations dans la Laravel

Dans Laravel, il existe un système qu'on appelle « migration ». Cela permet de créer les tables avec les attributs dans la base de données sans utiliser MySQL. Laravel peut insérer grâce à une commande les tables avec les attributs.

On utilise les migrations quand on veut déployer l'application et éviter de taper en ligne de commande de MySQL pour créer les tables.

Pour commencer, il faut ouvrir un terminal dans la racine du dossier de l'application puis il faut créer un fichier avec le nom de la table avec la commande suivante :

```
php artisan make:migration create_boitiers_table
```

« make:migration » est une commande d'artisan qui permet de créer un fichier de migration

« create » permet à Laravel de créer la table. On peut le remplacer par « after » qui permet à Laravel de modifier une table existante, comme rajouter un attribut.

« boitiers » est le nom de la table à créer.

Après avoir réalisé la commande, un nouveau fichier apparaît dans le dossier database/migrations. Dans ce fichier il faut taper les attributs que l'on veut y mettre dans la table comme l'exemple ci-dessous

Extrait du fichier « 2021_02_05_1500113_create_boitiers_table.php »

```
class CreateBoitiersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('boitiers', function (Blueprint $table) {
            $table->id();
            $table->string('adrSigfox')->nullable();
            $table->string('alarmeBatterie')->nullable();
            $table->string('statut')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('boitiers');
    }
}
```

On y retrouve les attributs du modèle de données « boitier »

Après avoir fini de mettre les attributs, il faut taper la commande suivante dans un terminal

```
php artisan migrate
```

Laravel va automatiquement ajouter toute les tables et les attributs associer à la table dans la base de données. Ce système est utilisé sur des applications qu'il doit être déployer sur des serveurs.

Utilisation du javascript dans l'affichage des graphiques sur la page « consultation campagne.blade.php »

Cette page doit permettre au client de visualiser la campagne de mesure demander et d'y voir toutes les statistiques liées à cette campagne de mesure. Le client pour voir cette statistique sous formes de graphiques



Pour réaliser ce graphique (image pris depuis le site), j'ai utilisé le bibliothèques HighCharts. Cette bibliothèque est spécialisée dans les graphiques.

J'ai utilisé leur code pour faire mes graphiques

Ceci est un extrait de code qui permet de dire au javascript où il doit afficher le graphique

Extrait du fichier « graphiques_camion.blade.php »

```
<div class="row featurette shadow p-3 mb-5">
  <div class="col-md-13">
    <h2 class="featurette-heading @if(Auth::user()->preference == 'theme_dark') tx-white @else tx-black @endif> Graphique voit<br>
    <hr>
    <div class="row">
      <div class="col-lg-4" style="width: 33%; ">
        <button type="button" class="w-100 btn btn-sm btn-outline-secondary" onclick="AfficherGraphiqueLigne_voiture();"></button>
      </div>
      <div class="col-lg-4" style="width: 33%; ">
        <button type="button" class="w-100 btn btn-sm btn-outline-secondary" onclick="AfficherGraphiqueCercle_voiture();"></button>
      </div>
      <div class="col-lg-4" style="width: 33%; ">
        <button type="button" class="w-100 btn btn-sm btn-outline-secondary" onclick="AfficherGraphiqueBarre_voiture();"></button>
      </div>
    </div>
    <hr>
    <div id="graphiques_lignes_voiture" style="display: block;"></div>
    <div id="graphiques_cercle_voiture" style="display: none;"></div>
    <div id="graphiques_barres_voiture" style="display: none;"></div>
  </div>
</div>
```

Extrait du fichier « graphiques_camion.blade.php »

```
Highcharts.chart('graphiques_lignes_voiture', {
    chart: {
        type: 'line'
    },
    title: {
        text: 'Limitation : ' + Limitation + ' km/h',
    },
    xAxis: {
        categories: statHeureStat,
    },
    yAxis: {
        title: {
            text: 'Nombre de voiture'
        },
        plotLines: [
            {
                value: Limitation,
                color: 'red',
                width: 1,
                label: {
                    text: Limitation
                }
            }
        ]
    },
    plotOptions: {
        line: {
            dataLabels: {
                enabled: true
            },
            enableMouseTracking: false
        }
    },
    series: [
        {
            name: 'VitesseInferieurOuEgale',
            data: statVitesseInferieurOuEgale,
        }, {
            name: 'VitesseLimitMoins20',
            data: statVitesseLimitMoins20,
        }, {
            name: 'VitesseLimitplus20',
            data: statVitesseLimitPlus20,
        }, {
            name: 'VitesseSuperieur20',
            data: statVitesseSuperieur20
        }
    ]
});
```

A gauche, nous avons le script qui permet de créer le graphique.

Ce qui se trouve dans le cadre rouge, permet au script de savoir à quelle div il doit afficher son graphique

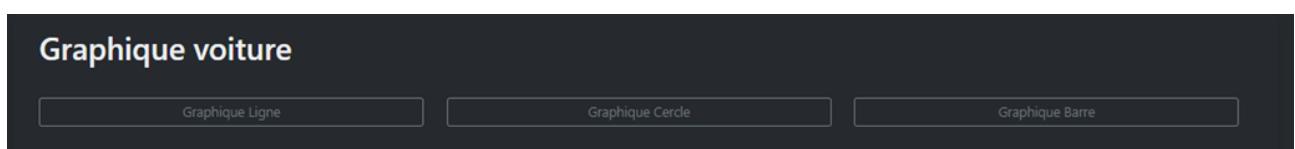
On peut mettre un titre au graphique,
La légende sur chaque axe et les données.

Le script de Highcharts récupère toutes les données passer en paramètres sous forme de tableau. Le script accède tout seul aux données contenues dans les tableaux.

Extrait du fichier « graphiques_camion.blade.php »

```
function AfficherGraphiqueLigne_voiture() {
    var div = document.getElementById("graphiques_lignes_voiture");
    var div2 = document.getElementById("graphiques_cercle_voiture");
    var div3 = document.getElementById("graphiques_barres_voiture");
    div.style.display = "block";
    div2.style.display = "none";
    div3.style.display = "none";
    document.getElementById("graphiques_lignes_voiture").disabled = false;
    document.getElementById("graphiques_cercle_voiture").disabled = true;
    document.getElementById("graphiques_barres_voiture").disabled = true;
}

function AfficherGraphiqueCercle_voiture() {
    var div = document.getElementById("graphiques_lignes_voiture");
    var div2 = document.getElementById("graphiques_cercle_voiture");
    var div3 = document.getElementById("graphiques_barres_voiture");
    div.style.display = "none";
    div2.style.display = "block";
    div3.style.display = "none";
    document.getElementById("graphiques_lignes_voiture").disabled = true;
    document.getElementById("graphiques_cercle_voiture").disabled = false;
    document.getElementById("graphiques_barres_voiture").disabled = true;
}
```



Le client peut choisir entre différent graphiques, soit en ligne, soit en camembert, soit en barre

Le graphique en ligne est le graphique par défaut puis le client pour changer le graphique. Pour cela (1ere image), on change le style de la div avec du javascript. Quand il clique sur un bouton, il cache le graphique déjà existant puis fait apparaître le nouveau graphique. Pour avoir aucun problème avec la div, on la désactive et on active la div correspondant au graphique.

L'envoie automatique d'e-mail

Quand on ajoute un compte, un e-mail s'envoie sur l'adresse mail du compte.
Pour effectuer cette envoie de mail automatique, j'utilise une api de google.

Dans le fichier de config «.env» qui se trouve à la racine du dossier

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.googlemail.com
MAIL_PORT=465
MAIL_USERNAME=kolantr2021snir@gmail.com
MAIL_PASSWORD=bipnxvvaydpbjxfa
MAIL_ENCRYPTION=ssl
```

Dans le fichier, on indique les informations de l'api google liée à l'adresse mail du projet

Pour créer un email, on doit créer une classe associer

Le fichier « MessageGoogle.php » définit la classe

```
class MessageGoogle extends Mailable
{
    use Queueable, SerializesModels;

    public $data; // Données pour la vue

    public function __construct($data)
    {
        $this->data = $data;
    }

    public function build()
    {
        return $this->from("kolantr2021snir@gmail.com") // L'expéditeur
            ->subject("Message Automatique Kolantr ( ne pas répondre )")
            ->view('emails.message-google'); // La vue
    }
}
```

On retrouve l'adresse mail du projet. L'objet de l'e-mail dans le « subject ». Le view permet de créer l'affichage de l'email. Ce que le client va voir dans l'e-mail.

Dans le fichier « message-google.blade.php » on y trouve le contenu de l'e-mail

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body style="background: #e5e5e5; padding: 30px;" >

@if(!empty($data['identifiant']))
<div style="max-width: 500px; margin: 0 auto; padding: 20px; background: #fff;">
    <h3>Bonjour {{ $data['name'] }} {{ $data['prenom'] }} </h3>

    <h3>Suite à votre demande de campagne de mesure sur Kolantr, nous avons créé votre compte </h3>
    <br>
    <h6> Vous pouvez vous connecter sur le site <a href="http://kolantr.sn-kastler.fr:8000"> kolantr </a> avec ces identifiants :
    <br>
    <h6> identifiant : {{ $data['identifiant'] }} </h6>
    <h6> Mot de passe : {{ $data['password'] }}</h6>
    <br><br><br>
    <h6 style="color : red"> Rappel: Veuillez sauvegarder votre identifiant ou mot de passe dans un endroit sécurisé et ne pas donner
    <br><br>
    <h6> Support : kolantr2021snir@gmail.com </h6>
</div>
```

On remarque qu'un e-mail est de l'html.

Puis pour envoyer un e-mail avec les données en paramètre on utilise le code suivant

```
Mail::to($client)->bcc("kolantr2021snir@gmail.com")
    ->queue(new MessageGoogle($request->all()));
```

Voilà à quoi ressemble l'e-mail reçu (avec l'extrait de code ci-dessus du fichier

« message-google.blade.fr »)

Bonjour HOUPIN Armand

Suite à votre demande de campagne de mesure sur Kolantr, nous avons créer votre compte

Vous pouvez vous connecter sur le site [kolantr](#) avec ces identifiants :

identifiant : /t21qeswqf2zk8415

Mot de passe : .gc9ihpbyr21rl06vye2b

Rappel: Veuillez sauvegarder votre identifiant ou mot de passe dans un endroit sécurisé et ne pas donner vos identifiants à autrui !

Support : kolantr2021snir@gmail.com

Le principe est le même pour les alertes batterie. Quand on batterie est vide, les techniciens et les administrateurs reçoivent une alerte par e-mail.

D. Tests

On va réaliser deux tests d'intégration.

Le premier est sur l'ajout du compte admin et le deuxième sur l'association d'une campagne de mesure avec un client.

AJOUT D'UN COMTE ADMIN

Regardons la table user dans la base de données

	id	role_id	name	prenom	identifiant	email	avatar	
user	1	1	admin	admin	123456789	admin@admin.com	users/default.png	/
user	3	2	HOUPIN Armand	c21uzpc08u7f8439	armando.houpin@hotmail.fr	users/default.png	/	

On a que 2 comptes créer pour l'instant.

Sur le site web, dans la liste des comptes admin et techniciens, il y a que le compte admin qui apparaît car c'est le seul compte administrateurs

The screenshot shows a user management interface with a header containing a person icon and a '+' sign. Below the header is a table with columns: Prénom, Nom, Email, Identifiant, Edit, and Delete. There is one visible row of data:

Prénom	Nom	Email	Identifiant	Edit	Delete
admin	admin	admin@admin.com	123456789		

Maintenant, ajoutons à compte client

Prénom	Nom
Hugo	Carricart
Identifiant	
a/tl5m6o75x2se30539	
Mot de passe	
15sxl4jz3szl5uvndfserl	
Mot de passe	
15sxl4jz3szl5uvndfserl	
Adresse Mail	
armandohoupin33380@gmail.com	
Role	
<input checked="" type="radio"/> Admin	
<input type="radio"/> Technicien	

CRÉE LE COMPTE

On crypte le mot de passe pour ne pas le voir dans la base de données

Le compte apparaît dans la base de données et dans la liste des compte administrateurs et technicien

id	role_id	name	prenom	identifiant	email	avatar	email_verified_at
1	1	admin	admin	123456789	admin@admin.com	users/default.png	NULL
3	2	HOUPIN Armand		c21uzpc08u7f8439	armando.houpin@hotmail.fr	users/default.png	NULL
4	1	Carricart Hugo		a/t15m6o75x2se30539	armandohoupin33380@gmail.com	users/default.png	NULL

Personne +

Prénom	Nom	Email	Identifiant	Edit	Delete
admin	admin	admin@admin.com	123456789		
Hugo	Carricart	armandohoupin33380@gmail.com	a/t15m6o75x2se30539		

On retrouve les informations qu'on a insérer au formulaire.

De plus, le client reçoit un e-mail avec son identifiant et son mot de passe

Bonjour **Carricart Hugo**

Suite à votre demande de campagne de mesure sur Kolantr, nous avons créer votre compte

Vous pouvez vous connecter sur le site [kolantr](#) avec ces identifiants :

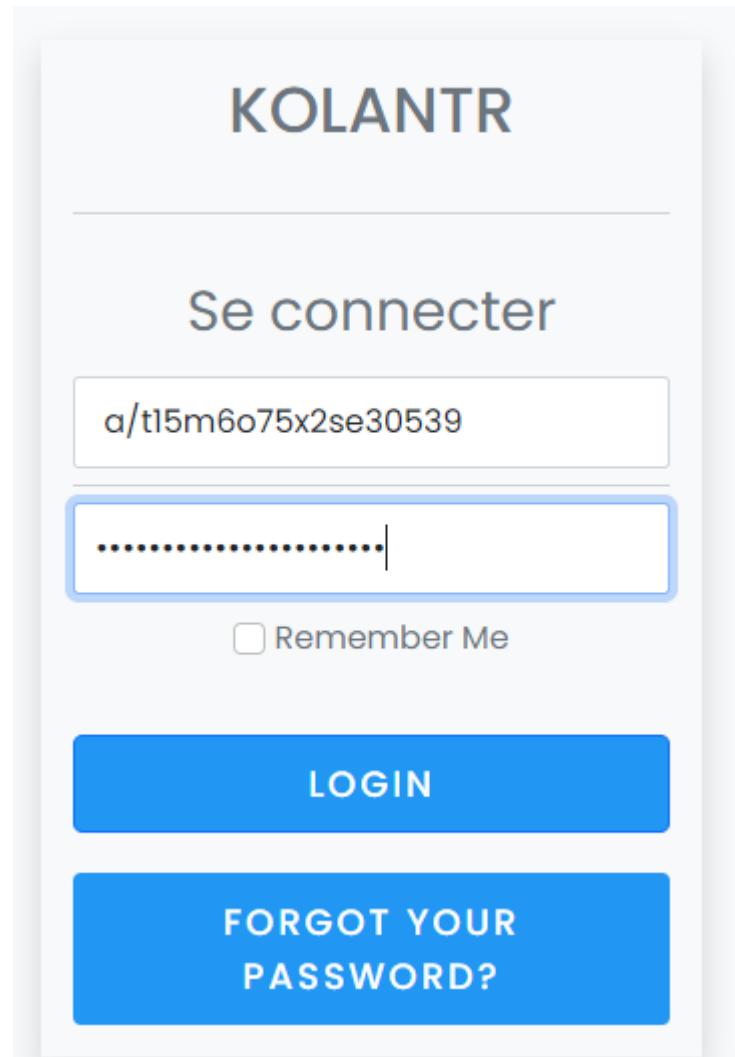
identifiant : a/t15m6o75x2se30539

Mot de passe : 15sxI4jz3sz15uvndfserl

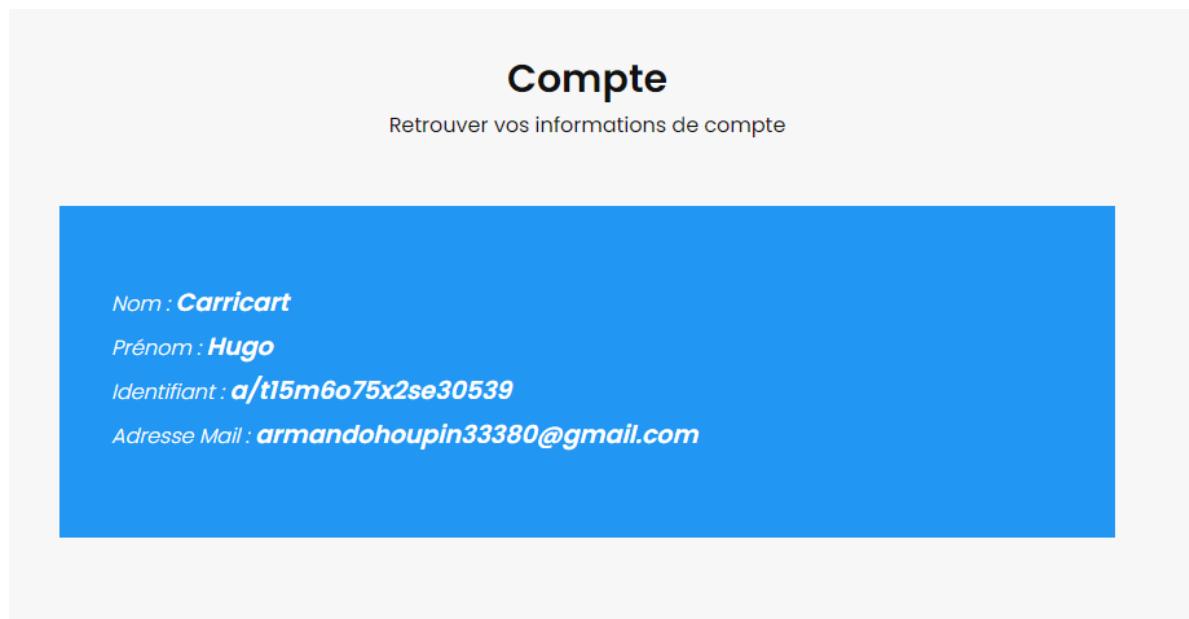
Rappel: Veuillez sauvegarder votre identifiant ou mot de passe dans un endroit sécurisé et ne pas donner vos identifiants à autrui !

Support : kolantr2021snir@gmail.com

Le client peut se connecter avec son identifiant et son mot de passe



Puis sur la page accueil, il peut retrouver les informations de son compte



et c'est un compte administrateur comme demandé dans le formulaire car il a accès au client

ASSOCIER UNE CAMPAGNE DE MESURE A UNE CLIENT

Après être connecter sur le compte client, le compte « ARMAND HOUPIN » en question.



J'ai caché mes informations personnelles

On peut accéder à l'onglet « Mes campagnes de mesure »

Adresse	Ville	Date début	Statut	limitation
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

Pour l'instant aucune campagne de mesure n'est associée au compte

On créer une campagne sans associer de compte client.

Adresse de la campagne

10 Rue Louis Blériot

Direction (information)

...

N° de la route

...

Ville

Marcheprime

Code Postal

33380

Date Début

06/04/2021



Date Fin (laisser vide si en cours)

jj/mm/aaaa



Statut

En cours

Fini

Limitation Vitesse

30

50

70

80

90

110

130

Associer client

Aucun client

Associer Boitier

123456

Puis on associe le client en choisissant le client dans la liste déroulante

The screenshot shows a user interface for associating a client. At the top, there is a label "Associer client". Below it is a dropdown menu with a blue border containing the text "HOUPIN Armand". The rest of the interface is mostly white space.

On clique sur le bouton pour mettre à jour la campagne de mesure.
Maintenant, retournons sur les campagnes de mesure du client

The screenshot shows a table with five columns: "Adresse", "Ville", "Date début", "Statut", and "limitation". A red box highlights the first column, "Adresse". The table contains one row of data: "10 Rue Louis Blériot", "Marcheprime", "2021-04-06", "en cours", and "30". To the right of the last column is a magnifying glass icon.

Adresse	Ville	Date début	Statut	limitation
10 Rue Louis Blériot	Marcheprime	2021-04-06	en cours	30

La campagne de mesure est apparue avec les informations de la campagne de mesure
Si on clique sur la loupe à gauche, on retrouve toutes les informations

The screenshot shows a detailed view of a campaign. At the top, it says "Campagne de mesure". Below that is a list of information items, each preceded by a small blue magnifying glass icon. A red box highlights this list. At the bottom, there are two blue buttons: "TÉLÉCHARGER LE CSV" and "HIGHSCORE".

Adresse de la campagne :	10 Rue Louis Blériot
Ville de la campagne :	Marcheprime
Code postal de la campagne :	33380
Statut de la campagne :	en cours
Date début de la campagne :	2021-04-06
Limitation vitesse :	30

TÉLÉCHARGER LE CSV **HIGHSCORE**

E. Partie physique

Ma partie physique se porte sur un comparatif entre le réseau Sigfox et le réseau Lorawan.

SIGFOX

Sigfox est un opérateur de télécommunications français créé en 2009 par Christophe Fourtet et Ludovic Le Moan. C'est un opérateur télécom de l'Internet des objets.



LORAWAN

LoRaWAN est un 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.



Ces deux réseaux sont spécialisé dans l'IOT (Internet des Objets)

L'Internet des objets ou IdO est l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques. L'appellation désigne un nombre croissant d'objets connectés à Internet permettant ainsi une communication entre nos biens dits physiques et leurs existences numériques.

POURQUOI UTILISONS CES RÉSEAUX :

Des réseaux longue portée

L'intérêt des technologies longue portée est non seulement de limiter le nombre d'antenne nécessaire mais aussi d'autoriser une émission depuis les zones enterrées. LoRaWAN comme Sigfox mettent en avant leur capacité à remonter des données depuis les sous-sols où se trouvent souvent les équipements techniques des bâtiments à superviser.

Des réseaux à faible coût

Sigfox et LoRa ont apporté des progrès significatifs dans les techniques de modulation (Ultra Narrow Band pour Sigfox et Chirp Spread Spectrum pour LoRa) pour augmenter la portée (le récepteur peut démoduler correctement un message arrivant avec -130 dBm d'amplitude). Ils annoncent 15 km de portée en champ libre, et 1 km en ville.

Des réseaux faible consommation

Les réseaux LoRaWAN et Sigfox ne se réveillent que ponctuellement pour émettre et à ce moment-là peuvent éventuellement recevoir. Cela implique un temps de latence important voire très important pour recevoir une instruction. C'est pour cela que l'on trouve essentiellement des capteurs sur ce type de réseau.

En dessous se trouve un tableau qui compare les deux réseaux

Feature	LoRaWAN	SIGFOX
Modulation	SS Chirp	UNB/GFSK/BPSK
Rx bandwidth	500 - 125 kHz	100 Hz
Data Rate	290bps - 50kbps	100 bit/sec 12/8 bytes Max
Max. # Msgs/day	Unlimited	UL : 140 msgs/day
Max Output Power	20 dBm	20 dBm
Link Budget	154 dB	151 dB
Battery lifetime - 2000mAh	105 months	90 months
Power Efficiency	Very High	Very High
Interference immunity	Very High	Low
Coexistence	Yes	No
Security	Yes	No
Mobility/localization	Yes	Limited mobility, No loc

Sigfox utilise la bande 868Mhz en Europe (902Mhz aux US). Un objet SIGFOX peut envoyer entre 0 et 140 messages à 300bits/s par jour et le payload de chaque message ne peut pas dépasser 12 octets. Le protocole Sigfox est bidirectionnel sous condition : un objet Sigfox peut recevoir 4 messages par jour à des instants définis.

LoRaWAN utilise la bande de fréquences 868Mhz. LoRaWAN est la gestion de la couche MAC, et permet de façon dynamique d'optimiser le lien entre l'objet LoRa et la station de base : canal de fréquence, puissance d'émission, débit, LoRaWAN peut être opéré par un opérateur TélécomBT, Orange, ... ou utilisé un réseau privé. Le protocole LoRa est bidirectionnel sous conditions.

Explication des modulations

Modulation Chirp Spread Spectrum (LORAWAN)

Le succès de LoRaWAN repose essentiellement sur la modulation LoRa qui permet une démodulation correcte dans des conditions de rapport signal/bruit très mauvaises. Cette qualité permet une longue portée, y compris à travers les murs et plancher.

Cette modulation nommée Chirp Spread Spectrum consiste à moduler les bits par des chirp. Un chirp est un signal dont la fréquence augmente de manière continue. La modulation par saut de fréquence, très utilisée, consiste à envoyer une fréquence pour le 1 et une autre pour le 0.

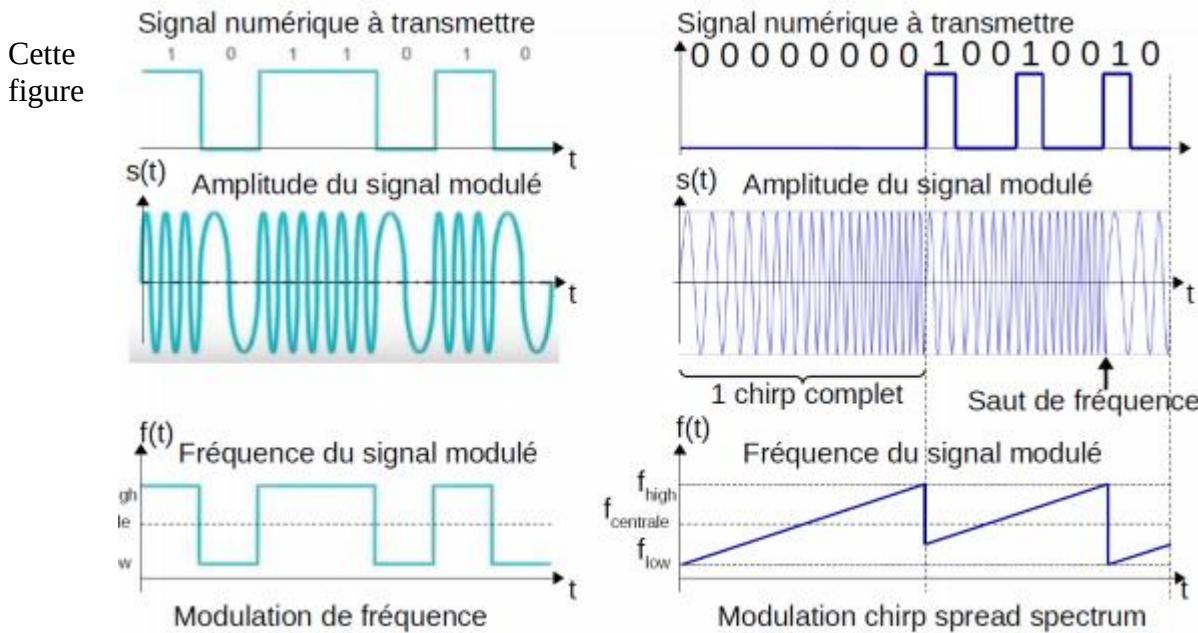


Figure 6 : Principe de la modulation chirp spread spectrum

compare l'utilisation d'une amplitude normal (à gauche) à l'amplitude du signal avec l'utilisation de la modulation chirp spread spectrum

SIGFOX

En Europe, la bande de fréquence ISM utilisée est celle de 868 MHz et les technologies de modulation sont DBPSK et GFSK. Il utilise aussi la modulation UNB

GFSK

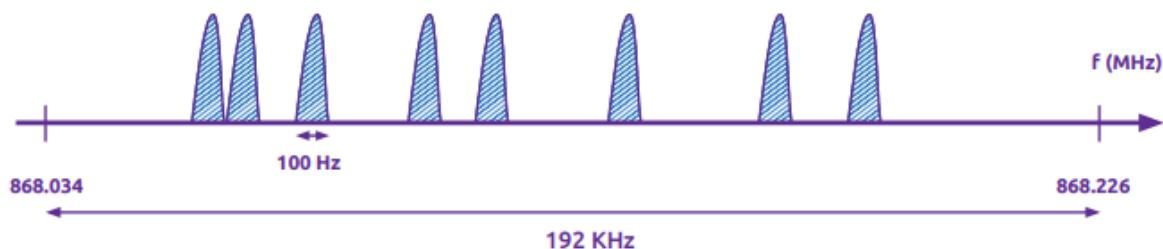
La modulation par déplacement de fréquence (MDF), plus connue sous sa dénomination anglophone frequency-shift keying (FSK) est un mode de modulation de fréquence numérique dans lequel la fréquence du signal modulé varie entre des fréquences prédéterminées.

DBPSK

La modulation BPSK (Binary Phase Shift Keying) est une modulation de phase à 2 états de la fréquence intermédiaire. Comme il n'y a, à priori, aucune relation de phase et de fréquence entre la FI et le signal modulant, on synchronise celui-ci sur la FI par une simple bascule D.

UNB

Sigfox utilise 192 KHz de la bande non licenciée ISM pour échanger des messages par liaison radio. La technique d'émission est ce que l'on appelle la bande ultraétroite (UNB – Ultra Narrow Band). Chaque message occupe 100 Hz (zones ETSI) ou 600 Hz (zones FCC) et est transféré à un débit de 100 ou 600 bits par seconde selon la région.



F. Bilan personnel technique

Tache à réaliser :

- Gérer les campagnes de mesure
- Gérer les comptes
- Gérer les boîtiers
- Alerter niveau batterie

Depuis un compte client

- Voir ces informations de compte
- Voir ces campagnes de mesure

Tache effectuée à ce jour :

- ✓ Gérer les campagnes de mesure
- ✓ Gérer les comptes
- ✓ Gérer les boîtiers
- ✓ Alerter niveau batterie

Depuis un compte client

- ✓ Voir ces informations de compte
- ✓ Voir ces campagnes de mesure

L'intégration de la partie de l'étudiant n°6, Théo Sernot, la partie avec Sigfox à déjà était faite et opérationnel. Les statistiques envoyés par Sigfox sont liées à la campagne de mesure.

Le site web à été déployé sur une machine OVH. La machine OVH à été préparée et sécurisée par l'étudiant n°3, Hugo Carricart.

Vous pouvez accéder au site web en allant sur ce lien <http://kolantr.sn-kastler.fr:8000/>

Compte Admin :

identifiant : adminKolantr2021

mot de passe : password

Compte client :

identifiant : c21fpoknptot8416

mot de passe : 21y9yvjq6zr21gewnbs1qv