Projet CPO ENIB S6:

DolMen

Allègre Timothée - t7allegr@enib.fr
Nougier Axel - a7nougie@enib.fr
De Saint Just Nathan - n6desain@enib.fr

1. Résumé du projet

Français:

Conçu pour le domaine spatial, le projet DolMen est bien plus qu'une simple interface, c'est un outil complet et modulable permettant de suivre une fusée expérimentale depuis son décollage jusqu'à sa récupération, de générer des rapports détaillés de chaque vol, ou même de simuler les données d'un tir à partir d'une trame de données externe.

Aussi ergonomique que simple d'utilisation, il s'adapte à n'importe quelle configuration de fusée, quelque soit le nombre et la nature des capteurs, et quelque soit le mode de communication avec le sol.

L'Équipe Dolmen.

English:

Designed for the space domain, the DolMen project is much more than a simple interface, it is a complete and modular tool allowing to follow an experimental rocket from its takeoff to its recovery, to generate detailed reports of each flight, or even to simulate the data of a launch from an external data frame.

As ergonomic as it is simple to use, it adapts to any rocket configuration, whatever the number and nature of the sensors, and whatever the mode of communication with the ground.

Dolmen team.

2. Table des matières

3. Problématique

3.1. Enjeux

Dans le cadre du projet d'informatique de 3ème année à l'ENIB, Nous devons réaliser un logiciel/application en programmation objet répondant au moins un objectif du développement durable ODD. Nous avons décidé de nous placer sous le cadre de l'objectif 9 : " Mettre en place une infrastructure résiliente, promouvoir une industrialisation durable qui profite à tous et encourage l'innovation ". En effet, en tant que élèves ingénieurs ainsi que dans notre futur métier, c'est l'objectif auquel nous serons le plus généralement confronté, il nous apparaît donc important de s'y atteler au plus tôt.

3.2. Démarche

En plus de ces enjeux, nous souhaitons mettre un place notre projet dans une situation et un contexte concrets en répondant aux besoins d'une entreprise ou d'une association.
C'est pour cela que nous avons décidé de répondre à la problématique du BDI qui est la réalisation d'une interface de traitement des données reçues du vol d'une fusée. Mais nous souhaitons proposer une solutions qui ne soit pas utile uniquement pour le BDI. En effet d'autres organismes ont les même besoins que le BDI. C'est pour ça que nous souhaitons que le logiciel soit en accès libre et le plus modulaires que possible afin de pouvoir s'adapter aux différents utilisateurs.

4. Contexte

4.1. Le BDI et pôle KSP

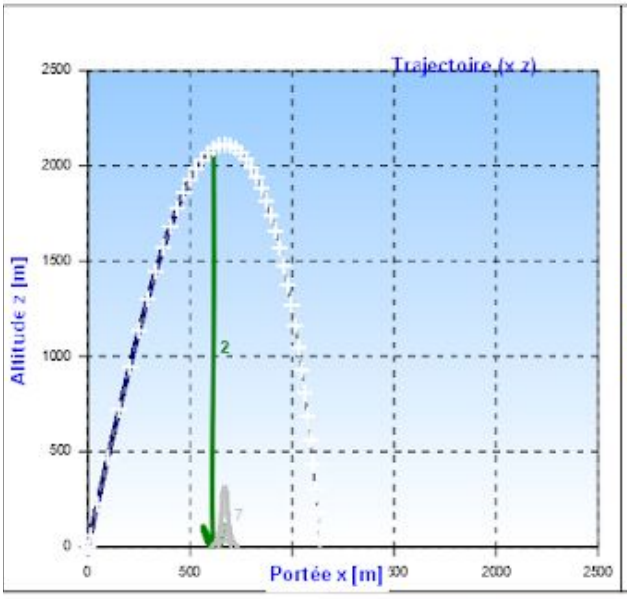
Le BDI est une association loi 1901 qui promeut et développe des projets d'ingénieries innovants à destination des étudiants de l'ENIB. L'association de décompose en plusieurs pôles portant chacun un ou plusieurs projets sur le même thème.

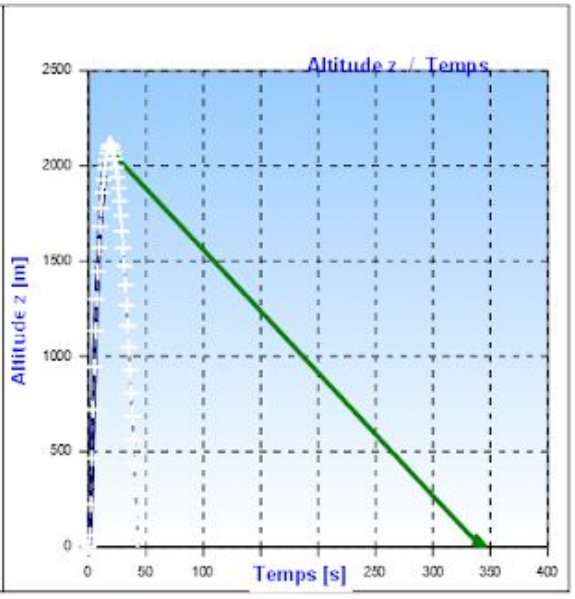
Le pôle Korrigan Space Program (KSP) qui a pour objectif final de développer une fusée capable d'aller dans l'espace. Bien sûr cette objectif ne sera pas réalisé du premier coup, c'est pour ça que l'association s'est inscrite dans le programme de la C'space organisé par planète-science et le CNES. Ce programme permet une session de tir de petites fusées de différentes tailles, en juilllet, allant de la mini-fusée (environ 60 cm) à la fusée expérimentale de 2 m. Ce pôle s'occupe aussi d'autres activités en lien avec l'espace, comme un échange avec l'association "les gens de la lune" autour de leur observatoire.

4.2. Le projet HERMIN

Le projet consiste au développement des différentes fusées expérimentales afin d'acquérir les différentes connaissance techniques pour la réalisation de la fusée finale. La première de ces fusées porte le nom de HERMIN 1. Le développement de celle-ci a commencé en mai 2018 avec la création du pôle KSP. L'objectif est de démontrer la capacité de conception et de fabrication de l'association.

Hermin 1 est une fusée expérimentale de 1m50 de haut pour 80 mm de diamètre et d'une masse de 6 kilos. Elle est équipée d'un moteur Pro54-5G / BARASINGA fourni par le CNES. Avec ce moteur, elle doit atteindre une altitude de 2100 m avec une vitesse de pointe approchant mach 0,8. Le retour s'effectue à l'aide d'un parachute de 2m² de surface à 5,6 m/s. Elle pourra être retrouvée au sol grâce à un module GPS embarqué à bord.





25/03/2020	Temps	Altitude z	Portée x	Vitesse	Accélération	Efforts
Sortie de Rampe				32,1 m/s		
Vit max & Acc max				272 m/s	135 m/s²	
Culmination, Apogée	19,0 s	2112 m	669 m	27 m/s		
Ouverture parachute fusée	17,0 s	2092 m	614 m	34 m/s		1442,0 N
Impact balistique	43,4 s	~0 m	1138 m	132,5 m/s		45352 J

stabtraj_v3-4-HERMIN-I-final.xls

<u>L'expérience:</u>

Hermin 1, pour être autorisée à décoller, doit embarquer une expérience scientifique avec elle. Pour son premier tir, elle part avec 7 capteurs (température x2, pression x2, accélération, rotation, magnétisme). Ces données seront ensuite comparées au modèle existant pour vérifier leur validités. Tout les données seront sauvegardées dans une carte micro-sd.

Avec l'avancement du projet, il est devenu envisageable de suivre la fusée depuis le sol, par le biais d'une interface dédiée, permettant d'afficher en temps réel les données acquises par les différents capteurs, et de générer un rapport détaillé du vol, avec par exemples des graphiques décrivant la courbe de position du vol, ou bien son placement sur une carte.

4.3. Le projet DolMen

Le projet Dolmen :

Le nom DOLMEN vient des fameux dolmens celtiques que l'on retrouve un peu partout en Europe mais surtout en Bretagne. Il rentre dans le "Breton" du Projet et va de pair avec le reste de la base sol (pas de tir, centre de communication, ...) qui n'est pas encore développée car nous n'avons pas encore l'utilité du nom de CUMULUS Center. (Le dolmen étant la structure interne du cumulus).

Le logo :

Le logo est là pour différencier le projet du KSP tout en lui faisant largement référence et de définir le projet DOLMEN comme un véritable projet certe lié mais qui se développe à part. Sachant que le projet KSP ancre beaucoup de ses noms dans le folklore celtique, une référence y a étée faite dans le nom et le style du logo. Vis à vis des couleurs, le bleu fait référence à l'espace, tandis que le rouge a uniquement été choisi pour un critère esthétique.

4.4. La communication

Afin de pouvoir suivre en temps réel la fusée depuis le sol, plusieurs questions techniques se sont posées. En effet, cette dernière est calculée pour monter à un peu plus de 2Km, pour une portée de 1Km (CF schéma). De plus, la fusée sera tirée en terrain militaire, donc certaines fréquences nous serons interdites, il importe donc de se mettre d'accord sur quelle bande de fréquence utiliser dans le cas d'une communication par ondes électromagnétiques. L'idéal serait de transmettre un grand nombre de données, ce qui impose un débit assez correct. Enfin, la solution technique trouvée doit consommer le moins possible, et être facilement implémentable dans notre fusée.

Suite à toutes ces contraintes, la communication par ondes radio (leurs grandes longueurs d'ondes leurs permettent une grande portée) semble la plus judicieuse, et parmis ces ondes radio, la communication par LoRaWan (868 MHz en Europe) a été choisie. Comme nous utilisons déjà des cartes LoPy pour notre expérience, un moyen de communication directement supporté par la carte est idéal car il ne demande pas plus de matériel qu'une antenne à ajouter dans la fusée, réduisant ainsi le poid, la consommation électrique, et augmente l'espace disponible.

Les LoPy sont en plus capable de communiquer entre elles par LoRaWAN, donc il suffit d'une LoPy supplémentaire au sol, connectée à un ordinateur hébergeant DolMen, pour communiquer avec notre fusée.

4.4.1. La trame de données

Pour utiliser le logiciel DolMen, il faut mettre en place une trame de données standardisé afin de transmettre, en temps réel (ou bien en pseudo temps réel si on est en mode "offline") les données des différents capteurs. Cela permet d'utiliser le logiciel quelque soit le mode de communication avec la fusée, tant que le moyen d'entrée des données reste le même.

Pour ce dernier, un fichier .txt à été choisi, car il peut aussi bien être lu en mode lecture qu'en mode écriture et ce, en temps réel. De plus, il est accessible à une très grande majorité de plateformes logicielles et codes existants.

Une contrainte importante est la taille de la trame de données, qui doit être la plus légère possible (le LoRaWAN ne permet de transférer que quelques milliers d'octets par seconde). Il faut donc réfléchir à simplifier au maximum la trame, tout en permettant de vérifier à l'arrivée l'intégrité des données.

4.4.2. Architecture embarquée

L'architecture embarquée pour la transmission des données comprend deux cartes lopy :

La première lopy (nommé lopy 1) enverra les données issues des capteurs, par cycle d'horloge et capteur par capteur.

La deuxième carte lopy (nommé lopy 2) enverra une trame composée d'une chaîne de caractères contenant toutes les informations des capteurs.

4.4.3. Format de la trame envoyé au sol

La trame débute par l'ID de la donnée à suivre sur 2 octets.

La donnée est sous forme de X octets.

La trame se termine par ' ;' pour être au format csv.

On recommence l'opération pour chaque capteur.

Le détail des trames se trouve en annexe.

4.5. Cahier des charges

Le BDI souhaiterait que le Projet DolMen répond à un certain nombre de critères :

- Interprétation d'une ou de plusieurs trames différentes et afficher les données obtenues sous diverses formes.
- Pouvoir utiliser le logiciel en tant que simulateur (importation d'une trame de données).
- Le logiciel doit être facile à prendre en mains.
- Le code doit être facilement modifiable.
- Le logiciel doit être parfaitement opérationnel et avoir été testé en conditions réelles.
- Le logiciel doit être modulable en fonction des besoins :
    - Différentes trames de différents capteurs.
    - Un affichage modulable en fonction de l'utilisation.

5. Mise en oeuvre

5.1. Fonctionnalités

5.1.1. L'implémentation des capteurs

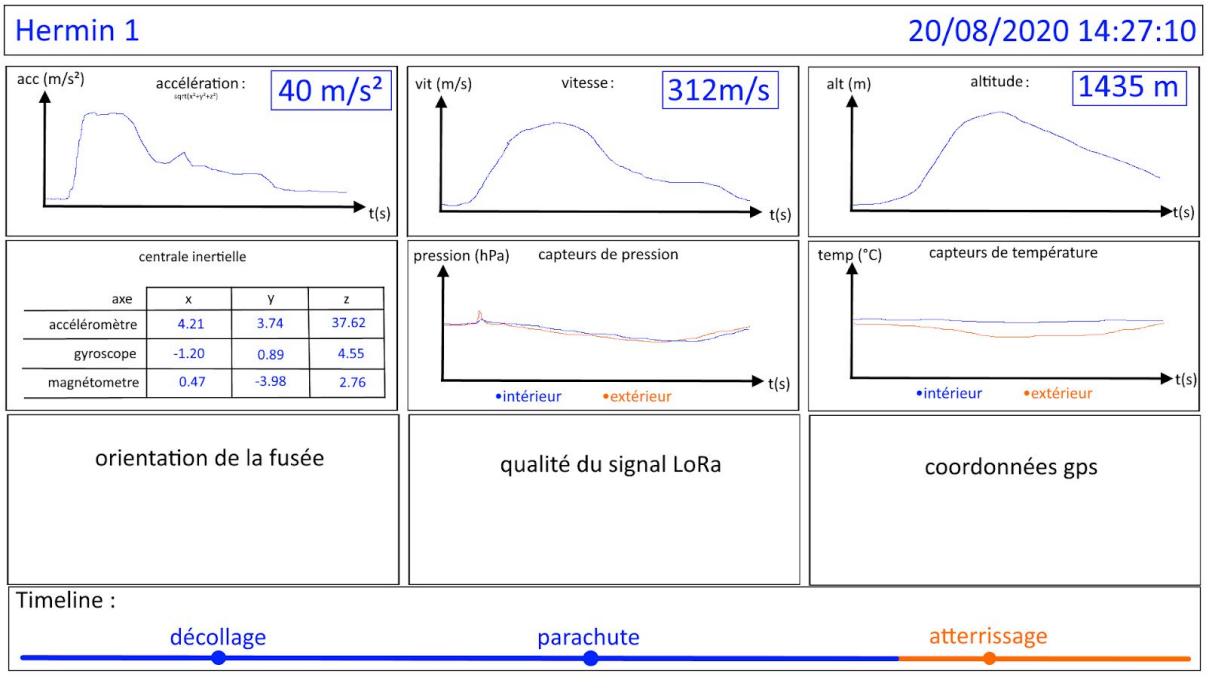
Chaque capteur sera géré par un objet au sein du programme, ce qui peut permettre au logiciel de ne pas avoir à se soucier du type de ce dernier. Un objet capteur permettra de gérer un ensemble de données (celles spécifiques au capteur), de les afficher, ou bien d'y appliquer un traitement.

L'ajout de nouveaux capteurs pourra se faire simplement, depuis le logiciel. En effet, ajouter un nouveau capteur dans le logiciel génèrera une nouvelle classe spécifique à ce capteur dans un nouveau fichier, dont il faudra compléter la méthode "décoder" de la classe, qui est propre à chaque capteur.

5.1.2. L'interface

Lorsque l'utilisateur lance le logiciel, il peut choisir entre plusieurs modes d'utilisation:

- le mode tir, permettant de suivre en temps réel les différentes données traitées, que ces dernières soient sous la forme de courbes ou bien de texte.
- le mode administrateur, permettant la gestion des différents capteurs, des fenêtres et des différents graphes.



Voici un exemple d'interface telle qu'elle pourrait être dans le mode tir, dans une configuration avec 9 capteurs. Cette apparence pourrait être amenée à évoluer au cours de la réalisation du logiciel. De plus, des configurations avec plus ou moins de capteurs pourront être choisies.

5.1.3. L'ajout et modification de capteur

S'il se veut évolutif, le logiciel doit permettre d'ajouter simplement de nouveaux capteurs ou de les modifier, et ce sans aller rajouter des lignes de code.

Nous aurons donc une fenêtre dans la partie gestion des capteurs du mode administrateur. A partir de celle-ci, nous pourrons créer un capteur en précisant le nom de celui ci ainsi qu'un identifiant permettant de de le repérer dans la trame de données. Nous pourrons via cette fenêtre modifier les données d'un capteur comme son nom ou encore son identifiant.

5.1.4. Le fichier de configuration

Un fichier du logiciel ('config.txt') permettra d'enregistrer la configurations du logiciel, et de sauvegarder la configuration de l'interface et des graphiques. Il permettra aussi de sauvegarder les noms, les données et les identifiant des différents capteurs.

Ce fichier sera importé dès l'ouverture du logiciel en fonction du mode choisi.

5.1.5. Le compte rendu de fin de vol

Un bouton sur le mode tir permettra de terminer le traitement de la trame de données. Une fois ce bouton cliqué, une fenêtre popup de confirmation s'affiche de manière à éviter de terminer le traitement par erreur. Un fichier .csv (pouvant donc être lu par un tableur) sera créé contenant les données et le moment auxquelles elles correspondent. Un tableur fourni avec le logiciel pourra lire ce .csv et afficher les courbes et graphes associés.

5.2. Scénarios d'utilisation

Scénario nominal (SN):

1. On ouvre le logiciel.
2. On choisit son mode d'utilisation (tir SA2, admin SA1 ou quitter SN étape 4).
3. On exporte les données et on génère un bilan de fin de vol en .csv.
4. On ferme le logiciel.

Scénario alternatif 1 (SA1): Mode Admin:

1. On choisit soit de gérer les fenêtres (SA1.1), soit de gérer les capteurs (SA1.2), soit gérer les graphes (SA1.3).
2. On sauvegarde les modifications et on retourne au scénario nominal SN à l'étape 2.

SA1.1: gestion des fenêtres:

1. On peut choisir d'ajouter ou de modifier les fenêtres.
2. On retourne au SA1 étape 1.

SA1.1: gestion des capteurs:

1. On peut choisir d'ajouter ou de modifier les capteurs, par le biais de popup.
2. On retourne au SA1 étape 1.

SA1.1: gestion des graphes:

1. On peut choisir quelle graphe est à quelle position.
2. On retourne au SA1 étape 1.

SA2: Mode Tir:

1. On choisit entre le mode tir en ligne ou hors ligne.
2. Si on est en mode hors ligne, on sélectionne notre fichier.
3. SA3.
4. On retourne au scénario nominal SN étape 3.

SA3: traitement des données:

1. Décodage de la trame.
2. Traiter les informations et SA4 si nécessaire.
3. Afficher les informations et enregistrer les données.
4. Retour au SA2 étape 3.

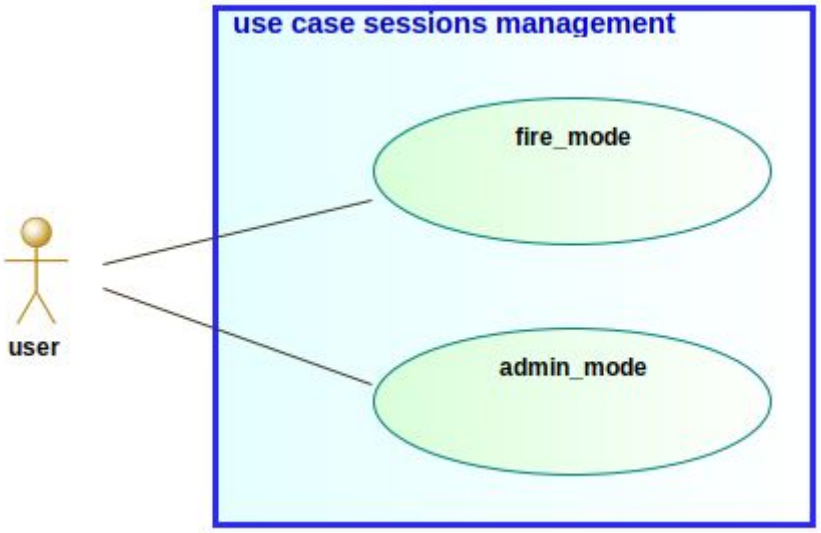
SA4: gestion des erreurs:

1. Identifier une erreur par rapport à une consigne ou de mesure d'un capteur.
2. Vérifier les erreurs, avec le choix ou non d'en ignorer certaines.
3. Si besoin, afficher une fenêtre avec l'erreur.
4. Sauvegarder l'erreur dans un fichier《log》(séparé de la sauvegarde des données).
5. Relancer le logiciel (SN étape 1) ou retour à l'état précédent.

5.3. Diagrammes UML

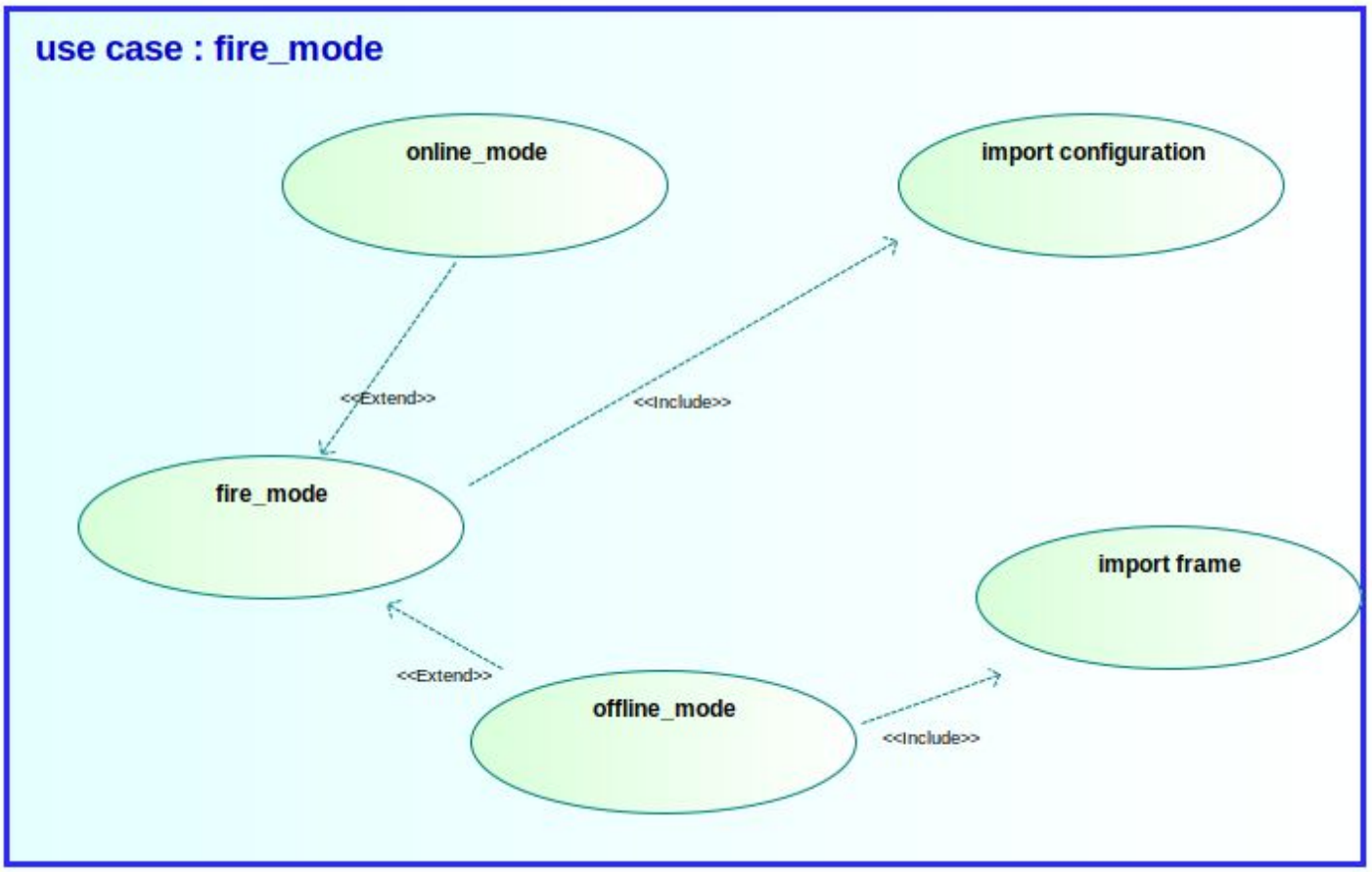
5.3.1. Cas d'utilisation

5.3.1.1. Gestion des sessions



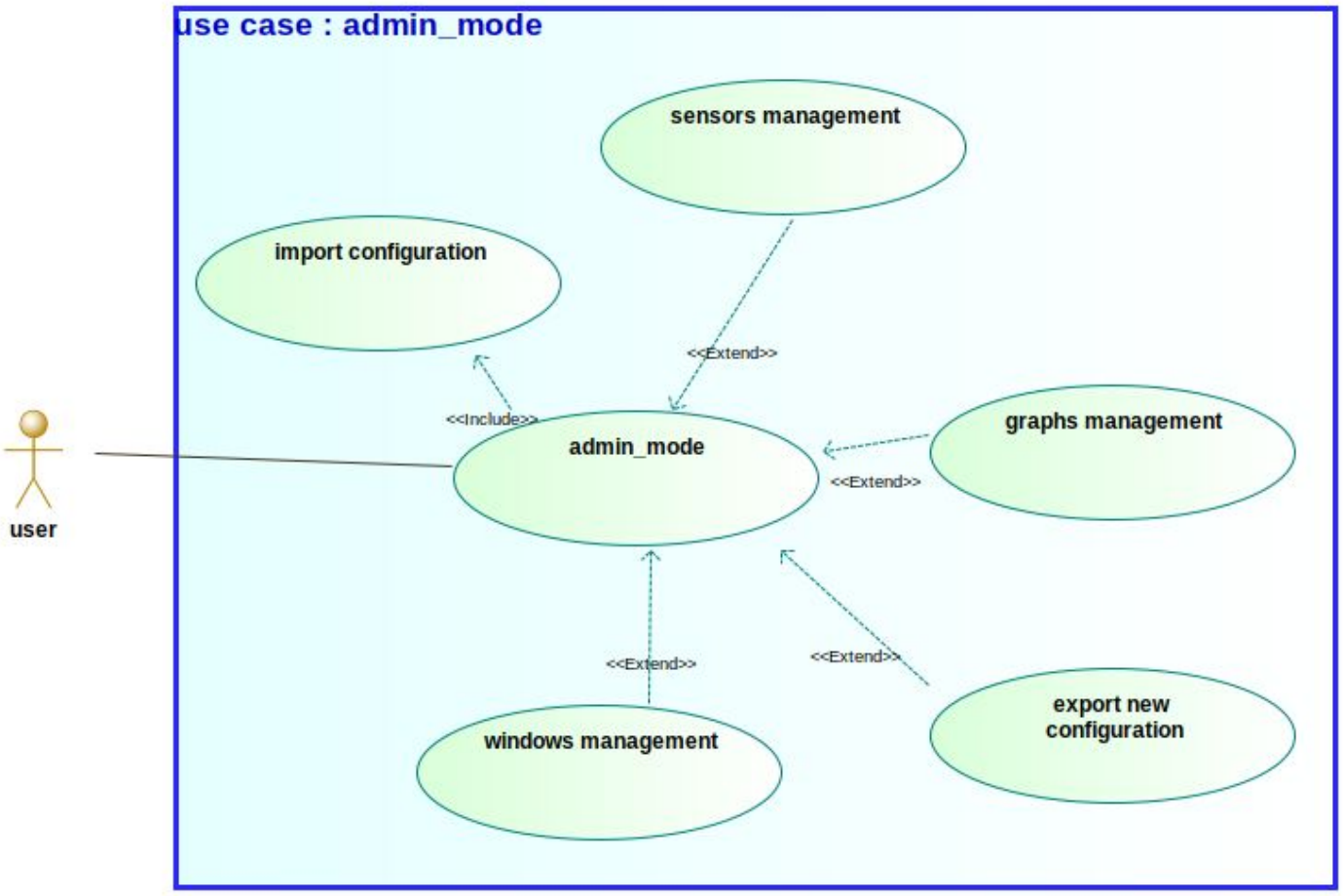
A l'ouverture de Dolmen, l'utilisateur peut choisir le mode dans lequel il veut entrer : le mode de "tir" ou le mode administrateur à partir duquel il pourra modifier les paramètres de capteurs, en ajouter de nouveaux, modifier les fenêtres d'affichage et gérer les graphes.

5.3.1.2. Mode “tir”



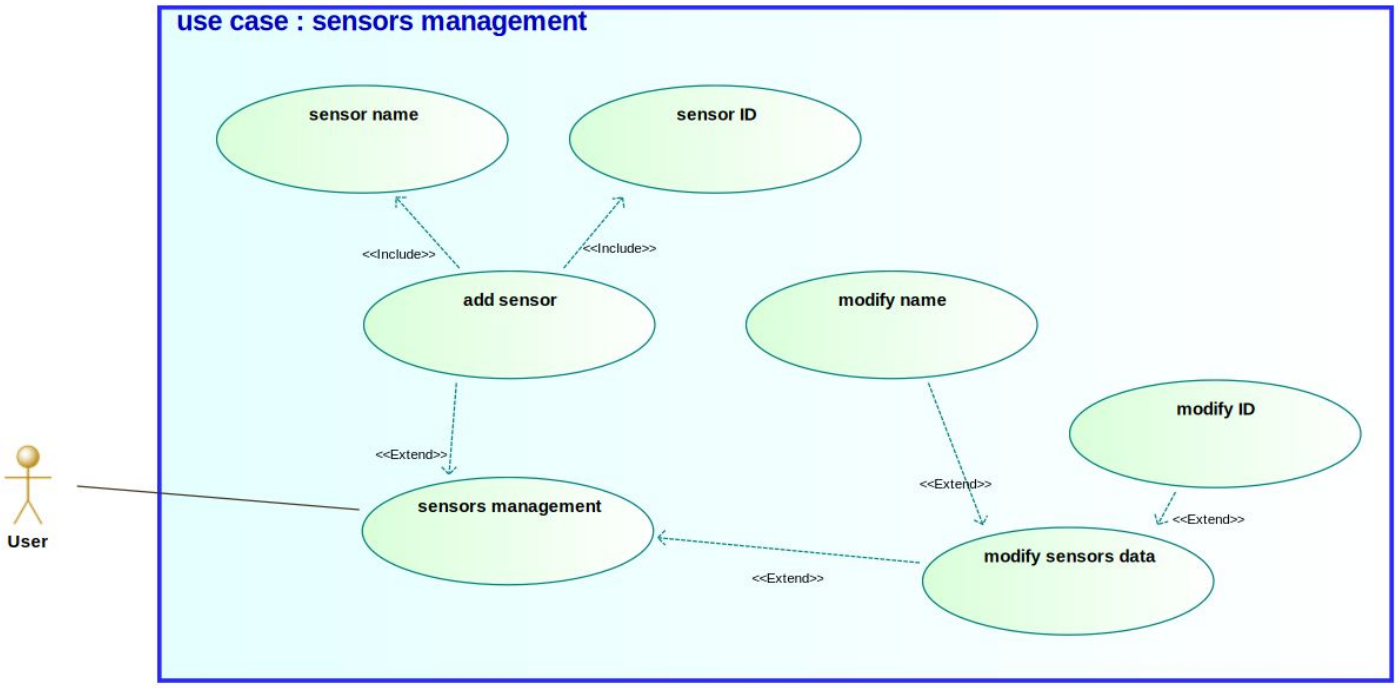
Une fois dans le mode tir, il sera possible soit d’utiliser le mode en ligne (avec une fusée connectée, et donc une trame qui s’écrit au fur et à mesure), soit le mode hors ligne (ou il faudra donc importer un fichier .txt contenant une trame). L’ouverture du mode tir réalisera l’import du fichier de configuration du logiciel, qui donne toutes les informations sur l’interface.

5.3.1.3. Mode administrateur



Une fois dans le mode administrateur, l'utilisateur peut soit gérer les fenêtres (nombre et position), soit les graphiques (la manière dont les données seront affichées), soit les capteurs (en ajouter ou en retirer). Tout cela sera enregistré dans un fichier de configuration (format .txt). L'ouverture du mode admin importera aussi le fichier de configuration, qui sera modifié au gré des réglages choisis.

5.3.1.4. Gestion des capteurs

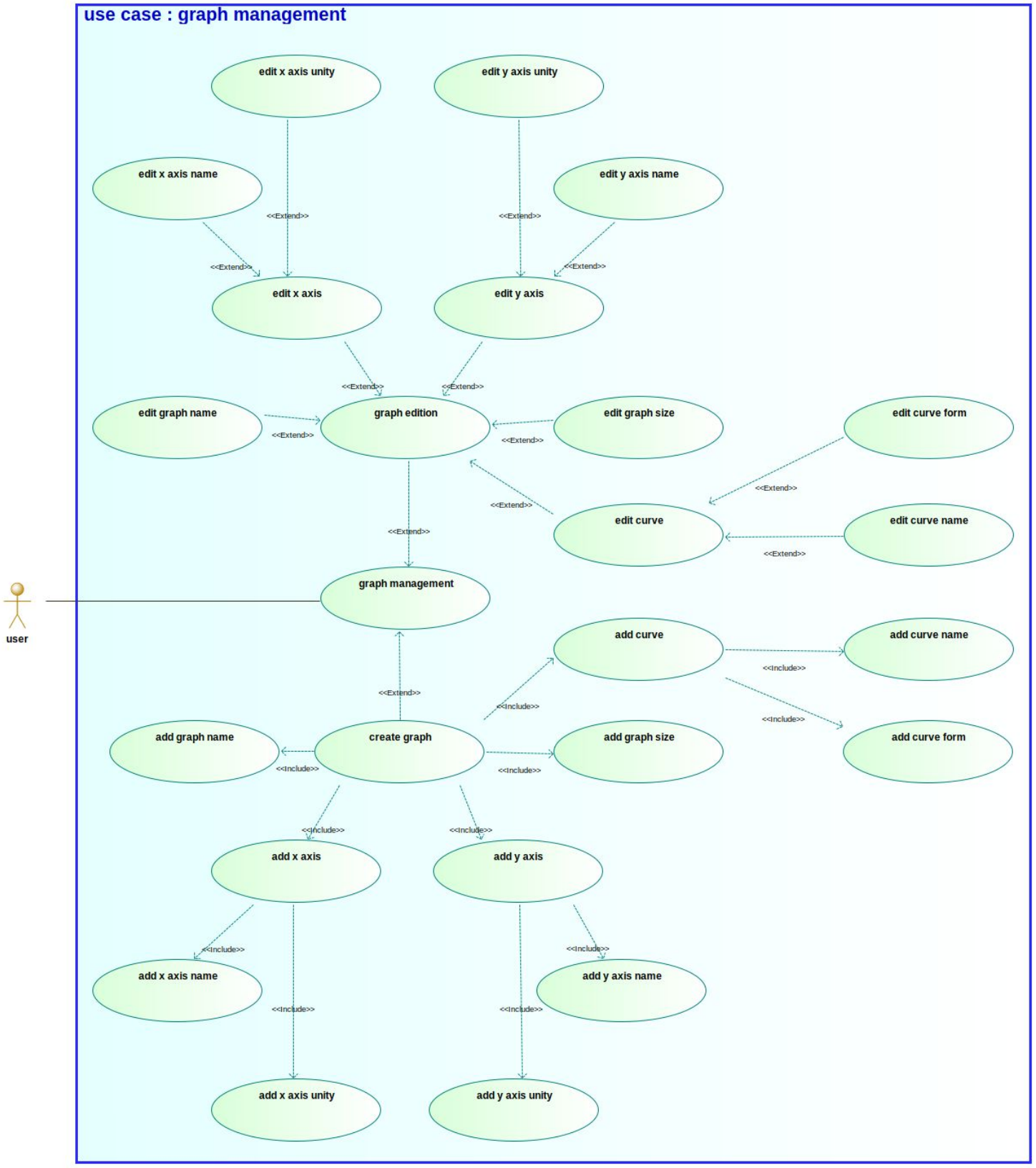


Une fois dans l'interface de gestion des capteurs, on pourra ajouter un capteur (en lui donnant un nom et un ID), ou si il le souhaite, modifier les données d'un capteur (nom ou ID).

Si l'utilisateur veut cependant ajouter des fonctions spécifiques à ce capteur, il devra aller dans le fichier .hpp généré automatiquement à l'ajout du capteur.

La configuration des capteurs sera elle aussi sauvegardée dans un fichier texte.

5.3.1.5. Gestion des graphes

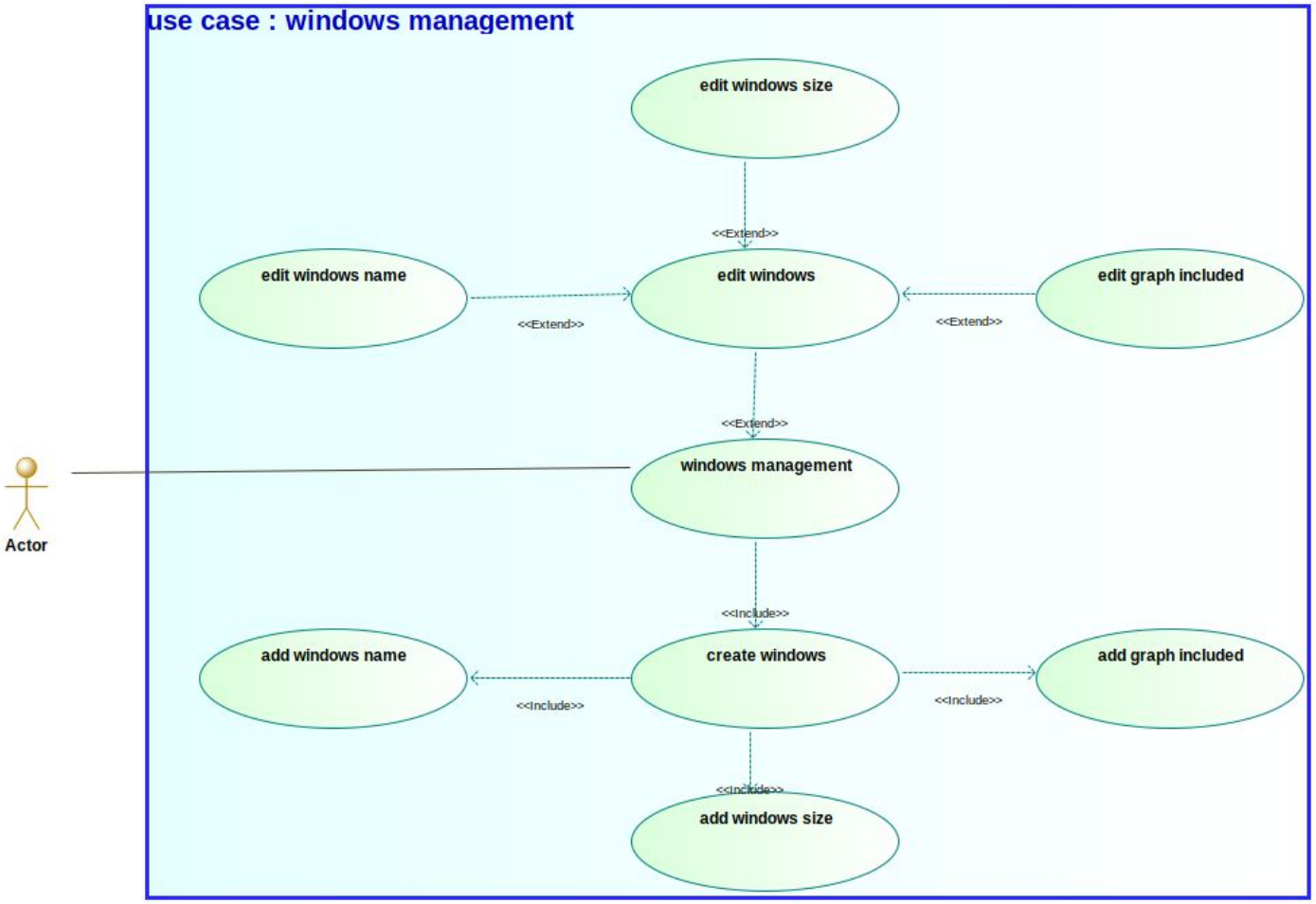


Au sein du code du logiciel, de nombreuses options seront possibles pour gérer les graphes, qui restent le point central du logiciel, puisqu'ils permettent de gérer les données.

Parmis la gestion des graphes, il sera possible soit de créer des graphes, soit de modifier des graphes existants.

Pour chacune de ces options, les points ajustables sont les mêmes. En effet, il est possible de jouer sur le nom et la taille du graphe, les courbes affichées (correspondent aux données, ces courbes auront un nom et une forme). On peut aussi jouer sur les axes en leur donnant un nom et une unité.

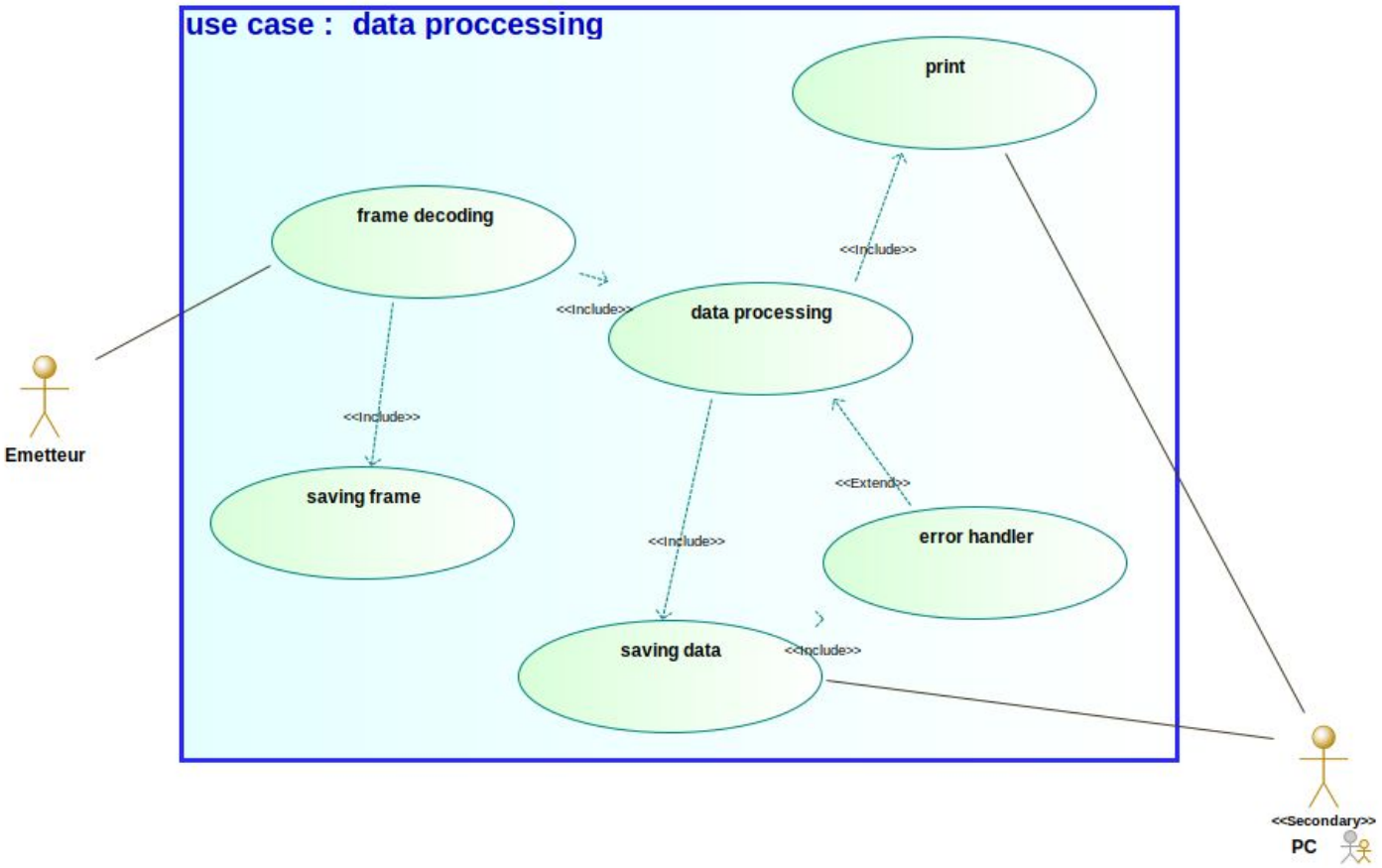
5.3.1.6. Gestion des fenêtres



Comme pour les graphes, la gestion des fenêtres se fait soit par la création de nouvelles fenêtres, soit par la modification de fenêtres existantes. Dans chaque fenêtre, on pourra jouer sur les graphes (en en ajoutant/retirant), sur le nom de la fenêtre, et sa taille.

Une amélioration possible serait un choix de fenêtres préconfigurées, afin de simplifier l'utilisation du mode admin. L'utilisateur n'aurait qu'à choisir le nombre de graphes qu'il veut afficher, et pour chaque "case" ainsi créée, il choisira quelle donnée afficher dedans.

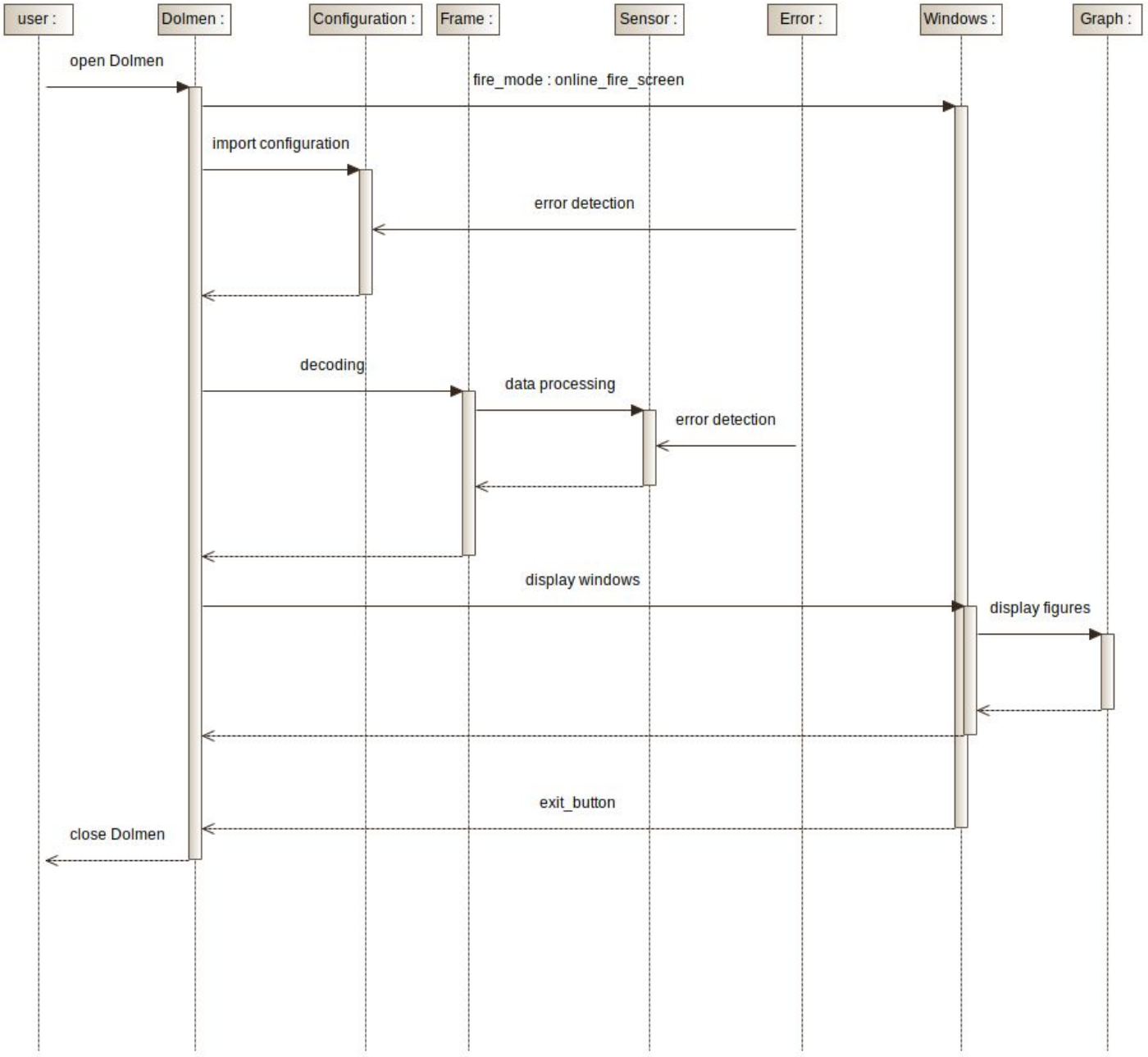
5.3.1.7. Gestion des données



Lorsque que dolmen reçoit une trame émise, il l'a décode tout d'abord avant de sauvegarder la trame reçue dans un fichier de sauvegarde. Ensuite vient le traitement des données, celle ci sont elles aussi sauvegardées dans un autre fichier de sauvegarde. Ces données sont ensuite affichées. Un algorithme sera exécuté afin de controle et vérifier les erreurs des données au niveau de la sauvegarde (fichier introuvable, ...).

5.3.2. Communication

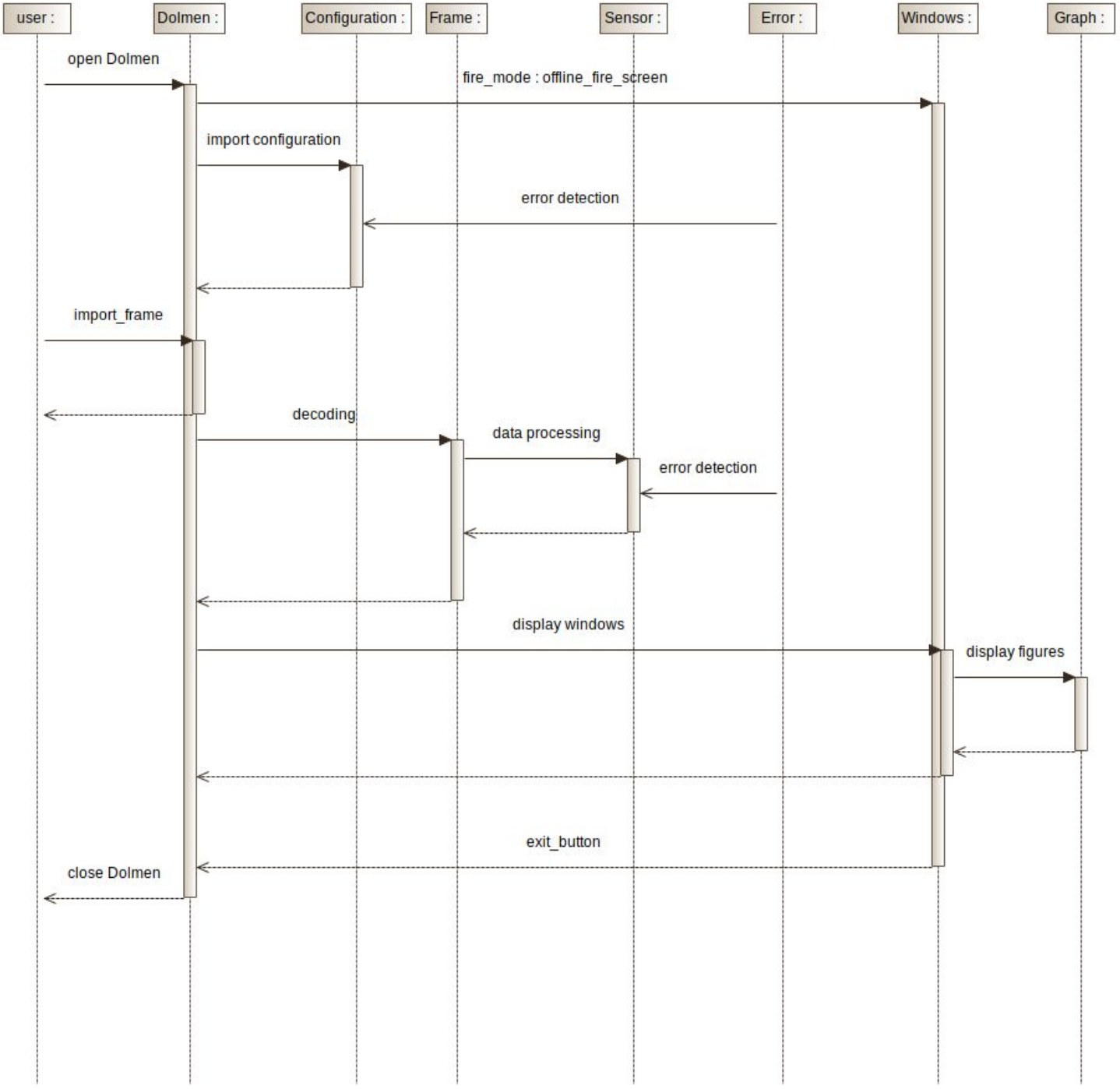
5.3.2.1. mode “tir” (mode réel)



A l’ouverture du logiciel en mode tir en ligne par l’utilisateur, le logiciel s’ouvre dans une fenêtre, et importe le fichier de configuration. La détection des erreurs vérifie que cette importation se passe bien, tout comme elle surveille l’action suivante: le décodage des informations par le logiciel, puis leur homogénéisation dans les classes capteurs (afin de rendre les unités affichables).

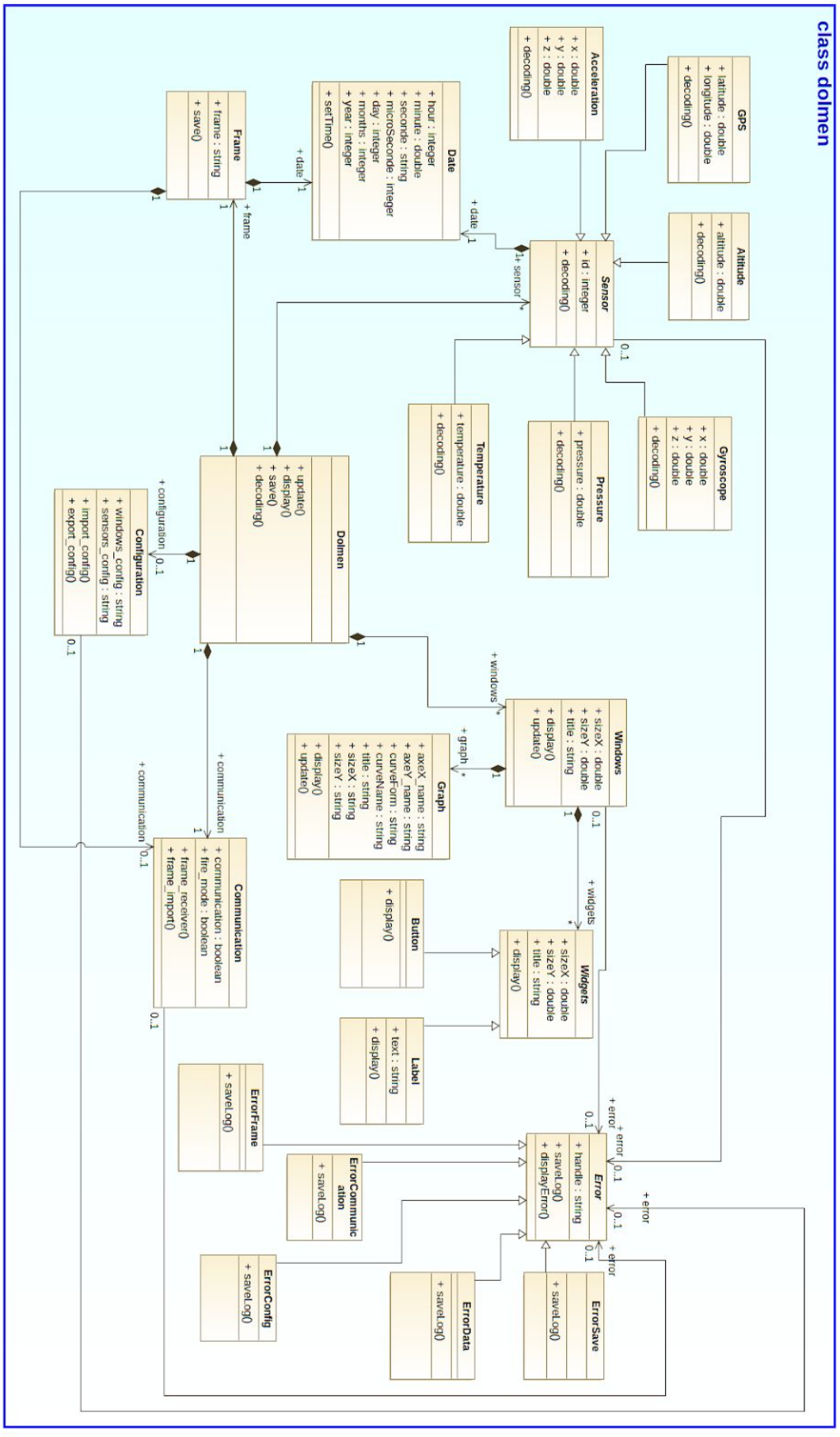
Les données sont ensuite renvoyées dans le logiciel, qui va les afficher au travers des graphes.

5.3.2.2. mode “tir” (mode simulation)



Le fonctionnement ici est presque identique au mode de tir en ligne (5.3.2.1), mais avec un import de la trame de données après l’import du fichier de configuration et avant le décodage de la trame.

5.3.3. *Classes*


class dolmen
GPS
+ latitude : double
+ longitude : double
+ decoding()
Acceleration
+ x : double
+ y : double
+ z : double
+ decoding()
Altitude
+ altitude : double
+ decoding()
Frame
+ frame : string
+ save()
Date
+ hour : integer
+ minute : double
+ seconde : string
+ microSeconde : integer
+ day : integer
+ months : integer
+ year : integer
+ setTime()
Sensor
+ id : integer
+ decoding()
Temperature
+ temperature : double
+ decoding()
Pressure
+ pressure : double
+ decoding()
Gyroscope
+ x : double
+ y : double
+ z : double
+ decoding()
Dolmen
+ update()
+ display()
+ save()
+ decoding()
Configuration
+ windows_config : string
+ sensors_config : string
+ import_config()
+ export_config()
Windows
+ sizeX : double
+ sizeY : double
+ title : string
+ display()
+ update()
Graph
+ axeX_name : string
+ axeY_name : string
+ curveForm : string
+ curveName : string
+ title : string
+ sizeX : string
+ sizeY : string
+ display()
+ update()
Widgets
+ sizeX : double
+ sizeY : double
+ title : string
+ display()
Button
+ display()
Label
+ text : string
+ display()
Communication
+ communication : boolean
+ fire_mode : boolean
+ frame_receiver()
+ frame_import()
Error
+ handle : string
+ saveLog()
+ displayError()
ErrorFrame
+ saveLog()
ErrorCommunication
+ saveLog()
ErrorConfig
+ saveLog()
ErrorData
+ saveLog()
ErrorSave
+ saveLog()
+ date
+ frame
+ sensor
+ configuration
+ communication
+ graph
+ windows
+ widgets
+ error

Nous utilisons 3 héritages :

Le premier avec une classe Sensor, qui sera composé d'un identifiant et d'une méthode "decoding". Mais comme chaque capteur aura un décodage différent, nous créons donc pour chaque capteur sa propre classe dérivant de la classe Sensor mais en redéfinissant la fonction "decoding" et en ajoutant des attributs spécifiques au capteur concerné.

Pour le deuxième héritage, nous faisons de même avec la classe Error, qui généralise les classes ErrorSave, ErrorData, ErrorConfig, ErrorCommunication et ErrorFrame. En effet toutes ces classes dérivées de la classe Error ont en commun un attribut handle permettant de décrire l'erreur, et deux méthodes : une méthode pour afficher cette erreur et une autre pour la sauvegarder dans un fichier log. Cette dernière fonction est redéfinie dans chaque classe dérivé de Error qui permettra de sauvegarder l'erreur dans un format personnalisé propre à chaque type d'erreur.

Pour le dernier héritage, les classes Button et Label dérivent de la classe Widgets. En effet la classe Button permet de créer des boutons et Label, d'afficher des fenêtres Popup. Elles ont des attributs et des fonctions en communs comme la taille, le titre ou encore la fonction display (qui sera redéfinie selon si c'est un bouton ou une popup). Mais elles possèdent des attributs propres à elles même comme pour la classe Label avec l'attribut "text" qui permettra d'afficher le texte qui sera présent sur la fenêtre popup.

La classe Dolmen aura pour fonction de gérer et décoder la trame (en utilisant la fonction "decoding" propre à chaque capteur). Elle sera composée de fenêtres (classe Windows) elles même composées de graphes et de Widgets.

La classe Windows aura comme attributs un titre et une taille et aura deux méthodes : une méthode pour afficher la fenêtre et une autre pour la mettre à jour.

La classe Graph aura comme attributs un titre, une taille, des axes, et une courbe et aura deux méthodes : une méthode pour afficher le graph et une autre pour le mettre à jour.

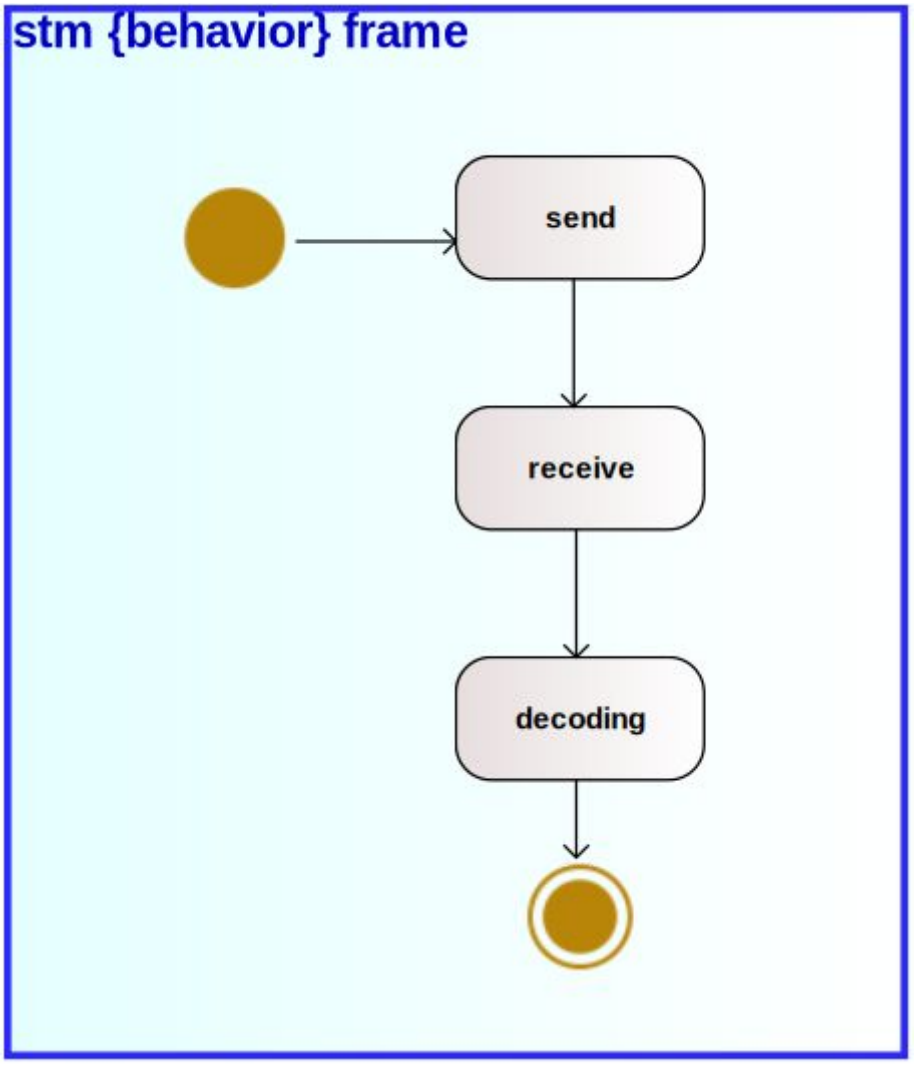
La classe Dolmen est composée d'une classe Communication qui permettra de gérer la réception des trames et d'activer ou non la réception des trames. On pourra importer notre propre trame, nécessaire pour mode de tir simulé. En booléen fire_mode permettra de définir dans quel mode de tir nous nous trouvons.

La classe Dolmen est aussi composée d'une classe configuration permettant d'importer et d'exporter la configuration.

La classe Frame permettra la sauvegarde de la trame reçue et sera composée de la classe Date afin d'acquérir l'heure.

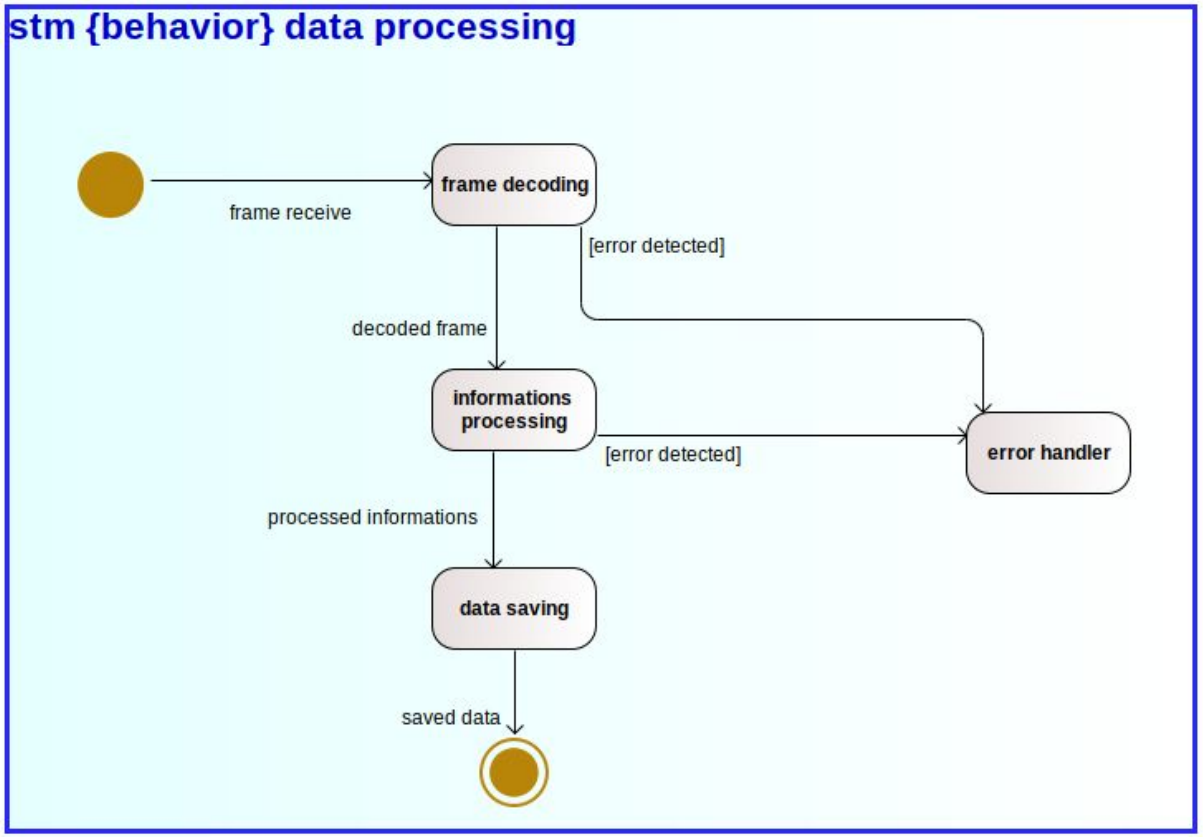
5.3.4. Machine à états

5.3.4.1. Traitement de la trame



Une trame va être traitée de manière très simple: elle est envoyée par la fusée, reçue par le logiciel, puis décodée.

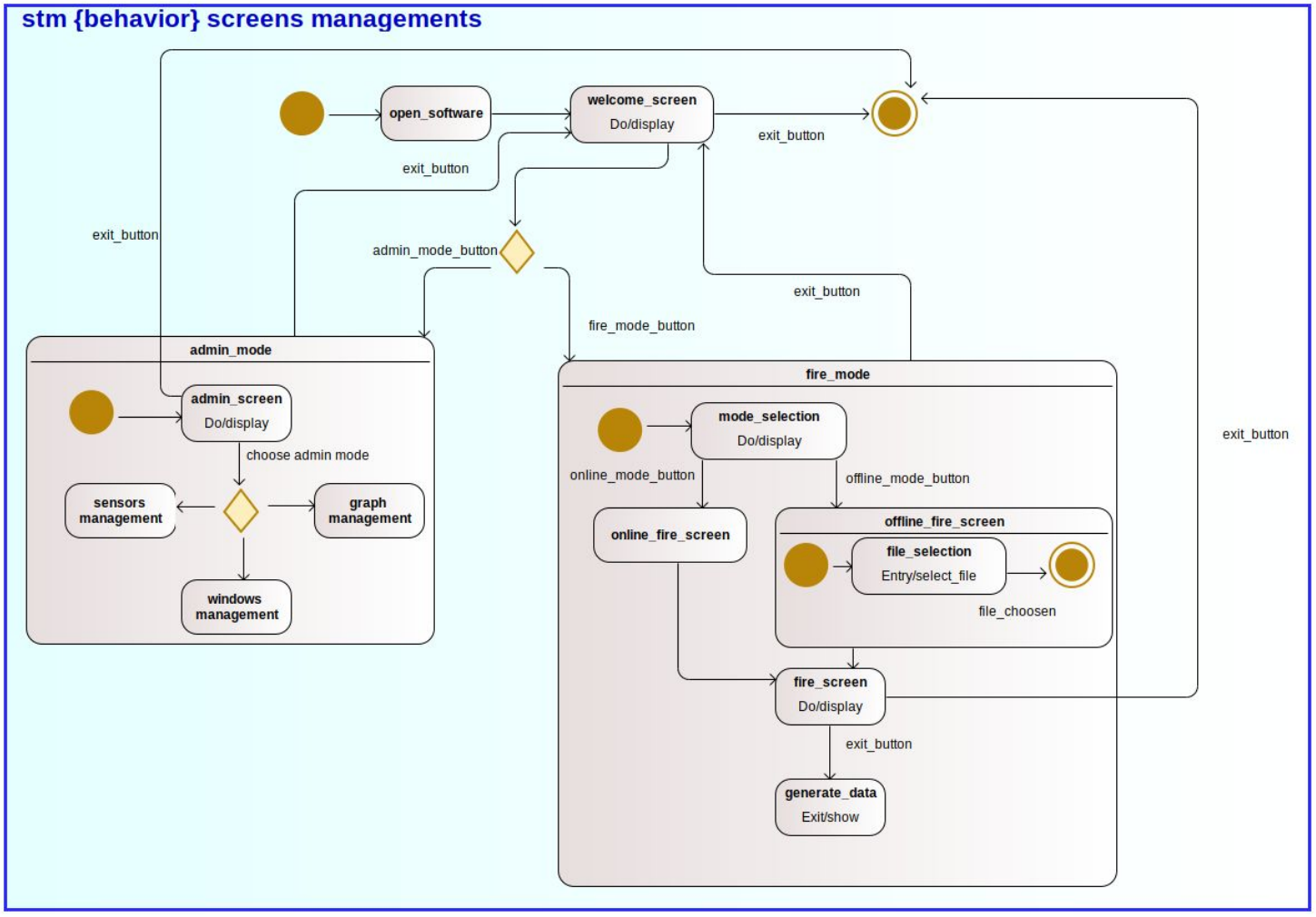
5.3.4.2. Traitement des données



Lorsque la trame de données est reçue, elle est lue ligne par ligne par le logiciel. Les données sont traitées (assignation à un timecode et au capteur originel), puis enregistrées.

Lors de la lecture et du traitement des informations, la gestion des erreurs veille au bon déroulement des traitements.

5.3.4.3. Gestion des écrans



Lorsque l'on ouvre le logiciel, on se retrouve face à un écran d'accueil, nous demandant de choisir entre le mode administrateur (admin_mode) et le mode tir (fire_mode). Il est toujours possible de fermer le logiciel grâce à un bouton de sortie (exit_button).

Le mode tir ouvre une fenêtre permettant de choisir si on tire en mode connecté (la fusée est connectée en temps réel), ou en mode hors ligne (on simule un tir avec un fichier .txt de trame de données déjà rempli, une fenêtre popup s'ouvre donc, nous demandant de sélectionner le fichier à étudier).

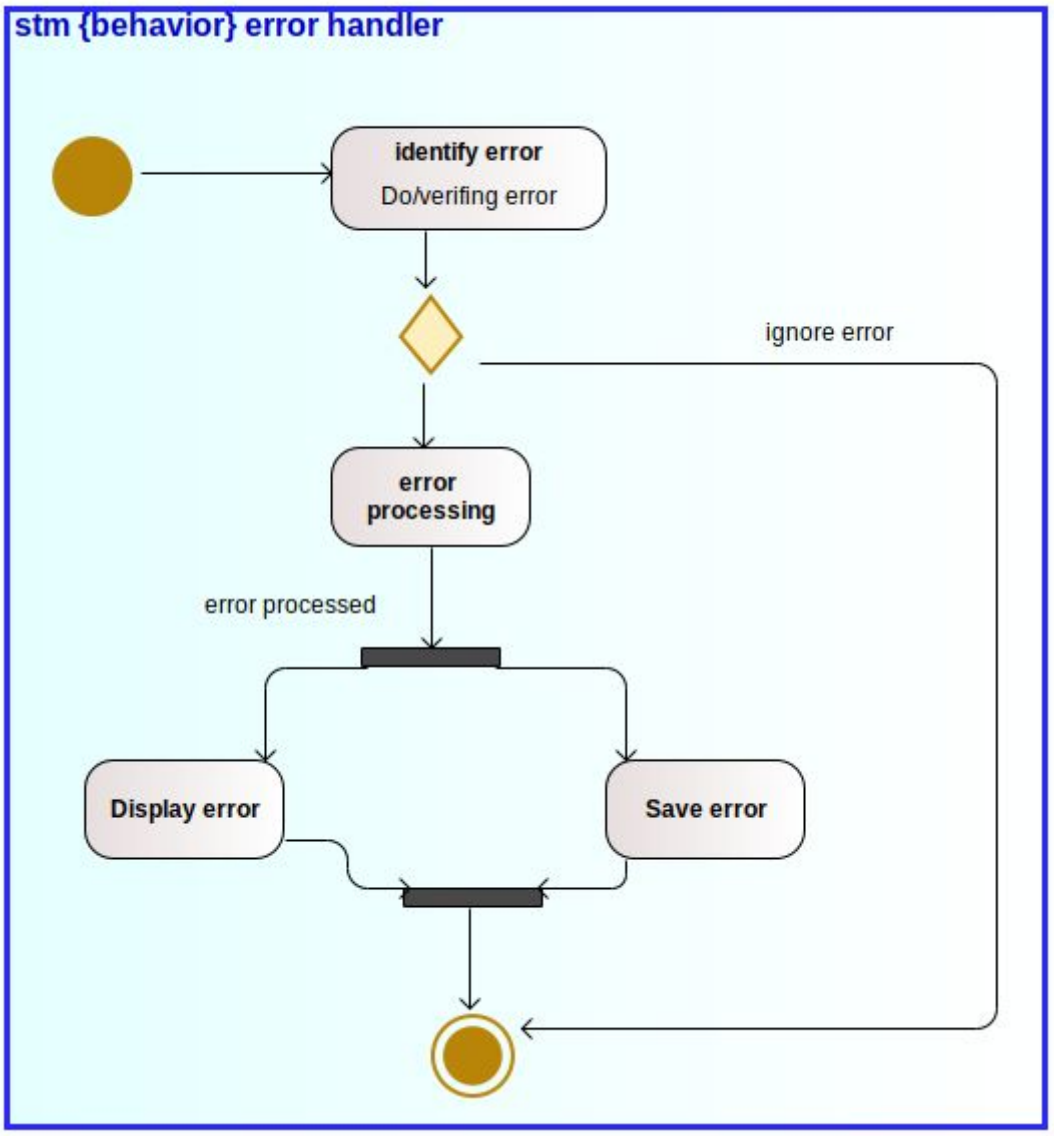
Une fois le mode de tir sélectionné, on entre sur l'écran de tir à proprement parler, ou il suffit d'observer nos données, et éventuellement de venir bouger sur les graphiques (zoom, déplacement de l'origine). Un bouton de sortie permet de générer le fichier .csv servant de rapport de vol.

Dans le mode administrateur, il est possible de choisir parmis 3 fenêtres:

- Gérer les fenêtres
- Gérer les capteurs
- Gérer les graphes

Dans chacune de ces fenêtres, il est possible d'ajouter et/ou de modifier des éléments.

5.3.4.4. Gestion des erreurs

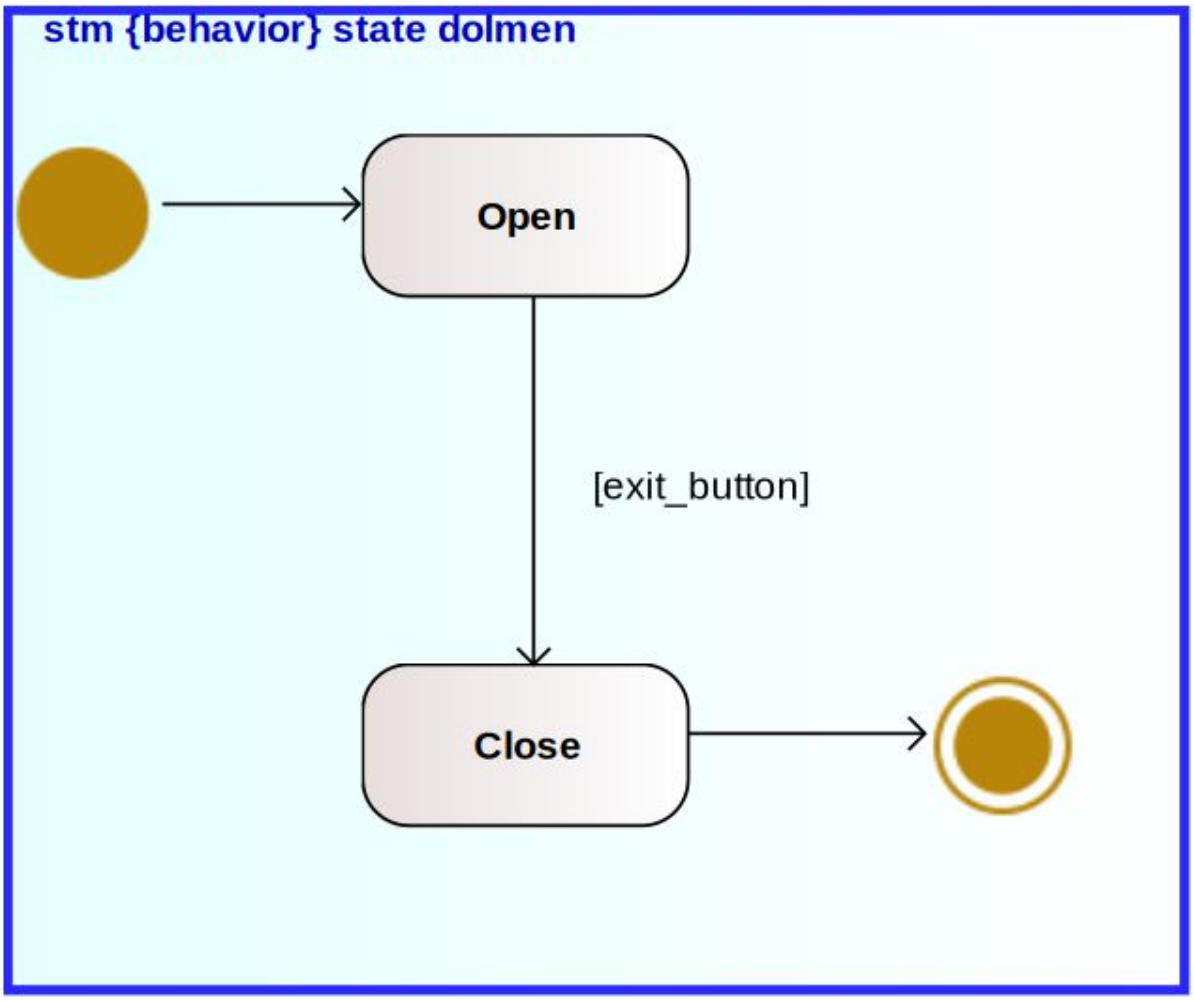


Lorsqu'une erreur est detectée, on peut soit l'ignorer (si elle est jugée non importante), soit la traiter.

Lors du traitement de l'erreur, soit c'est une erreur mineure (une donnée avec une valeur absurde par exemple), elle est simplement enregistrée dans un fichier log d'erreur.

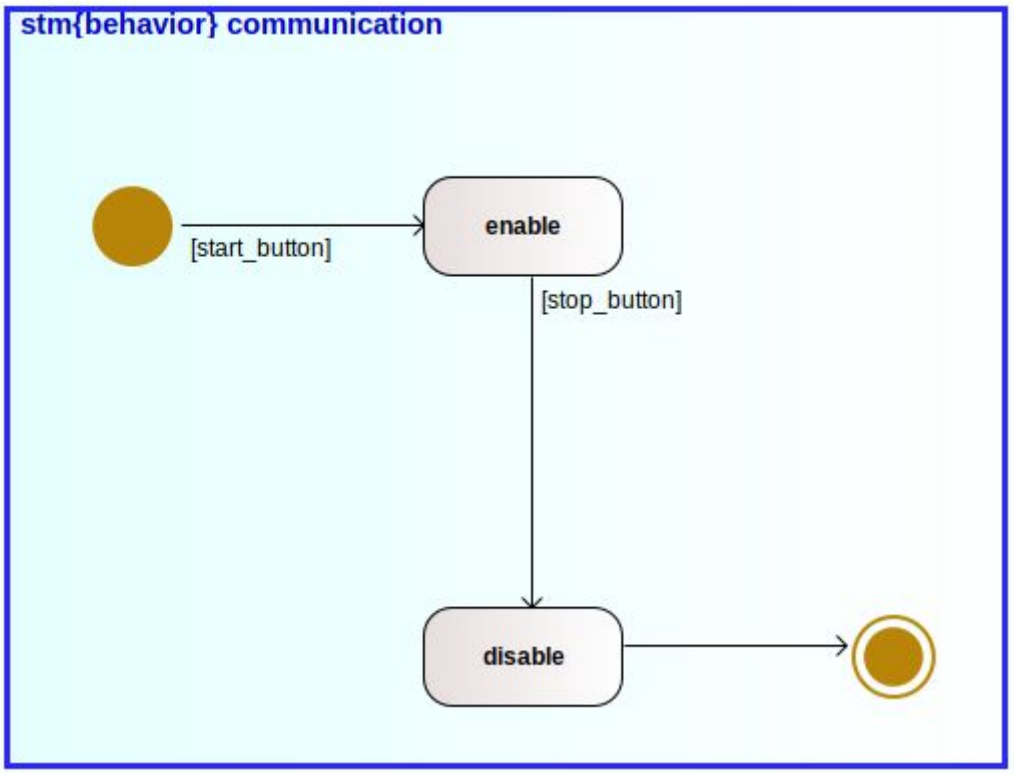
Si l'erreur est majeure (par exemple un plantage du logiciel ou une déconnection de la fusée), elle sera affichée à l'écran en plus d'être sauvegardée.

5.3.4.5. Etat logiciel



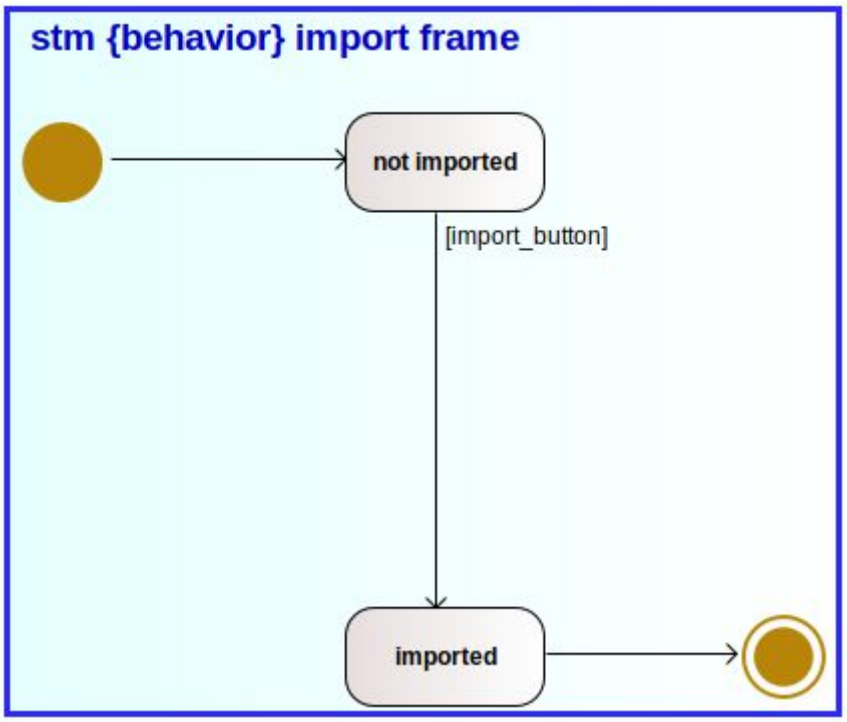
Le logiciel s'ouvre, et un bouton de sortie (exit_button) permet d'en sortir.

5.3.4.6. Communication avec l'émetteur



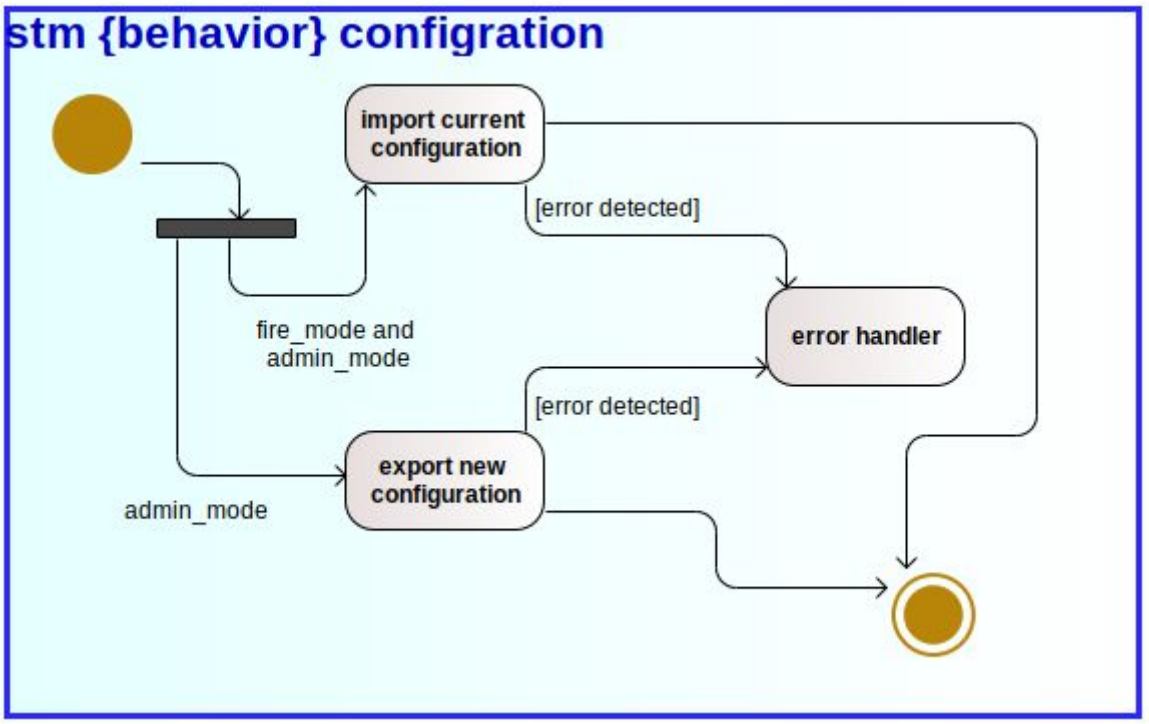
Lorsque le logiciel est lancé (mode tir réel), la communication avec la fusée (lecture de la trame) est établie, et cette dernière se coupe si le bouton stop est pressé.

5.3.4.7. Importation d'une trame



Lorsqu'on l'on est dans le mode tir mais en mode simulation, on choisit d'importer une trame de donnée, ce qui se fait depuis un fichier externe en .txt. Pour cela, il faut cliquer sur un bouton 'import_button'.

5.3.4.8. Configuration

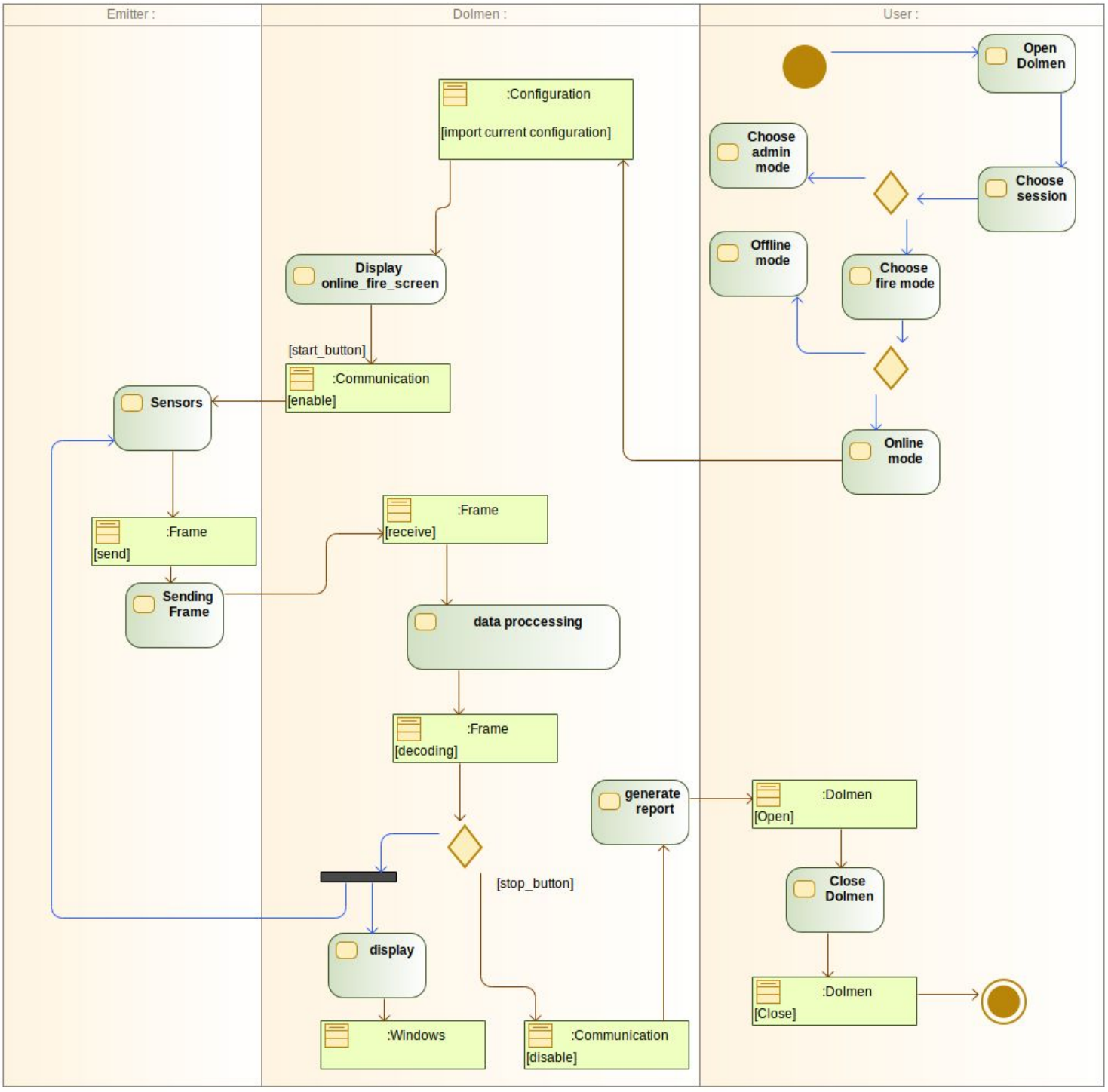


Le fichier de configuration sera importé lors de l'ouverture du mode tir ou du mode admin, mais le mode admin pourra, contrairement au mode tir, écrire dans ce fichier.

Que l'on soit en import ou en export du fichier config, une gestion des erreurs sera faite.

5.3.5. Activités

5.3.5.1. Exemple mode “tir” (mode réel)



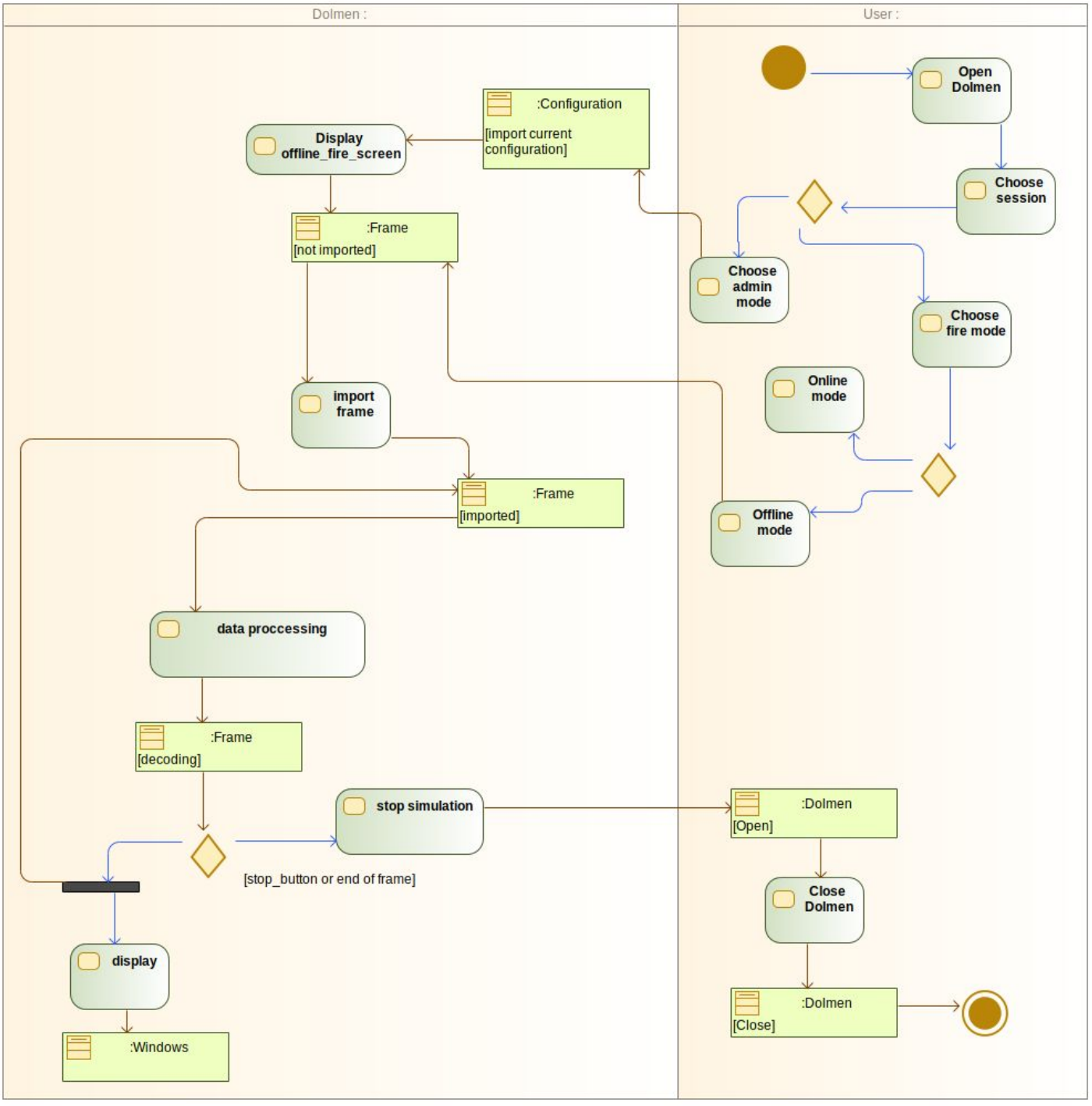
A l'ouverture du logiciel, l'utilisateur décide s'il veut aller en mode tir ou en mode admin. Le cas étudié ici est celui du mode tir.

Une fois le mode choisi, c'est au logiciel de jouer, comme ici dans le cas du mode "online" (ou la fusée est connectée).

Les capteurs vont donc envoyer une trame de donnée au logiciel, qui va la traiter en temps réel (séparation des éléments par capteur et affichage des données de ces derniers).

Lorsque le bouton d'arrêt est appuyé, on génère le rapport (au format .csv), et on peut fermer le logiciel.

5.3.5.2. Exemple mode “tir” (mode simulation)

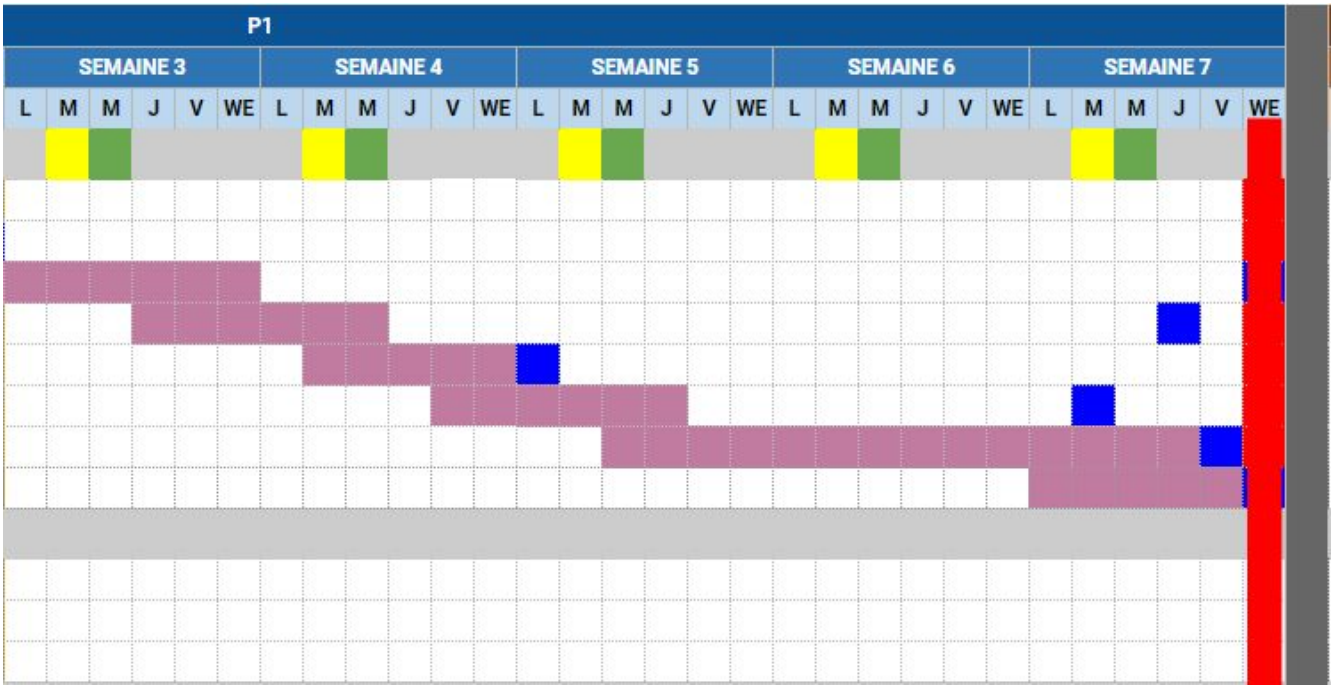


Lorsque l’on tire en mode hors ligne, on ouvre le logiciel, choisit sa session (ici mode tir), puis on choisit le mode hors ligne. Il faut alors importer une trame de donnée (fichier .txt). Le logiciel traite ensuite la trame de données (affichage des graphes à l’écran en fonction du temps).

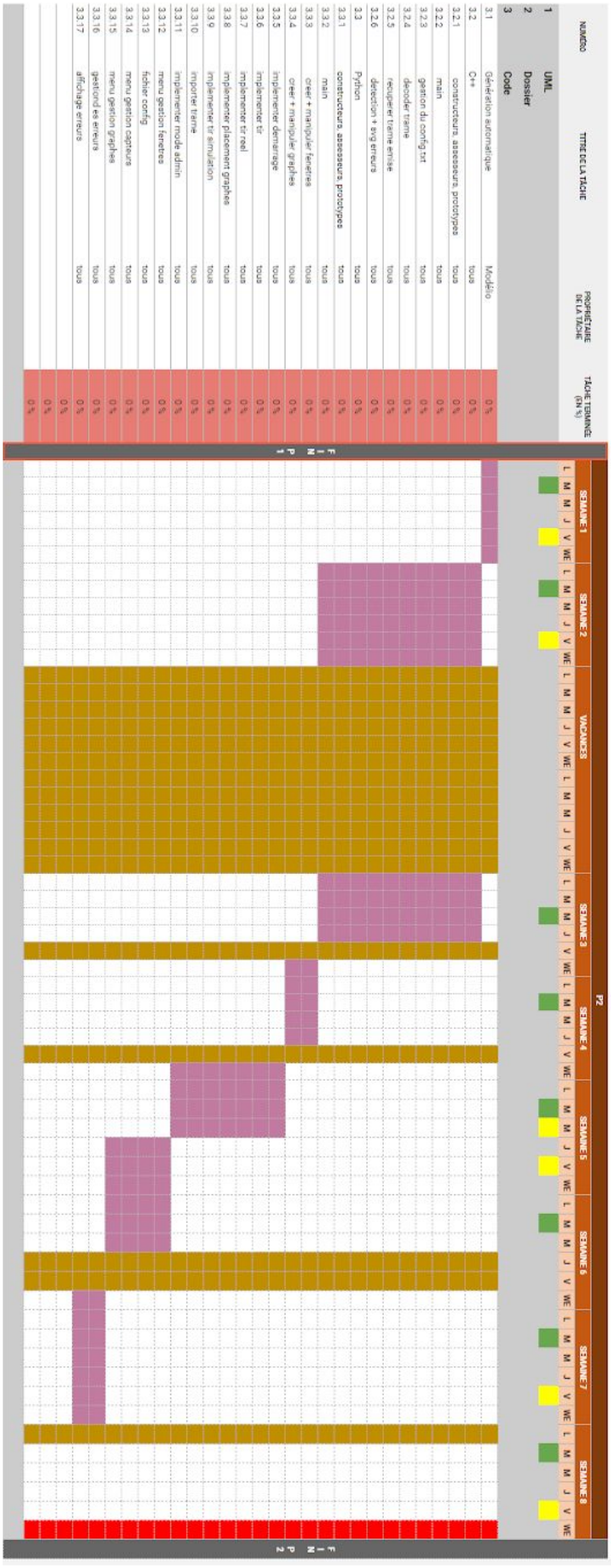
5.4. Planification

5.4.1. P1 (semaines 1 à 7)

TITRE DE LA TÂCHE	PROPRIÉTAIRE DE LA TÂCHE	TÂCHE TERMINÉE (EN %)
UML		
Texte	tous	100 %
Cas d'utilisations	tous	100 %
séquence	tous	100 %
classes	tous	100 %
stm	tous	100 %
activites	tous	100 %
homogenéisation/correction des diagrames	Tous	100 %
Correction	Tous	100 %
Dossier		
Squelette	Timothée	100 %
Ecriture du texte	Tous	90 %
modelio et ajout de l'uml	Axel	90 %



5.4.2. P2 (semaines 8 à 14)



5.5. Choix des langages de programmation

Nous allons coder DolMen en utilisant deux puissants langages de programmation : Python et C++. Nous avons les avons choisi car ils sont à la fois puissants et très répandus. Cela permettra au logiciel de pouvoir être repris et développé par la plupart des programmeurs, et notre code pourra être aisément compris.

Chacun de ces langages aura une fonction bien précise : nous utiliserons le Python afin de créer l'interface graphique, tandis que le C++ servira pour le décodage de la trame et le traitement des données. La gestion des erreurs liées aux trames ou aux données sera aussi gérée par ce langage.

5.6. Stockage et pérennité

Le logiciel sera stocké sur un git en accès libre afin que tout le monde puisse profiter du logiciel, voire même proposer des améliorations.

Le logiciel sera maintenu est amélioré au fur et à mesure des années par le BDI.

6. Conclusion

Nous espérons que ce DolMen puisse aider de nombreuses équipes à l'avenir, et que son action s'étende bien au delà du simple projet de CPO S6, ou même du tir de Juillet 2020. Sa mise en libre accès pourrait le diffuser, ou même lui permettre d'évoluer dans l'avenir. Le choix de langages répandus (C++ et Python) devrait faciliter la compréhension et la reprise du code par quelqu'un voulant faire évoluer notre logiciel.

Nous voulons développer une architecture suffisamment robuste pour que ce logiciel puisse servir de base à d'autres solutions techniques, tout en fournissant quelque chose d'aussi complet que possible.

7. Annexes

7.1. L’équipe

Timothée Allègre (t7allegr@enib.fr):

En tant que membre du BDI depuis quelques années, et suivant le projet FUSEX depuis ses débuts, j’ai souhaité m'investir dans ce projet autant que possible. J’ai rapidement tenté de formuler les besoins auxquels une telle interface pourrait répondre, et ai tenté de mettre ma connaissance du projet FUSEX au profit du projet DOLMEN. Je souhaite que ce projet puisse servir non seulement à notre équipe de cette année, mais aussi à d’autres équipes dans le futur, et qu’il puisse vivre quelques années supplémentaires.

Nathan De Saint Just (n6desain@enib.fr):

En tant que président du BDI depuis 2 ans et fondateur du Pôle KSP au sein de celui-ci avec Evan Roué, Il me tenait à coeur de participer à ce projet que je vois ce développé depuis sa création.
Avec grand espoir que tout soit opérationnelle pour la C’space 2020 avec le tire de la fusée avec le projet DOLMEN en base sol. Montrer qu’il marche nous permettra de le réutiliser et de la partager à tous les autres associations spatiales.

Axel Nougier (a7nougie@enib.fr):

En tant que membre du BDI depuis quelques années, et suivant le projet Enigma Robotics, j’ai souhaité mettre mes connaissances à profit dans le projet Dolmen afin de découvrir le Pôle KSP. Je souhaite que Dolmen soit un logiciel, simple, mais à la fois robuste et modulable afin qu’il puisse être appliqué à d’autres projets et pouvoir être modifié selon d’autres besoin et être amélioré par d’autres personnes.

7.2. Documents de référence

Description du projet de la Fusée Expérimentale, mené par le BDI depuis 2018: https://drive.google.com/file/d/1zeJgROn9wTeRf15NVJXXd0YqrwvZvNKZ/view?usp=sharing

Le diagramme de Gantt utilisé par l'équipe (et donc actualisé): https://docs.google.com/spreadsheets/d/16bBCIR2m5sRPZAXbCfMdWU7cPq7aEI2dDCHcjqEQBks/edit?usp=sharing

7.3. Glossaire

ENIB - Ecole National des Ingénieurs de Brest

BDI - Bureau Des Innovations (l'association menant le projet fusée).

KSP - Korrigan Space Program (le nom du pôle du BDI chargée de la fusée).

Fusex - Fusée Expérimentale.

7.4. Logiciels et ressources utilisées

Modelio 3.8 (Diagrammes UML, génération de code)

Googles docs

(rédaction du dossier de projet)

Google drive

(Stockage des documents)

Atom / Sublime text / VScodium / Visual Studio Code

Discord

(communication au sein de l'équipe)

Zotero

(Gestionnaire de bibliographie)

Python version 3

(Interface)

C++

(gestion de la trame, des données et des erreurs)

Gimp (Logo)

Canva (Page de garde et logo)

Github (Gestion du code du projet)

7.5. Détails de Trames

Capteur/Donnée	ID	Format data
GPS	01	20 octets
accéléromètre	02	17 octets
gyroscope	03	17 octets
température 2	04	7 octets
température 2	05	7 octets
pression 1	06	7 octets
pression 2	07	7 octets

7.5.1. Codage du GPS (ID 01)

Exemple 01048365375007409373 ; Latitude : 48°36'32.25" Nord Longitude : 7°40'56.238" Est

octets:	Description	Format
octet 0	ID	int
octet 1	ID	int
octet 2	Centaine Latitude	int
octet 3	Dizaine Latitude	int
octet 4	Unité Latitude	int
octet 5		int
octet 6		int
octet 7		int
octet 8		int
octet 9		int
octet 10		int
octet 11	Centaine Longitude	int
octet 12	Dizaine Longitude	int
octet 13	Unité Longitude	int
octet 14		int
octet 15		int
octet 16		int
octet 17		int
octet 18		int
octet 19		int
octet 20	;	char

7.5.2. *Codage de l'accéléromètre (ID 02)*

Exemple 02+1023-0213+0125 => accéléromètre x: +10,23m/s^2 , y: -02,13m/s^2 et z: +01,25m/s^2

octets:	Description	Format
octet 0	ID	int
octet 1	ID	int
octet 2	Signe accélération suivant x	char
octet 3	Dizaine accélération suivant x	int
octet 4	Unité accélération suivant x	int
octet 5	Dixième accélération suivant x	int
octet 6	Centième accélération suivant x	int
octet 7	Signe accélération suivant y	char
octet 8	Dizaine accélération suivant y	int
octet 9	Unité accélération suivant y	int
octet 10	Dixième accélération suivant y	int
octet 11	Centième accélération suivant y	int
octet 12	Signe accélération suivant z	char
octet 13	Dizaine accélération suivant z	int
octet 14	Unité accélération suivant z	int
octet 15	Dixième accélération suivant z	int
octet 16	Centième accélération suivant z	int
octet 17	;	char

7.5.3. *Codage du gyroscope (ID 03)*

Exemple 031023-0213+0125 => gyroscope x: +10,23rad/s , y: -02,13m/s et z: +01,25m/s

octets:	Description	Format
octet 0	ID	int
octet 1	ID	int
octet 2	Signe gyroscope suivant x	char
octet 3	Dizaine gyroscope suivant x	int
octet 4	Unité gyroscope suivant x	int
octet 5	Dixième gyroscope suivant x	int
octet 6	Centième gyroscope suivant x	int
octet 7	Signe gyroscope suivant y	char
octet 8	Dizaine gyroscope suivant y	int
octet 9	Unité gyroscope suivant y	int
octet 10	Dixième gyroscope suivant y	int
octet 11	Centième gyroscope suivant y	int
octet 12	Signe gyroscope suivant z	char
octet 13	Dizaine gyroscope suivant z	int
octet 14	Unité gyroscope suivant z	int
octet 15	Dixième gyroscope suivant z	int
octet 16	Centième gyroscope suivant z	int
octet 17	;	char

7.5.4. *Codage température (ID 04 et ID5)*

Exemple : 0X+3015 ; => température (X = 4 ou 5) température = +30,15°c

octet 0	ID
octet 1	ID
octet 2	Signe température
octet 3	Dizaine température
octet 4	Unité température
octet 5	Dixième température
octet 6	Centième température
octet 7	;

7.5.5. *Codage pression (ID 06 et ID 07)*

Exemple : 0X+3015 ; => température (X = 7 ou 8) température = +30,15Pa

octets:	Description	Format
octet 0	ID	int
octet 1	ID	int
octet 2	Signe pression	char
octet 3	Dizaine pression	int
octet 4	Unité pression	int
octet 5	Dixième pression	int
octet 6	Centième pression	int
octet 7	;	char

8. Bibliographie

« A Hackable Text Editor for the 21st Century ». *Atom*. https://atom.io/ (25 mars 2020).

« Build Software Better, Together ». *GitHub*. https://github.com (29 mars 2020).

« Collaborer et créer d'incroyables designs gratuitement ». *Canva*. https://www.canva.com (29 mars 2020).

« cppreference.com ». https://fr.cppreference.com/w/ (25 mars 2020).

« Discord — Discord - Chat Vocal et Textuel Gratuit ». *Discord*. https://discordapp.com/ (25 mars 2020).

« GIMP ». *GIMP*. https://www.gimp.org/ (29 mars 2020).

« Google Docs ». https://docs.google.com/document/u/0/ (25 mars 2020).

« Google Drive ». https://drive.google.com/ (25 mars 2020).

« Modelio Open Source - UML and BPMN Free Modeling Tool ». *Modelio Open Source*. https://www.modelio.org/ (25 mars 2020).

« Sublime Text - A sophisticated text editor for code, markup and prose ». https://www.sublimetext.com/ (25 mars 2020).

« Visual Studio Code - Code Editing. Redefined ». https://code.visualstudio.com/ (25 mars 2020).

« VSCodium - Open Source Binaries of VSCode ». https://vscodium.com/ (25 mars 2020).

« Welcome to Python.Org ». *Python.org*. https://www.python.org/ (25 mars 2020).

« Zotero | Your personal research assistant ». https://www.zotero.org/ (25 mars 2020).