Création d'un logiciel pour l'utilisation du Laser STIMUL 1430

CHAPELLE Quentin

Centre d'Investigation Clinique (C.I.C.) de Clermont-Ferrand

J'autorise la diffusion de mon rapport sur l'intranet de l'IUT

Remerciements

Je tiens tout d'abord à remercier M. Le Professeur Claude Dubray, chef de service du Centre d'Investigation Clinique, de m'avoir offert la possibilité de travailler sur ce projet.

J'adresse ensuite mes remerciements à Mme Alice MARTIN, ma responsable de stage, pour son soutien et sa disponibilité tout au long de ce stage.

Je remercie aussi mes professeurs d'IUT qui m'ont prodigué ces deux dernières années les connaissances et les conseils adaptés pour devenir un bon informaticien et qui m'ont permis de me sortir de bien des ennuis durant ce stage.

Enfin je tiens à remercier également M. Grégoire Astruc, stagiaire de l'ISIMA au Centre d'Investigation Clinique, qui m'a transmis son savoir sur quelques techniques informatiques.

Sommaire

Introduction

C'est en tant qu'étudiant et candidat au diplôme Universitaire de Technologie (D.U.T.) de Clermont-Ferrand en informatique, que j'ai accompli ce stage au Centre d'Investigation Clinique (CIC) de Clermont-Ferrand. Il a été conduit sous la tutelle de Mme Alice Martin, pour une durée de dix semaines.

Le principal objectif de ce stage était de développer un utilitaire permettant de contrôler et d'automatiser le laser Nd:YAP Stimul 1340, matériel destiné à produire des stimulations douloureuses chez des volontaires pour la recherche clinique. Cet outil devait être simple et convivial pour permettre à des non-informaticiens de l'utiliser aisément.

J'ai abordé le projet par étapes successives. Dans un premier temps, je suis passé par une phase d'apprentissage pour travailler en toute sécurité avec l'appareil. En effet, l'utilisation du laser pouvant être dangereuse, seul un personnel qualifié et formé est autorisé à manipuler ce dispositif. Dans le même temps, j'ai produit des maquettes et les ai présentées aux futurs utilisateurs, ce qui m'a permis de capter au mieux leurs besoins.

La deuxième étape de ce projet a été consacrée à la réalisation du logiciel. Deux défis majeurs se sont alors présentés. Le premier a été de maitriser parfaitement la communication avec le laser à l'aide d'une interface "Manuelle". Le second a consisté à automatiser des procédures utilisées dans les protocoles de recherche.

Enfin, la dernière étape a concerné le déploiement de cet utilitaire dans d'autres équipes de recherche, ainsi que la rédaction d'un manuel d'utilisation.

1. Présentation du service

Durant ces dix semaines, j'ai évolué au Centre d'investigation Clinique (C.I.C.) de Clermont-Ferrand. Ce centre est directement rattaché au Centre Hospitalier Universitaire (C.H.U) Gabriel Montpied.

En 2008 le C.H.U ne comptait pas moins de 1031 étudiants et internes en formation médicale et 918 étudiants en formation des professionnels de santé et il a mis en place plus de 130 nouvelles études cliniques.

Figure1 : Logo C.H.U

Le C.I.C. est l'un des centres chargés de mettre en place ces études (entre vingt et trente par ans). Il a été créé en 2005, et est composé de plusieurs structures telles que le Centre de Pharmacologie Clinique (C.P.C.) et l'Unité de Recherche Clinique en Cancérologie du Centre de Lutte Contre le Cancer Jean Perrin. Le C.P.C. est dédié depuis 1996 à la recherche en pharmacologie clinique et est agréé par le ministère de la santé pour la prise en charge d'essais cliniques incluant les premières administrations de nouvelles molécules chez l'homme.

Cet établissement a donc pour but de mettre en place des tests cliniques sur l'homme dans les domaines suivants :

- La pharmacologie clinique
- La Douleur
- Les Neurosciences
- La génétique et le Développement
- L'Hépato-gastroentérologie
- La nutrition
- L'oncologie
- La Thérapie cellulaire
- La vaccinologie

Figure 2 : Logo C.I.C.

Ce centre de 750m² est spécifiquement aménagé pour la réalisation d'études cliniques. Il est équipé de 18 lits médicalisés. Chaque lit possède un moniteur individuel de paramètres vitaux. Tous les moniteurs sont reliés à une station de surveillance centralisée.

Le CPC dispose de 5 postes d'explorations fonctionnelles ainsi que d'une salle de tests psychométriques et d'un laboratoire pour le traitement des échantillons biologiques.

En pharmacologie clinique, le développement d'un nouveau médicament ou procédé thérapeutique doit impérativement suivre 4 phases distinctes :

- **Phase 1 :** Test de tolérance pour un petit nombre de volontaire sain. C'est la première évaluation du principe actif chez l'homme, elle est donc très surveillée. C'est durant cette phase que sont étudiés les effets pharmacocinétiques (réactions se produisant dans l'organisme après l'introduction du médicament) et pharmacodynamiques (actions du médicament sur l'organisme) du produit à l'étude.
- **Phase 2 :** Définition des doses sur un petit nombre de volontaire sain. L'efficacité du produit est évaluée, souvent par comparaison des résultats obtenus avec le produit à l'étude et un produit placebo.
- **Phase 3 :** Augmentation du nombre de volontaires et tests d'efficacité et de tolérance par rapport au traitement de référence. Evaluation des effets indésirables et des interactions médicamenteuses.
- **Phase 4 :** Phase de pharmacovigilance qui intervient après l'Autorisation de Mise sur le Marché, délivrée par l'AFSSAPS (Agence Française de Sécurité Sanitaire des Produits de Santé), afin de valider la tolérance du produit sur le long terme et sur une population plus vaste.

Le C.P.C. peut être sollicité pour les études de phase 1 à 4, mais son activité principale concerne les phases 1, 2 et 3.

Lorsqu'une étude clinique est confiée au C.P.C., la démarche adoptée est toujours la même :

1. **Rédaction des protocoles**
2. **Démarches administratives :** demande d'autorisation à l'AFSSAPS, au CPP (Comités de Protection des Personnes) et à la CNIL (Commission Nationale de l'Informatique et des Libertés). Cette étape est assez longue, elle peut durer plus de deux mois.
3. **Rédaction des documents d'étude:** Questionnaires, contrats, ...
4. **Commande des traitements**
5. **Recrutement des patients ou volontaires sains**
6. **Réalisation du protocole (4 phases)**
7. **Data management :** gestion des données du protocole
8. **Tests Statistiques**
9. **Rédaction du rapport final**

Pour mener à bien les études qui lui sont confiées, le CPC emploie différents corps de métiers :

- Trois ETP (Equivalent Temps Plein) Infirmiers - A.R.C (Attachés de Recherche Clinque)
- Un ETP Technicien de laboratoire
- Trois ETP médecins de recherche
- Deux ETP A.R.C
- Un ETP Ingénieur Biomédical
- Un et demi ETP Administratif

Figure 3 : Vue d'extérieur du C.I.C.

Le domaine dans lequel mon travail s'est inscrit est celui de la douleur. Ce service utilise différents dispositifs de stimulations douloureuses (électrique, thermique, mécanique, ...) sur des patients ou des volontaires sains.

Ces études sur la douleur servent notamment à comprendre l'origine et le fonctionnement de celle-ci pour, à terme, réussir à mieux soigner certaines pathologies (comme les douleurs neuropathiques).

2. Présentation synthétique du stage

Le laser Stimul 1340 est un produit de la société ElEn, plus précisément de leur division médicale Deka. Il est utilisé pour effectuer des stimulations douloureuses sur des patients et ainsi observer leur réaction et celle de leur organisme.

Pour chaque impulsion le laser à besoin d'un certain nombre de paramètres tels que le « time pulse » en ms, l'énergie en joule et le diamètre du faisceau laser en mm. Ces paramètres combinés permettent d'envoyer une stimulation plus ou moins douloureuse.

Les protocoles de recherche qui utilisent le laser comme source de douleur suivent tous le même principe :

- Le premier test chez chaque volontaire est de déterminer son *seuil de douleur*, c'est-à-dire la stimulation pour laquelle la sensation ressentie est clairement douloureuse.
- A partir de ce seuil de douleur, il existe deux types de test : Le *potentiel évoqué* et le *Wind-up*. Tous deux fonctionnent sur le même principe, qui consiste à répéter une même stimulation douloureuse un certain nombre de fois, à une certaine fréquence. La différence entre les deux tests réside dans la fréquence utilisée, plus élevée pour le *Wind-up* que pour le *potentiel évoqué*.

Ces trois techniques seront détaillées dans la partie « **4.1.2 Notions médicales importantes** ».

Actuellement pour utiliser le laser et pour mener à bien chaque étude, les médecins doivent suivre des procédures manuelles peu précises, contraignantes, et peu reproductibles. En effet, entre chaque opération, les paramètres doivent être modifiés manuellement, et les temps entre deux impulsions sont respectés à l'aide d'un chronomètre.

Ainsi, pour gagner en temps et en précision, et donc obtenir une meilleure qualité des opérations, ma mission était d'informatiser ces tâches, et de créer des modules pour chacun des tests décrits ci-dessus.

3. Outils

Avant de commencer le développement, il m'a fallu choisir les technologies et les outils que j'allais utiliser.

3.1. C++ & Qt

Le (C++) est un langage de programmation développé par Bjarne Stroustrup au cours des années 1980. Il inclut des fonctionnalités comme, entre autres, la prise en charge des classes, les fonctions virtuelles, la surcharge, ou encore la gestion d'exceptions.

Le C++ est un langage de plus bas niveau que le Java et le C#. C'est ce qui fait une de ses forces mais aussi une de ses faiblesses.

En effet, la gestion de la mémoire étant beaucoup moins automatisée que chez ses rivaux, il est possible pour un bon développeur de mieux optimiser ses applications. De plus le C++ n'utilise pas de machine virtuelle pour fonctionner.

D'un autre coté, il est plus long et difficile de développer un programme à l'aide du C++ et sa portabilité est limitée car chaque application doit être compilée sur toutes les plateformes ou il sera utilisé.

Figure 4: C++ illustration

L'utilitaire à développer lors de se stage devait avoir une interface graphique simple, agréable et conviviale. J'ai donc eu besoin d'outils pour créer celle-ci.

Mon choix c'est porté sur Qt qui est un Framework orienté objet développé en C++. La première version de Qt a vu le jour en 1995, elle appartient alors à la société Troll Tech. En janvier 2008 Nokia fait l'acquisition de cette société dans le cadre de son <<Développement dans les applications multiplateformes>>. Ce Framework ajoute ainsi deux notions très importantes au C++ qui sont les Signaux et les Slot. Grâce à ces deux fondamentaux, la programmation événementielle, qui est indispensable à la création d'interfaces graphiques, devient possible. Qt fournit aussi de nombreux outils permettant par exemple la communication réseau, l'exploitation et la gestion de fichier XML.

Figure 5: Logo Qt

3.2. X.M.L

X.M.L signifie Extensible Markup Language ou en français « langage extensible de balisage ». Ce langage utilise des balises que l'utilisateur peut créer à sa guise. il sert essentiellement à stocker ou transférer des données textes structurées en champs arborescents. De nombreux outils respectent la syntaxe XML comme : le XHTML pour les pages web, le RSS, le SVG, etc.

Dans notre cas, cet outil sera utilisé pour concevoir et exploiter les fichiers de sauvegarde et de configuration.

Figure 6: X.M.L. illustration

3.3. UML & bouml

U.M.L. ou Unified Modeling Language (en français « langage de modélisation unifié ») est un langage de modélisation graphique permettant de faciliter la conception orientée objet. Il propose plusieurs types de diagrammes ayant pour but de modéliser tous les aspects d'un programme informatique, aussi bien d'un point de vue dynamique que statique. Ce langage n'étant pas une méthode, l'utilisation de chaque diagramme est laissée à l'appréciation de chacun. Il y a néanmoins des conventions établies permettant de produire des analyses assez complètes. U.M.L. a été soumis à l'OMG (Object Management Group[1]) en 1997 et est né du rassemblement et de l'unification des principales méthodes d'analyse et de modélisation de l'époque.

Figure 7: Logo U.M.L.

Figure 8: Logo bouml

Pour faire l'analyse et créer rapidement des diagrammes propres, j'ai utilisé le modeleur UML bouml qui est open source. Les diagrammes ainsi réalisés m'ont servi à cadrer au mieux les attentes des utilisateurs et le développement de mon application.

Il faut savoir que bouml est capable de générer du code C++, java, PHP, ou autre, juste à l'aide d'une analyse complète et bien réalisé. Je n'ai pas utilisé cette fonctionnalité.

[1] Object Management Group : "association américaine à but non-lucratif créée en 1989 dont l'objectif et de standardiser et promouvoir le modèle objet sous toutes ses formes" (Wikipedia)

3.4. balsamiq

Pour qu'une analyse soit la plus complète possible et que l'utilisateur comprenne au mieux, il faut concevoir et présenter des maquettes d'écran. Pour réaliser ceci j'ai utilisé un petit outil en ligne nommé « balsamiq », qui permet de faire des croquis efficaces en très peu de temps.

Figure 9: balsamiq illustration

3.5. Qt creator

Qt creator est un IDE (Integrate Developpement Environnement) C++ dédié principalement à l'utilisation de la bibliothèque Qt. Il embarque tous les outils nécessaires, comme un débogueur, un compilateur, ou un assistant graphique de création d'interface. Actuellement en version 1.3.1, il permet aussi la création et l'intégration d'applications pour des plateformes tierces comme les téléphones mobiles.

Figure 10: écrans d'accueil de Qt creator

3.6. Serial Port Monitor

Cet outil permet d'intercepter, afficher et analyser les échanges de données entre une application et un périphérique série. Son aide a été plus que précieuse car grâce à celui-ci la communication avec le laser est devenue transparente.

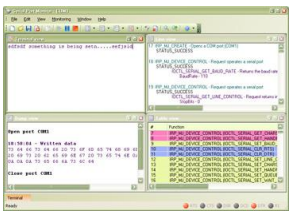


Figure 11: illustration serial port monitor

3.7. BitRock Install Builder

C'est un logiciel propriétaire, normalement payant, permettant de créer des installateurs personnalisés multiplateformes. Développé par BitRock, il constitue pour les projets sous licence libre une alternative aux produits commerciaux comme InstallShield. Il embarque tout le nécessaire pour créer une interface d'installation graphique et pour simplifier au maximum le déploiement d'un programme. La création et la personnalisation des installateurs se fait par le biais d'une interface graphique simplifiée et de scripts qui sont ensuite compilés pour donner un exécutable.

Figure 12: Logo BitRock

4. Etudes préliminaires

Pour être le plus productif possible et pour éviter d'éventuelles futures erreurs, une réflexion poussée sur le travail à effectuer est fondamentale. C'est pourquoi mon travail à commencé par une phase d'analyse et d'adaptation à l'environnement et aux utilisateurs.

4.1. Adaptation au service

4.1.1. Rencontres avec l'équipe

Connaitre et cerner un groupe de personnes qui vont travailler avec votre application est selon moi très important. En effet, cela permet de gagner beaucoup de temps au niveau de l'analyse et sur la création des maquettes. Pour effectuer ce travail, des rencontres régulières avec les utilisateurs ont été nécessaires. Cela a ainsi permis de vérifier que la logique et l'ergonomie de l'application correspondait au mieux à celles des utilisateurs.

4.1.2. Notions médicales importantes

Mon cursus n'ayant aucun contenu médical et mes connaissances personnelles dans ce domaine étant trop limitées, ma première tâche au C.I.C. fût un travail de bibliographie, d'apprentissage et de formation, où, j'ai dû m'approprier les tenants et les aboutissants des études pratiquées avec le Laser STIMUL 1340 ainsi que l'utilisation de ce dernier. En effet plusieurs contraintes techniques et médicales sont à prendre en compte pour que mon application soit utilisable et sans danger.

Ce laser envoie des stimulations douloureuses sur un patient, à la suite desquelles des analyses de son ressenti et de son comportement neurophysiologique sont effectuées. Pour mener ces expérimentations il y a trois procédures de bases.

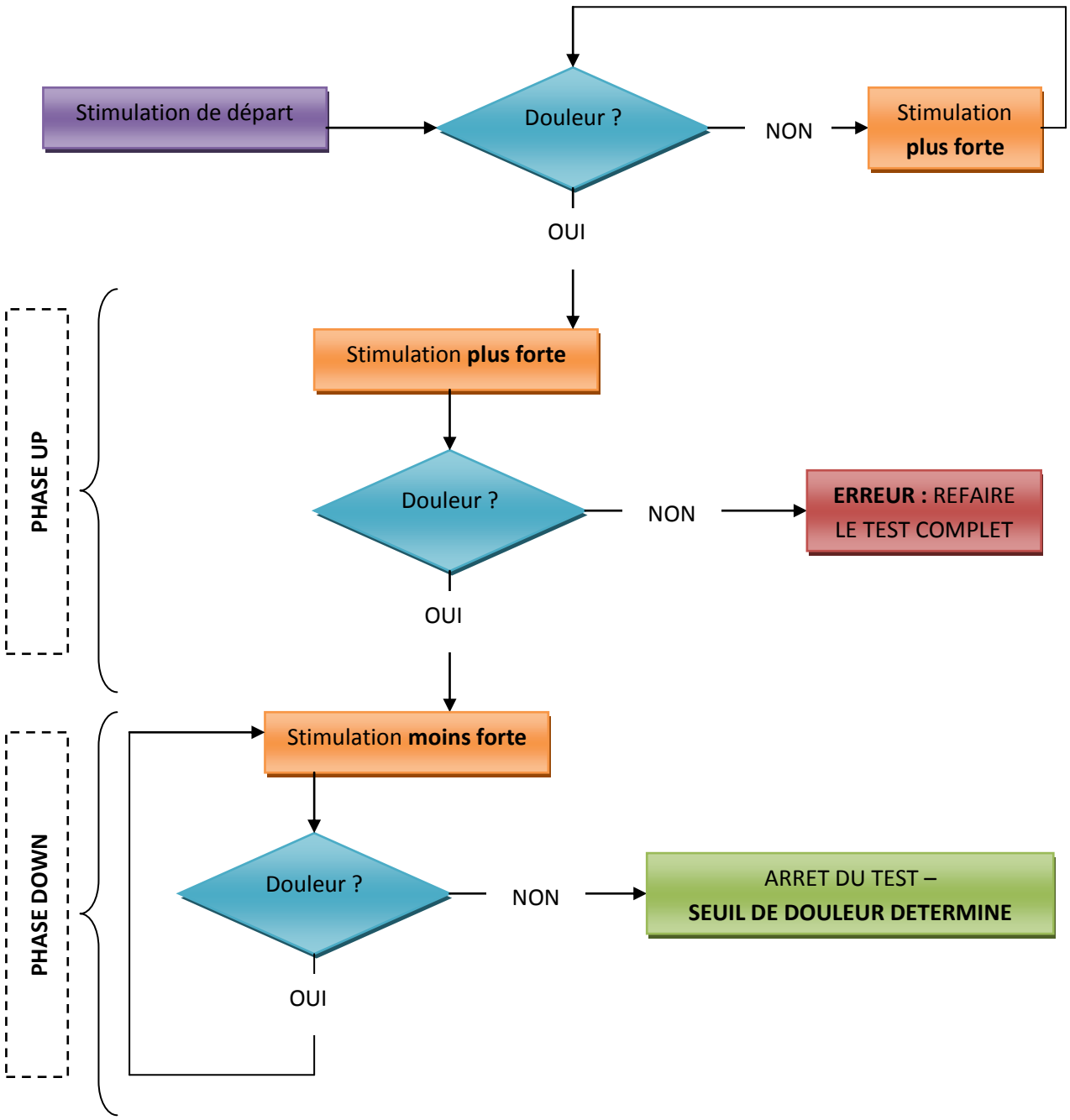
a) La détermination du seuil de douleur :

Le seuil de douleur correspond à la stimulation pour laquelle le patient ressent une douleur clairement définie. C'est la valeur de base qui va nous servir pour toutes les autres opérations. Pour obtenir ce seuil, on utilise la technique du « **Up and Down »**.

Cette technique consiste dans un premier temps à envoyer des stimulations laser de plus en plus fortes au patient, en suivant une progression linéaire prédéfinie (valeur de départ et pas entre deux stimulations). Ceci nous permet de connaître la valeur de la stimulation correspondant à la première sensation douloureuse.

Dans un second temps, cette valeur de stimulation est ajustée grâce à une phase de confirmation de seuil par le haut (phase up) et par le bas (phase down).

Pour plus de clarté, on peut schématiser la technique de la façon suivante :



Le calcul du seuil de douleur se fait ensuite de la façon suivante :

S1 = valeur de la stimulation pour la première réponse 'OUI' du test

S2 = valeur de la stimulation pour la dernière réponse 'OUI' du test

SEUIL = $\frac{(S1+S2)}{2}$

b) *Le potentiel évoqué*

Cette technique consiste à envoyer plusieurs impulsions identiques à intervalle régulier en laissant entre deux stimulations le temps à la zone stimulée de se désensibiliser.

A chaque stimulation, la réponse électro-physiologique du sujet est enregistrée au moyen d'un EEG (Electro-encéphalogrammes). Chaque stimulation étant identique, on peut obtenir une réponse neurophysiologique moyennée sur toutes les stimulations.

Ceci permet de calculer les vitesses de conduction des messages nerveux émis par les fibres nerveuses de la douleur.

c) *Le Wind-up*

L'opération est identique à celle du potentiel évoqué à la différence près que le temps entre deux stimulations est plus court. Ceci a pour effet de stimuler une zone encore sensibilisée par la stimulation précédente, afin d'observer une sommation des réponses neurophysiologiques.

4.1.3. Présentation et utilisation du laser

d) *Présentation*

D'après (Wikipedia/laser) un Laser est un appareil émettant de la lumière amplifiée. Il en existe 7 types différents:

1. À colorants
2. À gaz
3. Les Diodes laser
4. À électrons libres
5. À fibre
6. Téramobile
7. Cristallins.

Le laser STIMUL 1340 est un laser Cristallin ayant pour référence Nd:YAP. Cela signifie que son cristal est composé d'Yttrium Aluminium Pérovskite et dopé avec l'ion néodyme.

Pour mieux comprendre la constitution d'un laser cristallin, la figure ci-dessous montre comment est composé un laser rubis.

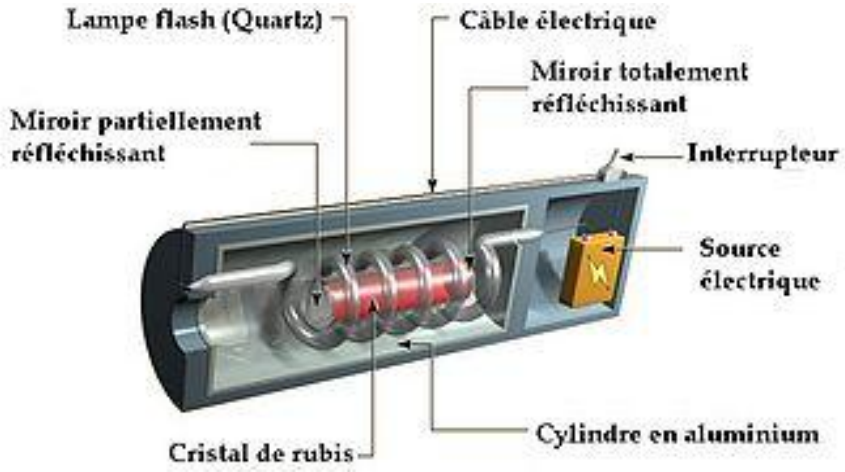


Figure 13: Constitution d'un laser

La composition générale du laser Stimul 1340 est à peu près similaire, seul le cristal est différent.

Les Laser sont classés par niveau de dangerosité. Il existe ainsi sept catégories:

- **Classe 1** : sans dangers utilisé dans des conditions normales
- **Classe 1M :** peut être dangereux en vision directe
- **Classe 2 :** Laser visible ou la protection de l'œil est normalement assurée par des réflexes de défenses.
- **Classe 2M:** Laser visible qui peut être dangereux en vision directe.
- **Classe 3R:** Laser dangereux pour l'œil et n'étant pas forcément visible
- **Classe 3B:** Laser toujours dangereux en vision directe mais normalement sans risque par réflexions diffuses
- **Classe 4:** Catégorie la plus dangereuse. Ces lasers peuvent provoquer des dommages sur la peau, des incendies, sont extrêmement dangereux pour l'œil même sans être en regard direct.

Le Laser Stimul 1340 fait partie de la classe 4, il faut donc veiller à ce que toutes les recommandations de sécurité soient respectées dans l'utilisation automatisée.

En utilisation manuelle, le système dispose de plusieurs niveaux de sécurité :

- A l'allumage du laser, un message d'avertissement indique que le port de lunettes de protection est obligatoire pour toutes les personnes présentes dans la pièce.
- Pour envoyer une stimulation, il faut impérativement que l'utilisateur enclenche une pédale. Cette sécurité permet d'éviter qu'une stimulation soit envoyée malencontreusement. Cette sécurité ne peut pas être évitée, ni informatisée, l'opération devra donc rester manuelle.
- A tout moment, l'utilisateur peut enclencher un bouton d'arrêt d'urgence en cas de problème, qui éteint complètement la source laser.

Le programme devra reprendre tous les messages d'avertissement nécessaires au fonctionnement sécurisé du laser.

Cela signifie aussi qu'il faudra réaliser des simulations de mauvaise utilisation du programme pour vérifier que les sécurités du logiciel sont bien pensées et efficaces.

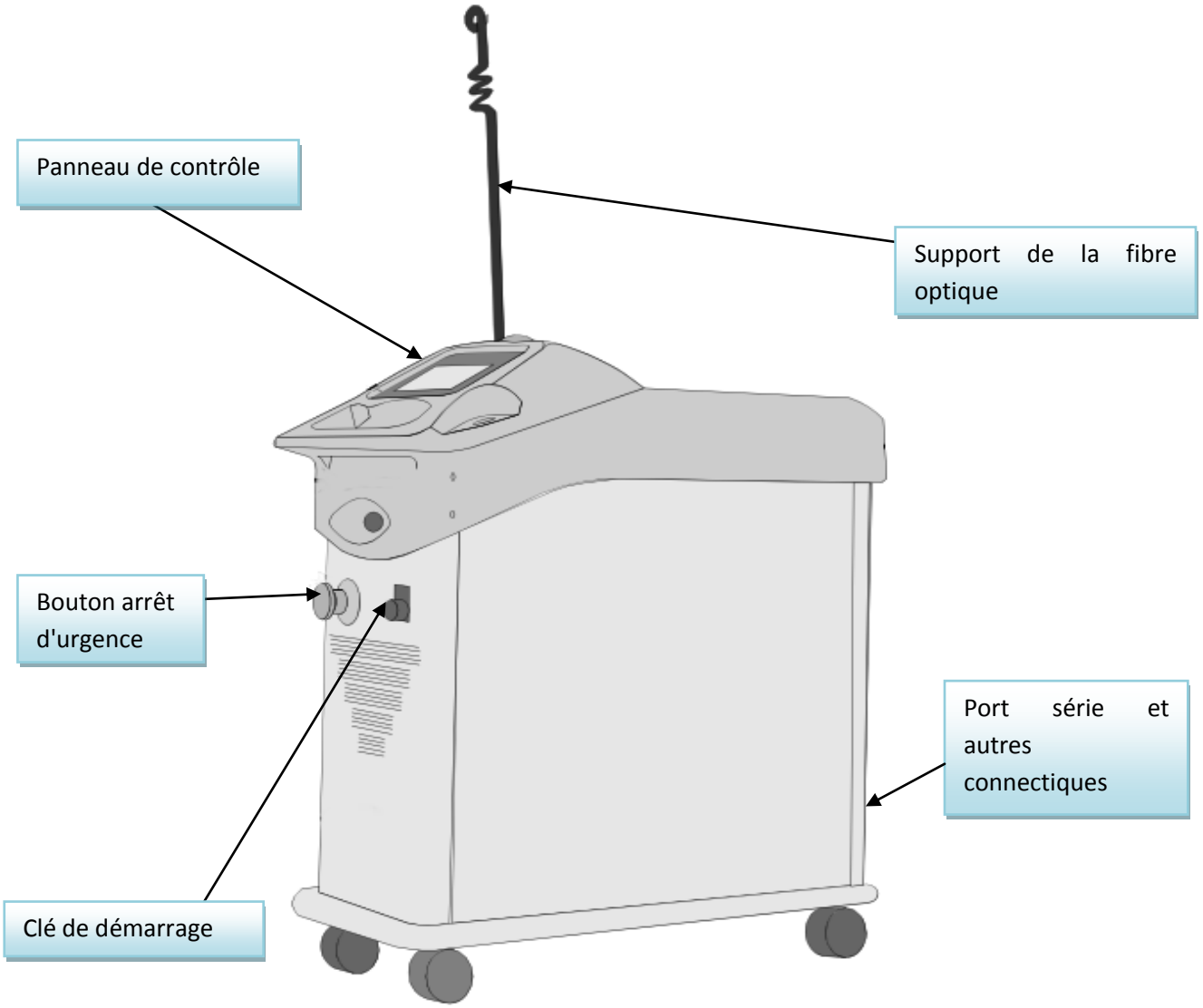


Figure 14: Laser Stimul 1340

e) *Utilisation du Laser*

Lorsque l'on utilise le Laser sans logiciel, plusieurs étapes sont nécessaires afin de pouvoir envoyer une stimulation :

- Tourner la clé d'allumage
- Valider sur le panneau de contrôle le port des lunettes
- Appuyer sur la touche LaserOn pour mettre en charge la source laser
- Régler les paramètres (pulse, spot, energy)
- Régler les options (son, pointeur laser)
- Appuyer sur la touche Operate pour lancer la calibration

Figure 15 : Ecran confirmation du port des lunettes

Figure 16 : Ecran de contrôle du laser

Ensuite la stimulation se fait à l'aide d'une télécommande externe au laser, en appuyant en même temps sur la pédale.

Le laser dispose aussi d'un bouton d'arrêt d'urgence qui éteint complètement le laser (en façade), et d'un bouton standby permettant d'éteindre la source laser (sur le panneau de contrôle).

4.2. Prise en compte des Besoins (cahier des charges)

Maintenant que toutes les données indispensables pour comprendre le problème ont été acquises, la prise en compte des besoins et l'établissement d'un cahier des charges est possible. Après les réunions et les discussions avec les utilisateurs trois mots sont ressortis :

Simplicité, efficacité et convivialité.

Tout en tenant compte de ces trois notions, le logiciel devra embarquer les éléments suivants:

- **Un mode manuel :** reprenant les fonctionnalités du panneau frontal du laser
- **Un mode d'administration des séquences :** permettant de créer, modifier et supprimer des séquences.
- **Un mode automatique :** permettant de charger et de lancer des séquences préalablement créées.
- **Sauvegarde et chargement de séquences :** permettant de sauvegarder des séquences de travail d'une fois sur l'autre.
- **Présence d'un fichier de configuration :** permettant d'adapter le logiciel à d'éventuels changements de protocoles de communication série et permettant le réglage de certains timeout.
- **Aide intégrée :** permettant de guider un utilisateur novice.
- **Présence d'un installateur :** pour un déploiement plus aisé.
- **Implémentation de certains protocoles :** Eux même utilisés pour créer des séquences.
- **Garder une certaine flexibilité.**

Voici donc tous les points directeurs de ce projet. A présent il est temps d'entrer un peu plus dans les détails et de commencer une analyse.

4.3.Première Analyse

Après une première réunion qui m'a permis d'établir la liste des besoins, j'ai pu pousser ma réflexion et l'exprimer de manière plus pragmatique.

4.3.1. Démarche adoptée

Avant tout développement il est indispensable de définir les buts, les fonctions et les solutions à adopter. Pour cela, une analyse UML est indispensable. Voici donc le schéma que j'ai adopté pour celle-ci:

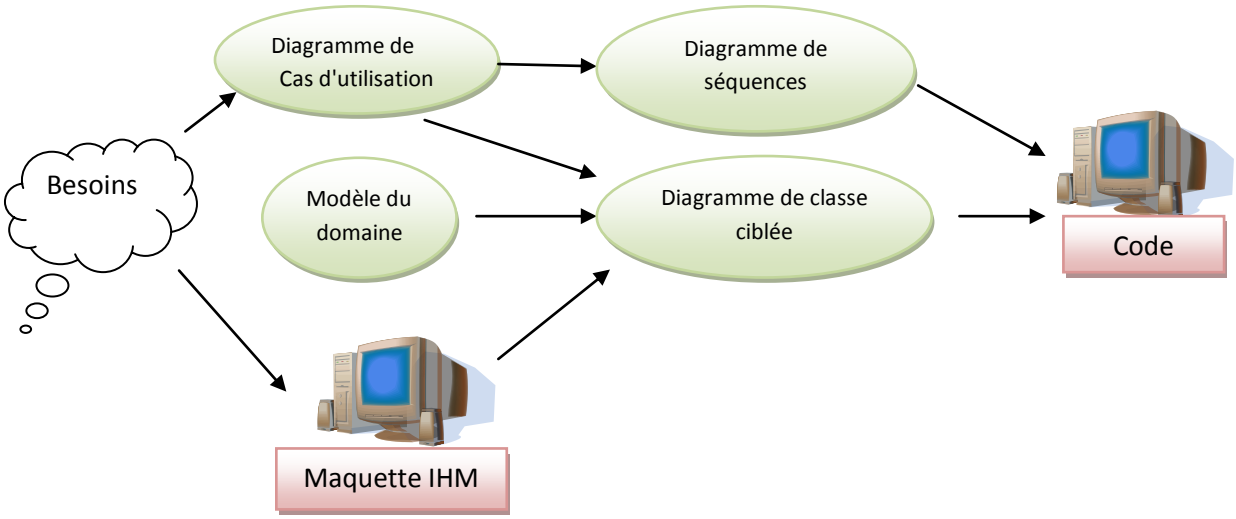


Figure 17: Schéma chronologique d'analyse

Chaque élément de ce schéma à un but précis:

- **Besoins :** Toute production informatique part de besoins à affranchir. Ceci va nous servir pour le diagramme de cas d'utilisation.

- **Diagramme de cas d'utilisation :** Reprend tout les cas que l'on peut rencontrer. Ce diagramme permet d'analyser tout ce que l'utilisateur peut faire en fonction de son statut.

- **Diagramme de séquence :** Permet de représenter les échanges entre des acteurs, des machines ou des objets. Ce type de diagramme est plus dynamique et permet de représenter assez aisément des algorithmes.

- **Modèle du domaine :** La programmation objet utilise des objets. Ceux-ci sont des instances de classe. Ce type de diagramme permet donc de les représenter et donne ainsi un aperçu global de la structure d'objet utilisé.

- **Diagramme de classe ciblée** : Ce diagramme est du même type que le modèle du domaine mais cible et détaille un point particulier.

- **Maquette IHM :** Maquette de l'Interface Homme Machine. Ce sont des schémas ou des compositions donnant un aperçu de l'interface graphique à venir.

Ce schéma théorique est seulement un guide. En effet les remarques et l'évolution de la réflexion des utilisateurs imposent une certaine flexibilité pour toujours rester au plus proche de leurs attentes. De plus, même si une réflexion sur chacun des points précédents est importante, tous les diagrammes ne sont pas nécessaires.

4.3.2. Mode connecté – déconnecté

Très vite la notion de travail connecté est apparue. En effet la connexion avec le laser est nécessaire pour l'utilisation à proprement parler du laser, mais celle-ci peut être contraignante lorsque l'utilisateur veut seulement modifier où créer des séquences. C'est pourquoi il a été nécessaire de développer les deux utilisations suivantes :

1. L'utilisation déconnectée : qui permet la création, la modification et la suppression de séquences ainsi que l'accès à la documentation.
2. L'utilisation connectée : qui reprend les mêmes fonctionnalités en ajoutant l'envoi de stimulation manuel ou automatisé.

4.3.3. Cas d'utilisation

En tenant compte de ces besoins et de cette notion de mode connecté la première étape de mon analyse à consisté à répertorier les besoins et à les représenter sous forme d'un diagramme de cas d'utilisation.

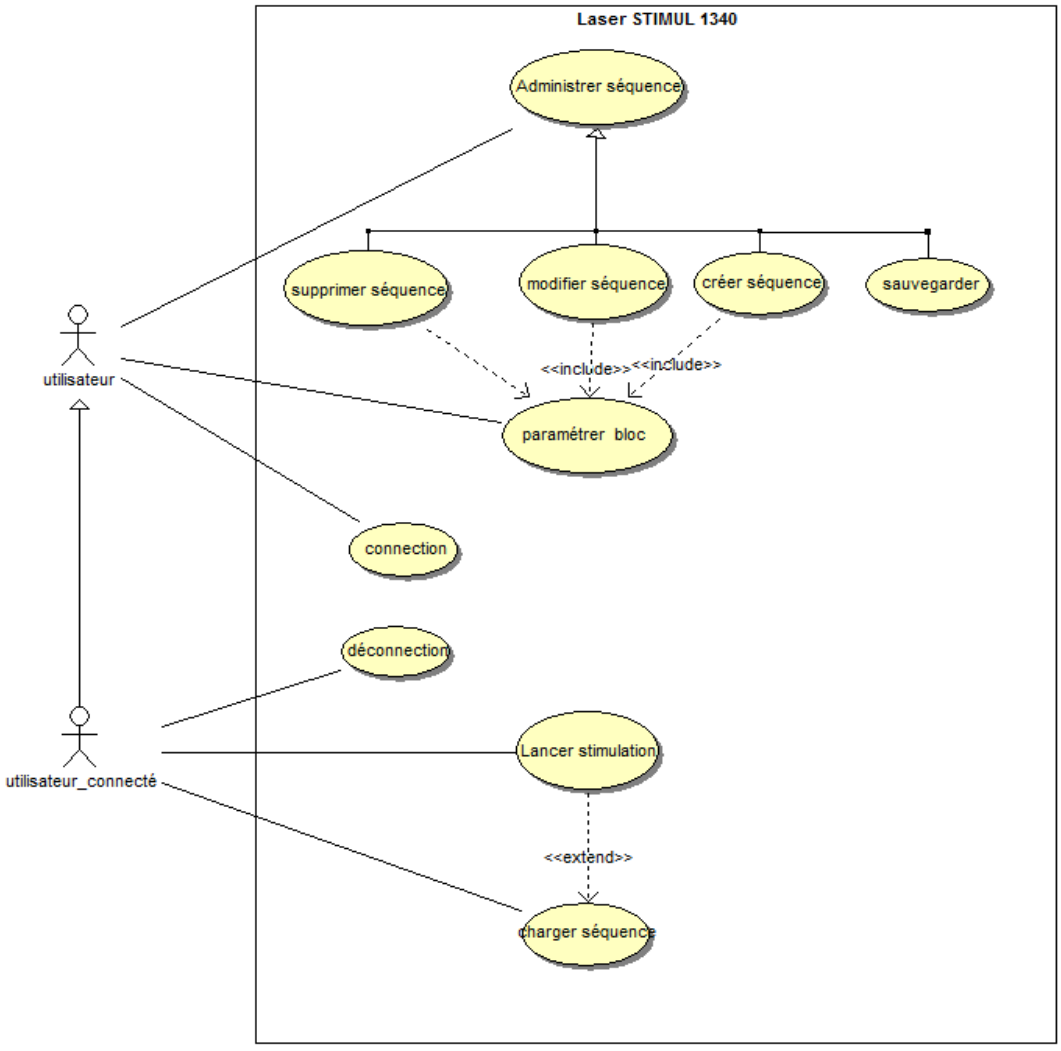


Figure 18: Diagramme de cas d'utilisation

4.3.4. Maquette

Une fois mes cas d'usage répertoriés, j'ai, grâce à l'outil balsamiq, crée des maquettes de chaque écran, ce qui m'a permis d'être plus clair auprès des utilisateurs.

Ci-dessous voici la maquette des écrans d'accueil, en utilisation connectée et non connectée.

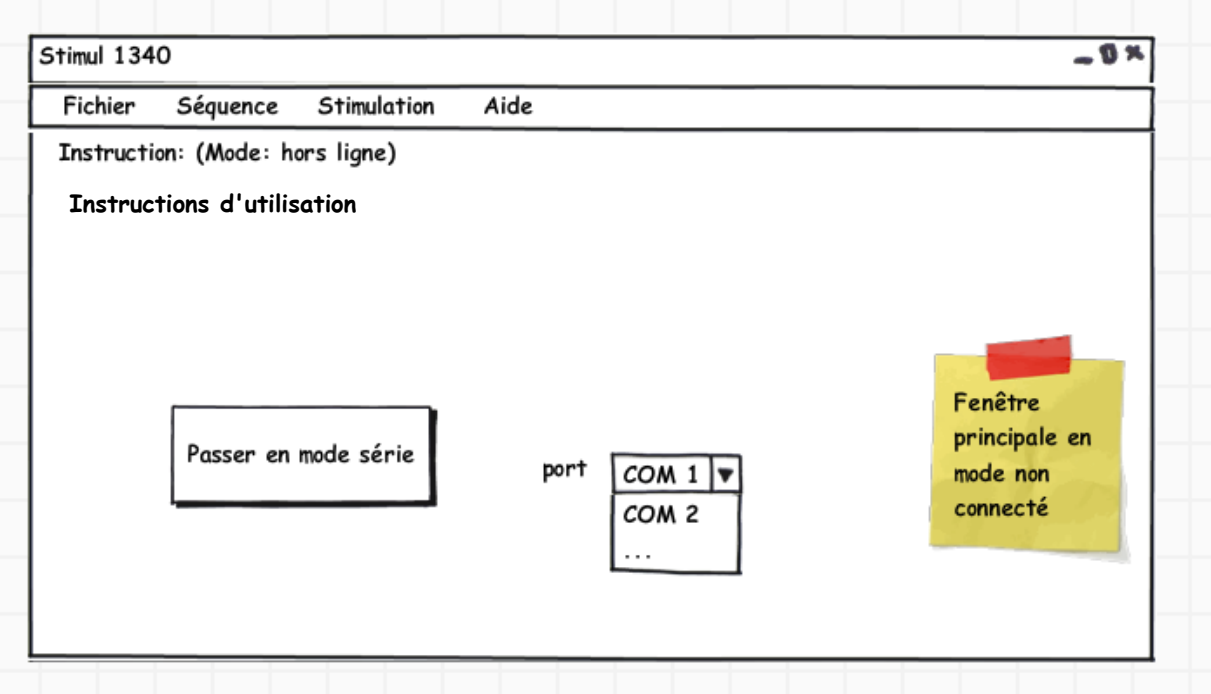


Figure 19: Maquette écran accueil utilisation déconnectée

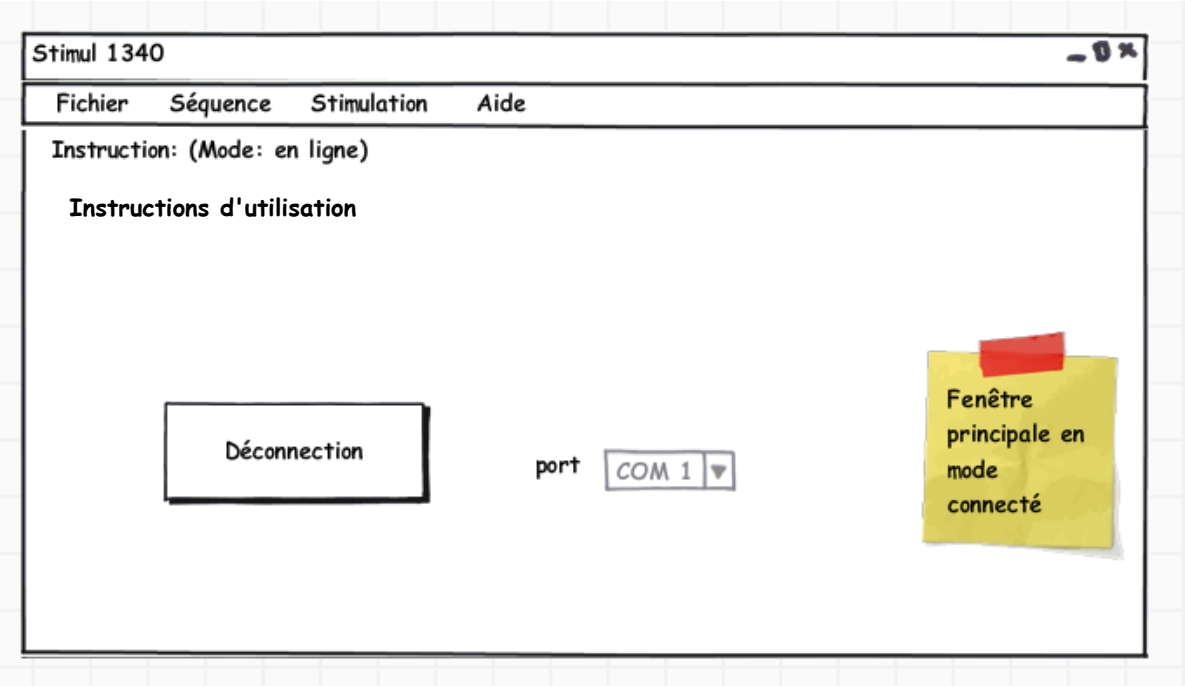


Figure 20: Maquette écran d'accueil en utilisation connectée

Ci dessous, voici l'écran de paramétrage et d'utilisation du mode manuel (disponible en utilisation connectée uniquement):

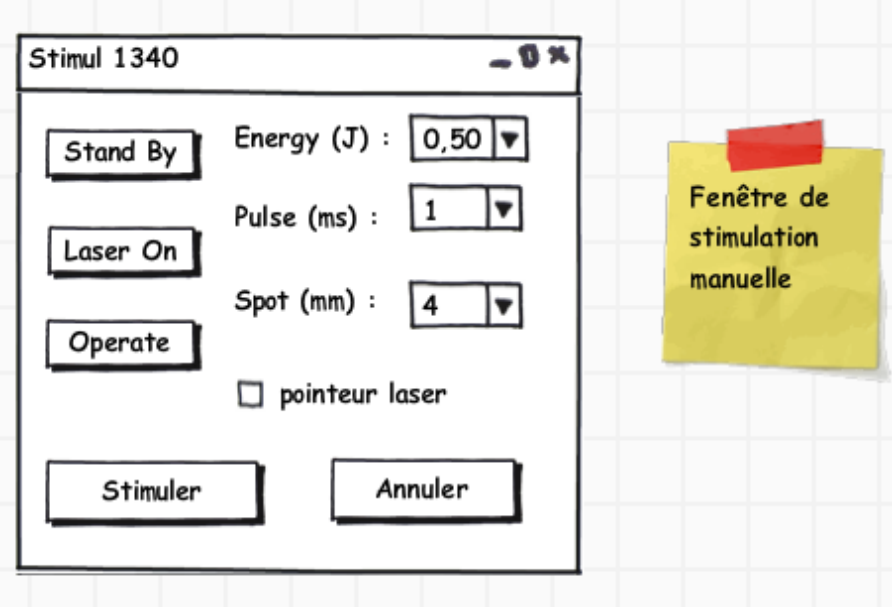


Figure 21: Maquette panneau de configuration mode manuel

Enfin, voici la maquette de création de séquences (disponible en utilisation connectée et déconnectée).

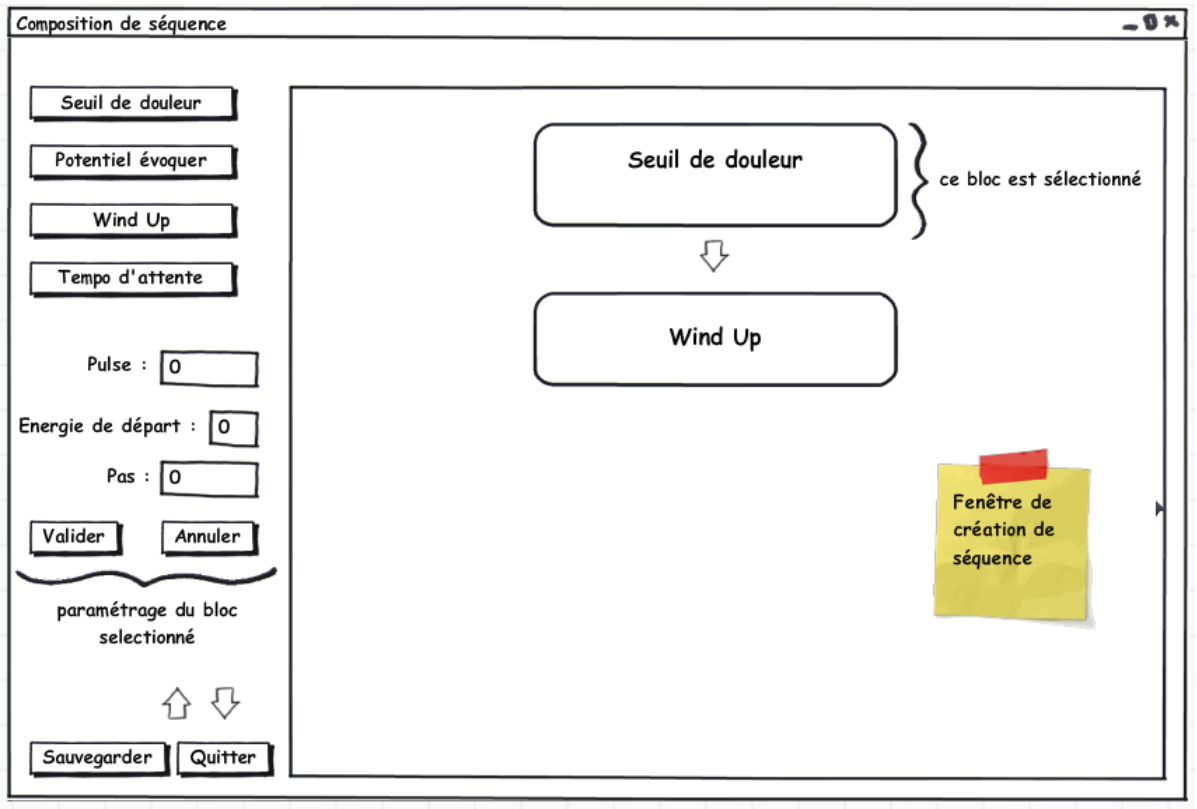


Figure 22: Maquette création et modification de séquence

Grâce à ces différents prototypes d'écrans l'utilisateur voit aisément la philosophie ainsi que les possibilités qu'offre le programme.

5. Les bases du programme

La première étape de mon travail à donc consisté à comprendre et à programmer la communication avec le laser. J'ai pu ainsi, dans un premier temps, l'utiliser manuellement.

5.1.La liaison série

Comme expliqué précédemment, la communication avec le laser s'effectue par le biais d'une liaison série RS232. La prise en main et la maîtrise de celle-ci furent les premiers défis. Le second fut de fiabiliser au mieux la communication. En effet, par nature, la liaison série n'est pas des plus fiable, mais en plus quelques manques se font sentir au niveau du protocole de communication du laser.

5.2.Diagramme de classe du Controller

Pour plus de flexibilité, le développement a appliqué au maximum le patron M.V.C (Modèle Vue Controller) qui a comme principe de découper le programme en trois grand axes :

- Les données (Modèle),
- Ce que l'utilisateur va voir (Vue)
- Le "cerveau" du programme (Controller).

En tenant compte du patron M.V.C, nous pouvons dès à présent déterminer une partie des objets avec lesquels nous allons travailler par la suite :

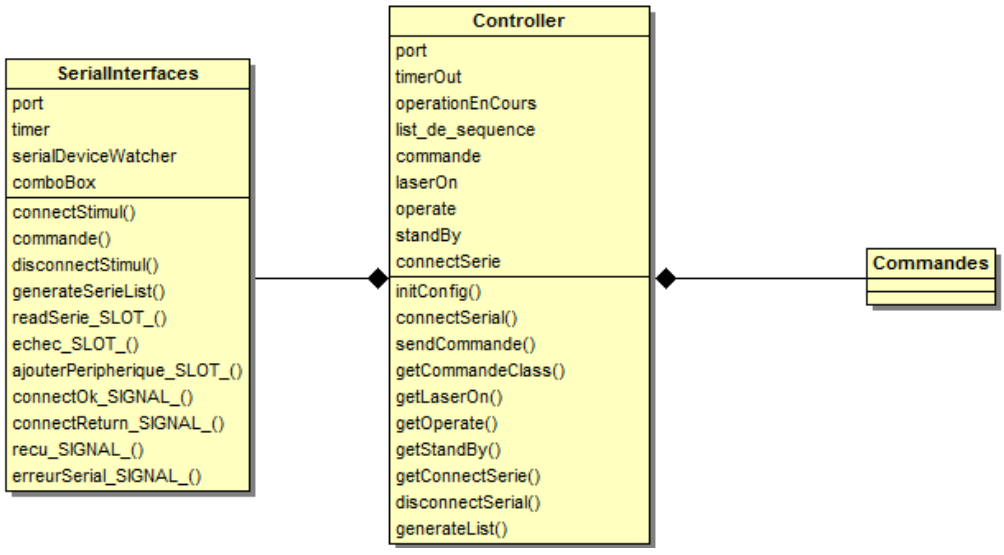


Figure 23: Diagramme de classe

Le diagramme ci-dessus n'est pas encore complet. En effet à ce stade du projet, l'agencement entre les classes est ce qui nous intéresse le plus. Ainsi les prochains diagrammes de classe verront apparaitre des méthodes, slot et signaux nouveaux. Les classes *SerialInterfaces, Controller et Commande* sont toutes les trois des contrôleurs. Elles sont séparées pour permettre une maintenance plus aisée.

5.3.Le protocole de communication

Le laser utilise un protocole de communication particulier qui est normalement fourni par le constructeur. Après quelques essais, il s'est avéré que la documentation était erronée. Par conséquent, tous les protocoles de communication ont du être vérifiés et corrigés. Pour ce faire l'utilisation de l'outil *Free Serial Port Monitor* a été indispensable. En effet celui-ci permet d'afficher les communications passant à travers un port série donné. Grâce à cela, la vérification des trames envoyées et reçues est aisée et sans appel.

Après de multiples tests, le protocole de communication a été entièrement corrigé. Pour éviter ce travail fastidieux à d'autres développeurs, la correction a été envoyée au fabricant pour mise à jour de la documentation.

5.4.Choix de la solution

Pour permettre au programme de communiquer avec le laser, mon choix s'est tout d'abord porté sur la bibliothèque *QExtSerialPort.* En effet après des recherches sur internet, il s'est avéré que cette bibliothèque s'adapte très bien à Qt en utilisant les slot et les signaux.

J'ai donc débuté mon développement avec celle-ci, jusqu'à découvrir que la bibliothèque me générait des problèmes.

J'ai alors fait le choix de changer pour QSerialDevice qui est plus stable tout en s'adaptant aussi bien à Qt. Cette opération ne m'a pas fait perdre trop de temps grâce au découpage choisi et explicité au diagramme de classe précédent.

6. Utilisation manuelle

6.1. Fonctionnement global.

L'interface manuelle du programme est la réplique exacte du panneau frontal du laser (présenté précédemment). Ainsi on retrouve les trois boutons (*standBy, laser on, operate*) ainsi que les quatre paramètres (*pulse, energy, pointer laser et spot*). Le paramétrage du son émis lors d'une stimulation n'est pas possible (non prévu par le constructeur du laser), il faut donc passer par le panneau de contrôle du Laser.

Le fonctionnement lui aussi reprend celui du laser. Pour envoyer une stimulation, il faut :

- Appuyer sur *laser On.*
- Attendre le chargement de la source laser.
- Renseigner les paramètres souhaités
- Appuyer sur *operate,* ce qui va lancer une procédure de calibration du laser
- Envoyer une stimulation (à condition que l'utilisateur appuie en même temps sur la pédale de sécurité).

Pour stopper la source laser, l'utilisateur pourra appuyer sur le bouton *Standby.*
L'interface reprend aussi le principe du bouton d'arrêt d'urgence.

6.2. Algorithme et fonctionnement

A présent, même si beaucoup de choses ont été comprises, corrigées et acquises, il reste une étape cruciale avant le développement. Il faut préétablir des algorithmes solides nous permettant de gérer tous les cas possibles et imaginables. Pour ce faire, en plus de l'écriture des algorithmes, les diagrammes de séquence semblent être l'outil le plus adapté (Certain vous sont présentés en annexe).

6.2.1. Algorithme de connexion

(Voir diagramme de séquence de connexion dans l'annexe p.46)

- L'utilisateur allume le laser
- Initialisation du laser
- L'utilisateur appuie sur la touche *serial* du laser,
- Le laser émet des 'P' pendant six seconde à raison de un par seconde.
- Pour être connecté le programme doit envoyer un 'P' pendant ces 6 secondes.

6.2.2. Algorithme de chargement du laserOn

Une fois la connexion établie pour mettre en marche la source Laser, il faut envoyer la trame L111. Le laser nous renvoie L001 lorsque la commande a été reçue et qu'il a lancé le chargement du laser. Il renvoie L000 s'il y a un problème ou que le laser est déjà chargé.
Enfin, quand la source laser est bien chargée, l'application doit recevoir L111.

6.2.3. Algorithme de calibration (Operate)

Au niveau du laser, cette opération est découpée en quatre temps et ne peut s'effectuer que si l'opération laserOn s'est bien déroulée.

- Premièrement on envoie la trame O111 qui signifie que l'on souhaite activer Operate. Si le laser renvoie O000, c'est qu'il n'est pas prêt (par exemple si le laserOn n'a pas été effectué). Si au contraire il renvoie O111 le passage à la seconde étape est possible.

- Cette deuxième phase consiste à envoyer les paramètres (taille du spot, pulse et énergie) au laser, grâce à la trame Pabc (a : pulse b : énergie c : spot). Si cette phase est réussie, le laser nous renvoie la chaine PZZZ et nous pouvons ainsi passer à l'étape de calibration

  Lors de cette phase de calibration, pour des raisons de sécurité, nous devons renseigner à nouveau les valeurs du pulse et de l'énergie. Il faut donc envoyer la trame Cabc (a : pulse, b : énergie, c : quelconque). Si l'opération de calibration à bien été prise en compte, le laser nous renvoie la même trame. Une fois la calibration terminée (elle peut prendre jusqu'à 2 minutes) le laser renvoie une trame commençant par V et reprenant les paramètres précédents.
- Enfin la dernière étape consiste à renvoyer la même trame de paramétrage qu'à l'étape deux pour vérifier la concordance de toutes les données envoyées.

6.2.4. Algorithme de stimulation

Une fois le laser chargé on peut envoyer une stimulation. Il faut pour cela envoyer la trame G111. Cependant, la stimulation ne sera réellement envoyée que si l'utilisateur a enclenché la pédale de sécurité.
A cet instant trois phénomènes sont possibles :

Figure 24: Pédale de sécurité

- Soit le laser nous répond la même chose, dans ce cas la stimulation à bien été envoyée
- Soit la trame reçue est G101, qui signifie que la stimulation ne peut pas être mise en place car une autre opération (laserOn ou calibration) est en cours
- Soit le laser renvoie G000 qui signifie qu'il y a eu un problème. La cause de celui-ci peut varier, cela peut provenir du fait que l'utilisateur a oublié de presser la pédale, que la calibration n'a pas encore eu lieu ou bien qu'une mauvaise trame à été envoyée.

6.3.Réalisation

La liaison série n'étant pas des plus fiables et le protocole de communication n'étant pas forcement des plus complets, l'implémentation des algorithmes ci-dessus a nécessité la mise en place de systèmes pour fiabiliser la communication.

a) *Etat de la machine*

La première chose à vérifier lors du lancement de l'interface manuelle est que la connexion avec le laser a bien été établie.

```
310     /// Vérifie le statut de la connection au prés du controller
311     if(controle->getConnect())
312     {
313         lancementStimul();
314     }
315     else
316     {
317         /// Si la connection n'est pas établie pour le controller on gére l'erreur
318         erreur->errorSimul(NEED_CONNECT_MODE);
319     }
```

Figure 25 : Première vérification de connexion

Par la suite, une deuxième sécurité permet de s'assurer que la connexion est toujours opérationnelle. Pour cela on envoie une trame au système pour lui demander son état (ÌS888[1]). Si celui-ci répond on analyse sa réponse et on agit en conséquence. Si le laser est déconnecté aucune trame ne sera émise et le système de time out et de temporisation prendra le relais.

b) *Time out et temporisation*

Ce deuxième test est utilisé pour s'assurer que le logiciel ne restera pas bloqué en attente d'une réponse, si le laser ne réagit plus ou si il a été brutalement déconnecté.
Le principe est simple : à chaque envoi d'une trame, un *chronomètre* va être enclenché. Si à un temps donné (time out) rien ne s'est passé, on considère que la tentative de communication a échoué et on en averti l'utilisateur.

Ce système est, comme beaucoup de choses dans Qt, basé sur les *signaux* et les *slot* de la manière suivante :

- On crée le *chronomètre* ou plus exactement le *timer* : ***QTimer *chrono = new QTimer();***
- On le lance en lui passant en paramètre une valeur de time out (ms): ***chrono->start(1000);***
- Ensuite on fait une connexion entre le signal émis par notre chronomètre et un slot qui n'est ni plus ni moins que l'action que l'on veut voir se produire lors de l'émission du signal : ***connect(chrono,SIGNAL(timeOut()),this,SLOT(problem()));***

- Si jamais le time out est atteint le programme exécutera l'action *problem()*. Donc si il n'y a aucun problème il ne faut pas oublier d'arrêter le chrono: ***chrono->stop();***

En utilisant quasiment le même principe, il a aussi fallu mettre en place des temporisations entre chaque envoi de trame pour ainsi permettre au laser d'avoir le temps de recevoir et de réagir correctement à chacune d'entre elles.

c) Système de question réponse

Dans un système tel que celui-ci, un des problèmes fondamentaux est de savoir quand et quelle trame envoyer. Pour ce faire, le meilleur moyen est de constituer un système le plus synchronisé possible.

La première chose à faire après avoir connecté le logiciel est donc d'écouter le port série en permanence. Ainsi, chaque fois qu'une trame arrivera du laser, le *Controller* sera mis au courant par le biais des signaux et pourra agir en conséquence.

Cette première étape va nous permettre de rendre notre système plus fiable. Avant chaque étape cruciale (*laserOn*, *Operate*, ...) un identifiant (ici nommé *operationEnCours*) se voit attribuer une valeur. Lorsque qu'une trame est reçue, le *Controller* regarde cet identifiant et agit ainsi en conséquence. De cette manière, lorsqu'une trame n'a eu aucun effet sur le système alors qu'elle aurait du, le logiciel se charge de réémettre la trame. Au contraire si le laser informe d'une erreur, le logiciel pourra se mettre en position d'erreur de façon presque instantanée.

d) Conclusion

Voici le résultat final en image. Du point de vue fonctionnel, après plusieurs dizaines d'essai de la version finale, aucun bug n'a été rencontré. Un résultat qui est donc assez satisfaisant.

(Vous pouvez voir en annexe l'évolution de l'interface tout au long du projet)

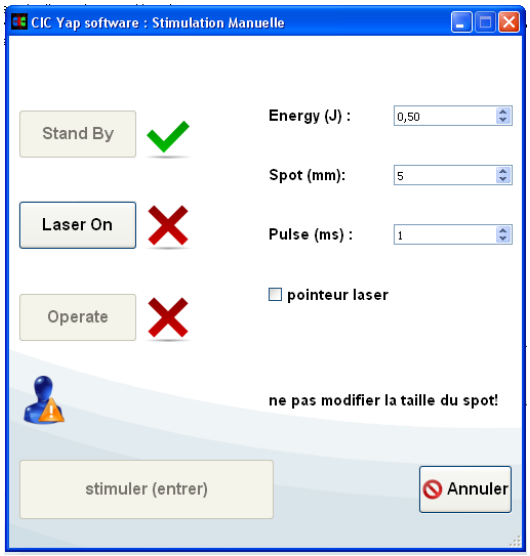


Figure 26 : Interface manuelle final

7. Utilisation automatique

Maintenant que la procédure manuelle fonctionne correctement, l'automatisation est possible. Toute fois elle implique de nouvelles notions. Commençons déjà par bien cerner ce qu'est l'automatisation.

7.1. Fonctionnement Global

Dans notre cas cette phase d'automatisation peut être coupée en deux :

- D'un coté, une partie de création, de modification, de suppression et d'enregistrement de séquences.
- De l'autre le lancement de ces séquences.

Autant au niveau du développement que de l'utilisation, ces deux axes sont très distincts. En effet lorsque que l'utilisateur lance une séquence, il faut qu'il soit dans l'impossibilité de la modifier pour éviter que le design du protocole soit modifié en cours d'étude.

7.2. Gestion des séquences

7.2.1. Analyse de base

Maintenant que les fonctionnalités sont définies et cadrées, nous pouvons commencer l'analyse des objets que nous utiliserons. Ce mode de fonctionnement introduit plusieurs nouvelles classes.

Tout d'abord, nous allons trouver dans le modèle la description des séquences et des blocs. Du coté du controller, on ajoute la possibilité de créer, supprimer ou modifier ces séquences. On va aussi devoir gérer les sauvegardes de ces séquences. Du côté de la vue, nous allons bien entendu avoir de nouveaux écrans permettant à l'utilisateur d'exploiter toutes ces nouvelles fonctions. Nous obtenons alors une structure comme ci-dessous (le diagramme de classe n'est pas complet car le nombre d'attributs et de méthodes rendrait ce dernier trop lourd, ici seul la structure est représenté)

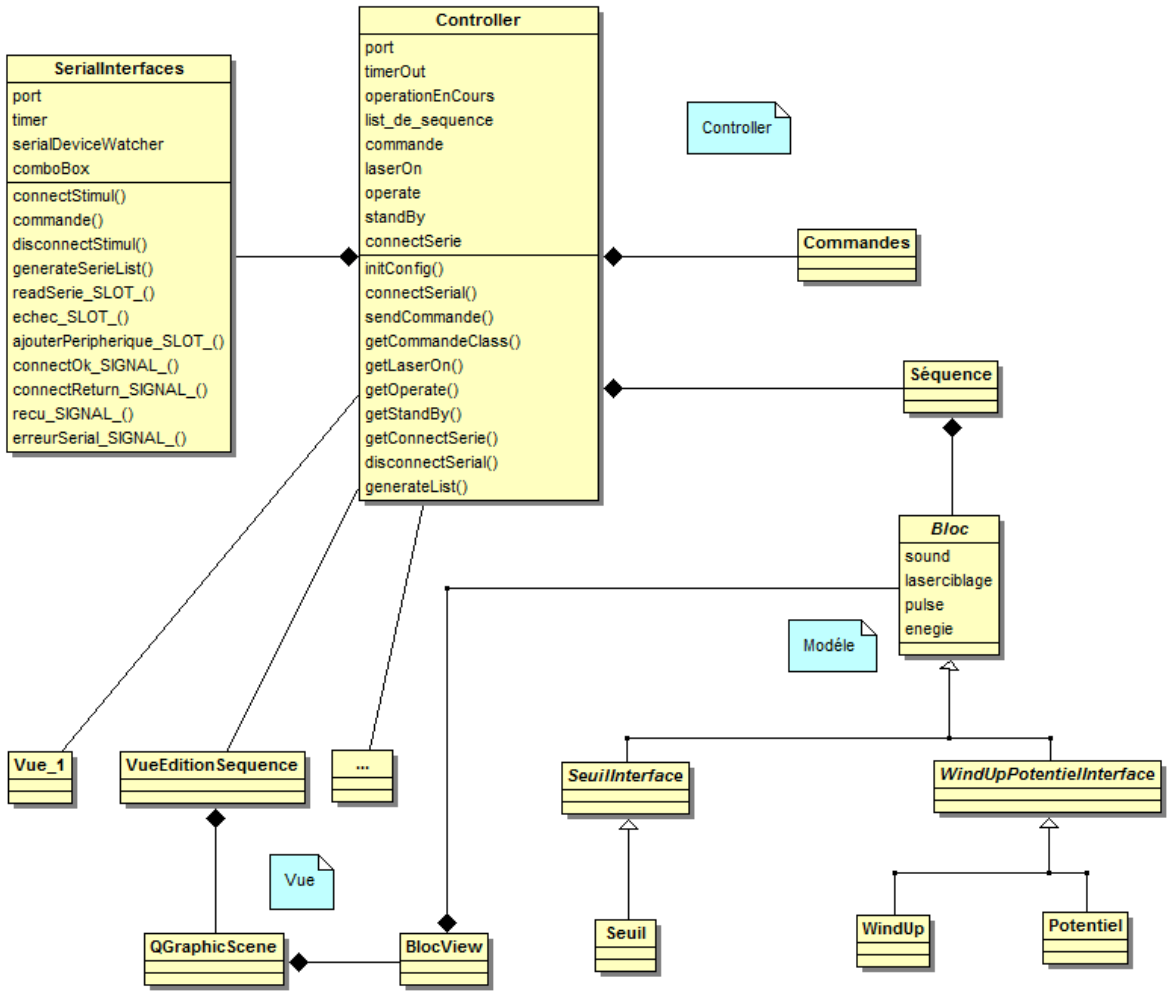


Figure 27 : Diagramme de classes M.V.C.

7.2.2. Stockage de données

La possibilité de sauvegarder des séquences d'une utilisation à une autre est un point indispensable. Pour ce faire l'utilisation du X.M.L. *(Extensible Markup Language)* semble être la méthode la plus appropriée.

Le fichier de sauvegarde devant contenir les différentes séquences ainsi que les blocs qui les composent est organisé de la manière suivante :

```
<? Doctype ?>
<sequences>
      <sequence nom="" validation="" nomAuteur="" tailleSpot="">
            <blocs>
                  <bloc nom="">
                        <param continuer="" laser="" pulse="" energy=""
pourcent="" frequencePas="" nombreStimulations="" />
                  </bloc>
                  <bloc (...)
                  </bloc>
            </blocs>
      </sequence>
      <sequence (...)
      </sequence>
</sequence>
```

On retrouve ainsi toutes nos données. De nouveaux éléments ont aussi fait leur apparition. Ainsi, l'attribut de séquence *validation* aura la valeur vrai si la séquence est valide, et la valeur faux si la séquence a besoin de modifications avant d'être chargée.

Le bloc *param* contient lui aussi des éléments à détailler :

- Tout d'abord l'attribut *continuer* : c'est un booléen (vrai ou faux) qui indique s'il faut demander ou non une confirmation à l'utilisateur avant de lancer le bloc.
- Ensuite les champs *frequencePas* et *nombreStimulation* permettent respectivement d'indiquer la fréquence des impulsions et le nombre de stimulations à effectuer une fois la séquence lancée

7.2.3. Réalisation

A ce moment précis du développement il va donc falloir mettre en place deux nouveaux concepts.

a) *Gestion des séquences*

Une séquence est un enchainement des trois opérations explicitées plus haut (seuil de douleur, potentiel évoqué et wind up). Cependant, comment permettre à un utilisateur novice en informatique de créer, et de modifier des séquences parfois complexes ?

La solution adoptée à été de proposer à l'utilisateur des blocs entièrement paramétrables et ayants chacun des actions différentes. Pour une flexibilité accrue, en plus des trois actions par défaut, nous allons retrouver deux nouveaux blocs :

- Un bloc nommé *Période réfractaire* qui n'est ni plus ni moins qu'un bloc d'attente
- Un *bloc manuel* permettant la mise en place de protocoles ou d'opérations qui n'ont pas été prévus à la base.

Pour créer une interface simple et efficace, le programme utilise les QGraphicsView de Qt. Les événements de fermeture et de sélection ont été surchargés pour permettre la mise en place des opérations de suppression de blocs, ou de sauvegarde.

b) *Sauvegarde des séquences*

Maintenant que l'interface à été créé et qu'elle est fonctionnelle il a fallut s'occuper de la mise en place du système de sauvegarde. Tout d'abord il a fallut choisir l'analyseur X.M.L. (ou parseur). Les deux plus connus sont S.A.X. (*Simple API for XML*) qui est axé événementiel, et D.O.M. (*Document Object Model)* qui utilise une approche hiérarchique. Le choix s'est donc porté sur D.O.M. En effet le coté événementiel du S.A.X. ne nous étant d'aucune utilité, D.O.M. est la solution la plus rapide à mettre en place de part sa simplicité.

Une fois ce choix fait il ne reste plus qu'à gérer le fichier de sauvegarde.

Tout d'abord, au démarrage du logiciel, le fichier sert de base pour recharger toutes les séquences enregistrées. Si ce fichier n'existe pas, il est automatiquement crée.

Ensuite au niveau de la sauvegarde à proprement parler, celle-ci intervient lors de la fermeture de la fenêtre de modification ou de création de séquence. En effet pour permettre à l'utilisateur de choisir s'il veut garder ou non ces modifications, toutes les actions de modification ou de création de séquence sont effectuées sur des séquences factices. A la fin de la session de modification, les vraies séquences sont comparées aux séquences factices . S'il y a une différence on propose à l'utilisateur de sauvegarder. Si celui-ci accepte, la séquence factice va remplacer l'ancienne.

7.3.Automatisation des séquences

Maintenant que la gestion des séquences est opérationnelle et que notre interface manuelle l'est aussi il ne reste aucun obstacle à la création du module d'automatisation de séquences. En effet grâce aux bases solides posées jusqu'ici, il ne reste plus qu'a implémenter les algorithmes de base du seuil de douleur, du wind up, du potentiel évoqué, du bloc manuel et de la période réfractaire.

Figure 28: Ecran de séquence automatisée

8. Déploiement et formation

Ces deux objectifs n'étaient pas les priorités de ce stage mais ce sont des points très importants pour le développement du logiciel.

8.1. Création d'un installateur

Un parc informatique n'est jamais inerte. Des anciennes machines partent, des nouvelles arrivent. C'est pour cela que la possibilité d'installer rapidement et simplement une application est très importante. La création d'un installeur s'impose donc.

Dans un premier temps le choix c'est porté sur NSIS, très efficace et flexible. Le développement d'un installateur demandait cependant beaucoup de temps avec cet outil. De plus, il fallait que je me forme aux scripts d'utilisation de NSIS. Enfin, cet outil n'est pas multiplateforme.

Disposant d'une licence LGPLv3 le choix c'est finalement porté sur BitRock qui est normalement payant, mais qui fournit gratuitement des licences aux projets libres. Tout en étant aussi flexible, le paramétrage et la mise en place peuvent se passer de scripts, ce qui rend le développement de ce dernier beaucoup plus rapide.

8.2. Formation et documentation

Pour permettre une meilleure prise en main, la formation et la documentation du logiciel a été indispensable, autant au niveau développeur qu'au niveau utilisateur.

Pour ce faire deux types de documentation ont été mises en place.

- Une documentation HTML 5 expliquant comment utiliser les différentes options du logiciel (certaines par des tutoriaux vidéo).
- Une documentation plus axée développeur administrateur, et par conséquent disponible à la racine de l'installation.

Cette documentation a été réalisée par l'intermédiaire de Doxygéne.

En plus de ces deux documents, une liste des sites qui ont été utilisés tout au long de ce stage est disponible sur la documentation développeur.

A côté de ces documentations, certains utilisateurs ont été formés à l'utilisation du logiciel.

9. Choix de licence

L'aspect juridique est de plus en plus important. Les décharges, la protection des productions intellectuelles, ou encore la collecte de données sont actuellement des points de discorde importants et parfois contraignants. C'est pour toutes ces raisons que l'attribution d'une licence adaptée à ce projet a été indispensable. Comment fonctionnent les licences ? Laquelle choisir? Quel impact cela aura–t-il sur l'éventuelle distribution du logiciel?

9.1. Définition des besoins

Avant tout choix de licence, il est indispensable de définir quel est le besoin de l'entreprise et ce qu'elle attend d'une licence. Pour ce faire, la meilleure solution fût d'organiser une réunion avec les personnes influant ce genre de décision. Tout d'abord deux grandes " familles" ont été proposées :

- Celle du *Propriétaire,* à but lucratif, où les conditions et leurs applications sont à la charge de l'entreprise.
- Celle du Libre, ou les sources sont accessibles par tous, et qui embarque préalablement ses propres conditions.

Après avoir discuté des attentes, deux points principaux sont ressortis:

- Obligation de citation
- Suivi des potentielles évolutions du logiciel

L'objectif principal du C.I.C n'est pas de rentabiliser cette application, ou de faire du profit grâce à celle-ci, mais plutôt de se faire connaître par un plus grand nombre grâce à l'obligation de citation.

Le choix s'est donc porté sur le Libre.

Malgré un nombre impressionnant de licences répondant à nos deux critères, six d'entre elles, les plus connues et/ou les plus fiables ont été retenues. Parmi elle : MIT, BSD, GPL, LGPL,... . Après une étude approfondie, le choix c'est porté sur LGPLv3 (Lesser General Public Licence version 3).

9.2. Distribution du logiciel

Pour distribuer et suivre l'évolution du logiciel, l'installateur ainsi que les sources ont été placés sur le site de Google code. Ainsi grâce au serveur SVN qui s'y trouve, l'évolution du logiciel reste maitrisable sans être bridée. L'adresse ou est hébergée le projet est la suivante : http://code.google.com/p/cic-yap-software/

10. Bilan technique

Le bilan de ce stage est très positif. En effet, la après une phase d'appropriation du projet, d'analyse et de développement, l'utilitaire est opérationnel et rempli toutes les fonctions souhaitées.

Le développement de l'interface de communication avec le laser est une réussite et les systèmes de *fiabilisation* remplissent parfaitement leurs rôles. Tous les outils d'analyse de trame prouvent d'ailleurs que les trames envoyées et reçues sont cohérentes et correctes.

La mise en place du mode manuel reprenant la logique et l'utilisation du laser en mode manuel devient intuitive et simple. Les messages en cas d'erreurs et les indications d'utilisation semblent clairs et bien appropriés d'après les réactions des testeurs (non-informaticiens).

La gestion des connections et déconnections ainsi que les conséquences qui en découlent sont parfaitement gérées. En mode déconnecté toutes les zones dépendantes du laser ne sont pas accessibles (charger une séquence automatique et utilisation manuelle)

La gestion des séquences est elle aussi un succès, après des tests poussés de la version finale du projet. En effet, aucune erreur n'a été détectée, que ce soit au niveau de la création, de la modification ou de la suppression des séquences.

Au niveau plus technique le code est de plus en plus clair et est de plus en plus proche d'un M.V.C parfait. Cependant, pour gagner du temps, certains patterns qui auraient pu être mis en place ne sont pas présents.

A cette heure, le logiciel semble stable, cependant il devra être testé en conditions réelles et pendant un certain temps, avant que l'on puisse l'affirmer de façon ferme et définitive. Bien que les besoins exprimés aient été comblés, le logiciel pourrait être encore amélioré sur plusieurs points.

Tout d'abord au niveau de la gestion des erreurs, actuellement très présente. Les messages, certes suffisants, sont actuellement trop généralistes.

Ensuite au niveau de la gestion du fichier de sauvegarde, on pourrait envisager la possibilité de l'exporter, ou d'en charger un nouveau. De plus, un système de fichier temporaire, comme dans des logiciel tel que office, pourrait être mis en place pour permettre de conserver un maximum de données au cas où l'ordinateur de l'utilisateur boguerait pendant la sauvegarde.

Cependant comme vous avez pu le constater les améliorations restent minces et ne sont que des façons de rendre encore un peu plus confortable l'utilisation et/ou la maintenance.

Conclusion

Cette expérience fût l'une des plus formatrice et enrichissante pour ma carrière informatique, autant au niveau humain que technique.

Premièrement d'un point de vue culturel, puisque le lieu, les professions ainsi que le domaine dans lequel je suis resté immergé pendant dix semaines, m'étaient complètement inconnus. De plus mon sujet demandait des connaissances dans le domaine de la douleur que j'ai dues m'approprier très rapidement. Cela m'a permis de travailler ma capacité d'adaptation.

Le projet étant long, à plein temps avec une obligation de résultat plus forte que dans le milieu scolaire, j'ai appris à prendre plus de recul, quitte a perdre quelques heures au démarrage pour en gagner le double voir le triple tout au long du projet.

Le contact avec les utilisateurs a aussi été très enrichissant. En effet réussir à cerner le vocabulaire à utiliser, la logique ainsi que l'ergonomie pour que le produit plaise à tout le monde, n'est pas la tâche la plus évidente qui soit.

D'un point de vue plus technique j'ai pu améliorer mes connaissances en Qt et en C++ en utilisant des concepts qui m'étaient encore jusque là inconnus ou juste survolés.

Pour résumer, ce stage a été très complet depuis l'analyse et la discussion avec l'utilisateur jusqu'au développement en passant même par une partie plus juridique qu'est le choix de la licence. De plus l'ambiance de travail a été des plus agréable et favorable.

English Summary

I'm currently a student in the *Institut Universitaire de Technologie d'Auvergne*. To validate my diploma, I did a placement for ten weeks in *the Centre d'Investigation Clinique (C.I.C.) de Clermont-Ferrand*. My tutor was Ms. Alice MARTIN who is an engineer. I will summarize this experience. Firstly I will explain my first days in this place and how this service works. Secondly I will present the needs of the C.I.C. team and how I reacted to satisfy them. Thirdly I will introduce you to the license choices and their reasons before concluding.

As written before, my placement was in a laboratory in Clermont-Ferrand. This centre is part of *Centre Hospitalier Universitaire (C.H.U.)* de Clermont-Ferrand and it makes studies about the pain. It has lot of medical instrument like resuscitation cart, patient monitors, stimulation machines etc. At the beginning of my job I met the team. I immediately tried to take its logic to make a software like they wanted and to understand how this service operates, which is why, during the first week I attended several reunions. I learnt the usual clinical test protocol which is divided into four parts. This centre can make all parts and it is swear to try new molecules on human. At this moment, I understood how C.I.C. works, hence focusing on what they need.

The first question before analysis and development has been " Will this software be useful? " , the answer was yes because thanks to it, doctors will gain time, efficiency and precision. During meetings and interviews, I kept three words which were: simple, efficiency and user friendly. I lead my analysis and my development to respect what they had asked for. Moreover they had lot of needs. Firstly the software can be operated in manual mode, secondly it can automate lot of operations and simplify the doctors' work, thirdly I can give lot of information to guide the user, etc. For a start, I began by a good analysis in order to forget nothing. After I developed the software with listening to the users to create something they like.

After this development the license question presented itself/was a necessary step. After reading documentation and knowing what the C.I.C. expected, I chose the LGPLv3 license. Thanks to it, all people can use the sources or the software if they say who created it. Moreover the C.I.C. is protected if there is any accident with this software.

As a conclusion, this placement has been very interesting and very multidisciplinary. I was able to act on every aspect throughout this development. I gained in autonomous and in technical skills. Moreover I liked the human contact with the future users. This was a very good experience and I thank Ms Alice MARTIN and the C.I.C. for this welcome.

Lexique

Design pattern : C'est un parton de conception ou motif de conception. Ce sont des concepts destinés à régler les problèmes récurrents en utilisant toutes les ressources de l'objet

Neuroscience : ensemble des disciplines biologiques et médicales qui étudient tous les aspects des neurones et du système nerveux.

Object Management Group : association américaine à but non-lucratif créée en 1989 dont l'objectif et de standardiser et promouvoir le modèle objet sous toutes ses formes.

Pharmacologie clinique : Discipline scientifique du vivant étudiant les interactions entre une substance active et l'organisme dans lequel il évolue. Souvent utilisée pour créer ou améliorer des médicaments. Cette discipline peut aussi être étendue à l'étude des effets et des interactions médicamenteuses.

Analyseur X.M.L. (parseur) : Outils permettant de parcourir un document X.M.L. et d'en récolter les informations.

Bibliographie

balsamiq. (s.d.). Récupéré sur http://www.balsamiq.com/builds/mockups-web-demo

C.H.U. (s.d.). Récupéré sur http://www.chu-clermontferrand.fr

C.I.C. (s.d.). Récupéré sur http://cic-cpc.chu-clermontferrand.fr

C++. (s.d.). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/C%2B%2B

D.U.T. (s.d.). Récupéré sur http://iutweb.u-clermont1.fr

Wikipedia. (s.d.). Récupéré sur http://fr.wikipedia.org/wiki/Object_Management_Group

Wikipedia/laser. (s.d.). Récupéré sur Wikipedia: http://fr.wikipedia.org/wiki/Laser

Table des figures

Annexe

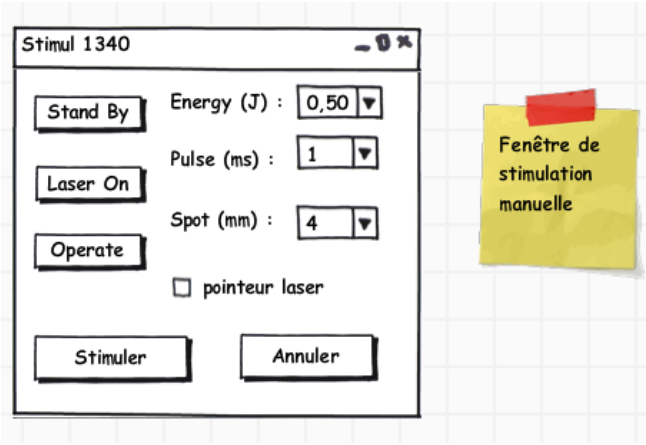


Figure 29: Maquette interface manuelle

Figure 30: Première version de l'interface

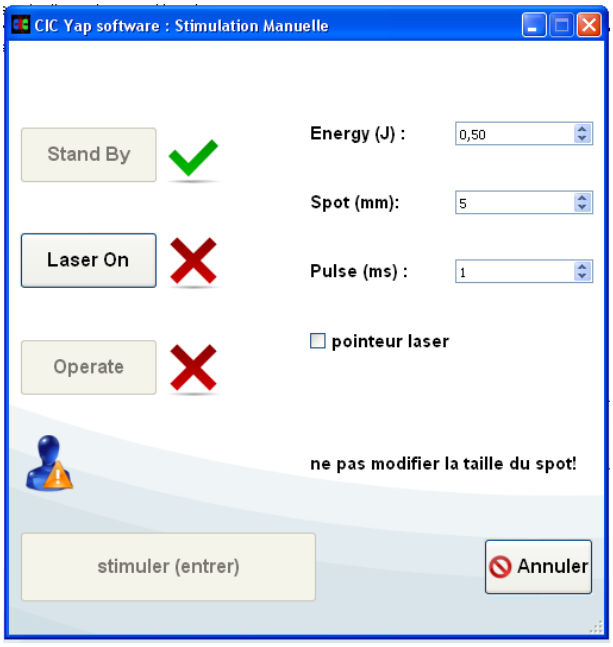


Figure 31: Version finale de l'interface

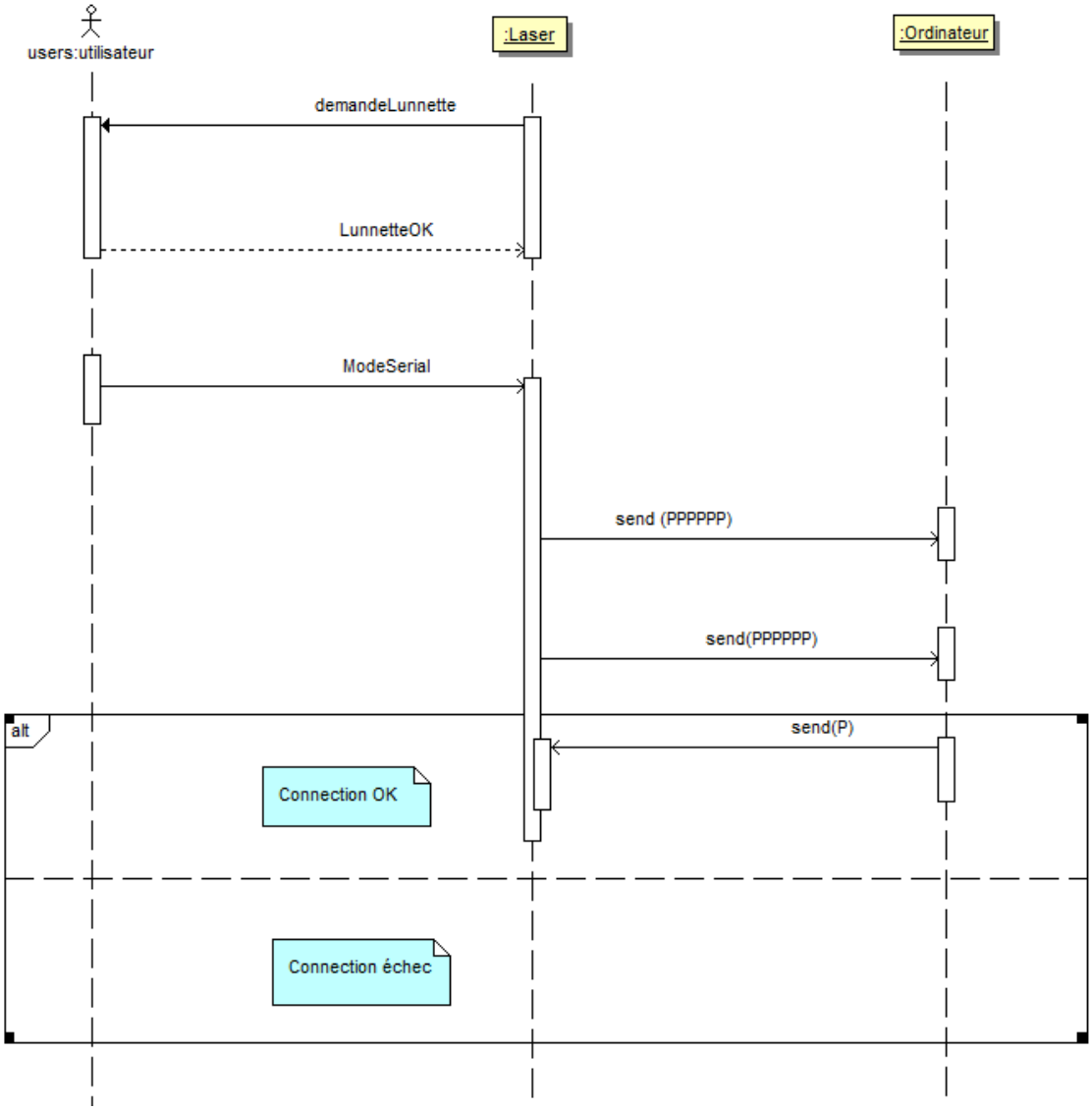


Figure 32 : Diagramme de séquence de connexion