INSA ROUEN
À taille humaine,
à l'échelle du monde

Projet de Physique P6-3

STPI/P6-3/2011 – 48

PROGRAMMATION D'UNE LAMPE UV POUR LE CONTROLE D'UNE REACTION DE POLYMERISATION PHOTO INDUITE

Etudiants :

Lancelot BOULET — **Jérémy BOYDEN**

Amélie DRAGEE — **Lihao LIU**

Margot LUCET — **Jérémy RENAUX**

Jiexi ZANG

Enseignant-responsable du projet :

Philippe LEBAUDY

Date de remise du rapport :

17/06/11

Référence du projet :

STPI/P6-3/2011 – 48

Intitulé du projet :

Programmation d'une lampe UV pour le contrôle d'une réaction de polymérisation photo induite

Type de projet :

Programmation

Objectifs du projet :

L'enseignant responsable de ce projet a mené des recherches pour déterminer théoriquement la fonction que doit suivre l'intensité de la lumière pour optimiser une réaction de polymérisation. La réaction consiste à solidifier une plaque de polymère en faisant varier l'intensité de la lampe puisque le fond du récipient à polymère ne doit pas être éclairé de la même façon que le dessus.

Il s'agit donc de programmer une lampe UV de façon à ce qu'elle suive une fonction mathématique. Le programme doit pouvoir contrôler la lampe et permettre d'avoir à disposition une application pratique et simple à utiliser.

Mots-clefs du projet :

Lampe UV – polymérisation – programmation – sigmoïde

Si existant, n° cahier de laboratoire associé :

Pas de cahier

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE ROUEN

Département Sciences et Techniques Pour l'Ingénieur

BP 8 – avenue de l'Université - 76801 Saint Etienne du Rouvray - tél : +33(0) 2 32 95 66 21 - fax : +33(0) 2 32 95 66 31

TABLE DES MATIERES

NOTATIONS, ACRONYMES

UV : Ultra Violet

Visual Basic : langage de programmation de type WYSIWYG (what you see is what you get, c'est-à-dire qu'on créé les fenêtres directement).

MatLab : langage de programmation et environnement de développement permettant de faire de nombreux calculs mathématiques mais aussi d'exploiter les ports « série » de l'ordinateur.

1. INTRODUCTION

Comme nous l'avons dit précédemment, l'enseignant responsable du projet mène des recherches sur la polymérisation photo induite. Ces réactions sont déclenchées par l'exposition à un rayonnement UV.

Pour ce faire, il utilise une lampe UV, qui peut être contrôlée soit manuellement, soit par l'intermédiaire d'un ordinateur. Il a besoin que l'intensité de cette lampe suive une fonction précise au cours du temps ; or, on ne peut le faire manuellement.

Notre projet consiste donc à programmer, dans un langage adapté, la lampe de manière à ce que son intensité suive la fonction mathématique requise pour la réaction.

Idéalement, notre programme doit permettre d'utiliser n'importe quelle fonction.

Cependant, notre encadrant a souhaité que l'on se focalise d'abord sur une fonction de type sigmoïde (voir image 1) utile pour sa réaction de polymérisation; En effet la réaction consiste à solidifier un polymère transparent en l'éclairant d'un rayonnement ultra-violet.

Le problème résulte dans le fait que l'échantillon possède une certaine épaisseur. D'après la loi de Beer-Lambert ($I(\lambda, X) = I_0(\lambda).e^{-\alpha X}$), le fond du récipient n'est pas éclairé de la même façon que le dessus. De plus, à mesure que la réaction avance, le polymère change de structure, il durcit, transformant ainsi l'influence du rayonnement UV (réflexion différente). De plus, pour cette réaction, la longueur d'onde nécessaire est de 360nm, or pour éclairer à cette longueur d'onde avec une lampe à filament, il faudrait chauffer le filament à au moins 6000K, température à laquelle il serait complètement fondu. C'est la raison pour laquelle notre encadrant utilisait une lampe UV à vapeur de mercure. Cependant ce type de lampe ne permet pas un réglage de l'intensité lumineuse grâce à l'intensité électrique, il faut utiliser un système mécanique d'obturateur qui se règle pas à pas.

C'est donc avec une lampe de ce type qu'il a fallu commencer la programmation.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

2.1. Répartition des tâches

Pendant toute la durée du projet, nous avons occupé des postes différents afin de mener à bien et dans des conditions idéales la mission qui nous avait été confiée.

Amélie, tout d'abord, a écrit l'essentiel du programme sous Visual Basic puisqu'elle avait déjà quelques connaissances sur le logiciel. Les livres prêtés par notre professeur encadrant, M. Lebaudy, lui ont tout de même été d'une grande utilité pour réaliser la communication entre le programme et la lampe UV.

Margot aidait principalement Amélie dans la conception de notre programme. Elle a aussi réalisé l'un des algorithmes les plus importants du projet : la fonction sigmoïde, permettant à la lampe Hamamatsu d'augmenter son intensité au cours du temps des principales tâches du projet, et qui a été finalisé par Amélie.

Jeremy (Boyden), quant à lui, a réussi les premières communications entre l'ordinateur et la lampe UV grâce au logiciel Matlab. Il a aussi cherché les principales fonctions utilisées sous Visual Basic, permettant, par exemple, de prendre en compte la durée d'utilisation de la lampe ou la communication entre celle-ci et l'ordinateur.

Lancelot épaulait Amélie lors de la programmation en indiquant les erreurs dans les procédures ou lors des problèmes de compilation. Il effectuait également les tests sous le logiciel Synchronie lors des essais de la lampe grâce à un capteur optique, permettant de visualiser l'intensité évoluant au cours du temps.

Jiexi supervisait le fonctionnement de la lampe lors de chaque essai en notant les problèmes de coordination ordinateur/lampe ou à la lampe elle-même (comme le défaut de synchronisation de l'intensité entre la montée et la descente [expliqué plus loin dans le rapport]).

Jeremy (Renaux) assistait Margot dans la réalisation des algorithmes du programme, tout en supervisant, lui aussi, le travail de programmation effectué et détectait chaque problème lié aux erreurs de programmation, notamment les difficultés à unifier l'ordinateur et la lampe.

Cette répartition indique la fonction principale que chacun d'entre nous a occupée. Il est évident que ce partage des tâches n'était pas fixe et que nous avons régulièrement changé de poste tout au long des semaines, dans l'optique de mettre au maximum à profit cette expérience de groupe. Cependant, les futurs aspirants au département ASI ont été les plus efficaces sur l'ordinateur, étant bien plus à l'aise que les autres quant à l'utilisation du logiciel Visual Basic et à la programmation en général.

2.2. Carnet de bord

Le créneau horaire auquel notre groupe correspondait était le jeudi matin, de 8h00 à 9h30. A la fin de chaque séance, l'un d'entre nous rentrait chez lui avec notre « cahier de bord » et y inscrivait tout ce qui avait été vu lors de la précédente séance. De même, lors de chacune de celles-ci, si une grande avancée avait été faite dans la conception du programme informatique ou dans la réalisation du futur projet, elle y était aussitôt inscrite avec la date du jour et quelques informations la concernant.

C'est pourquoi, au terme de notre projet et après treize semaines de recherches et de réflexions, nous possédons un cahier résumant l'ensemble de notre parcours et de nos découvertes. Nous allons par ailleurs vous en faire un bref aperçu ci-dessous avec l'évolution décrite au fil des séances.

Première séance *(10/02/2011)*

Lors de ce premier rendez-vous, nous prenons tout d'abord connaissance des autres membres du groupe. En effet, depuis notre arrivée dans l'école, c'est la première fois que nous sommes répartis de manière totalement aléatoire dans l'élaboration et l'aboutissement d'un projet. Ceci fait, nous sommes mis en commun avec l'autre groupe du même créneau horaire et le responsable du projet nous donne un certain nombre d'informations et de consignes concernant les séances à venir. Il nous explique bien évidemment en quoi le projet va consister.

Seconde séance *(17/02/2011)*

Dans la continuité de la première réunion, le responsable finit de nous donner des précisions sur les réels objectifs à atteindre lors du projet, puis il nous présente l'ensemble des machines et autres matériels que nous utiliserons afin de mener à bien le projet. Parallèlement, nous recevons de la documentation qui nous sera utile tout au long des séances et une première recherche d'informations est abordée.

Troisième séance *(10/03/2011)*

Les présentations sont faites, les explications sont données, le travail peut donc réellement commencer. Nous nous réunissons pour la première fois dans la salle de spectrométrie dans le bâtiment Darwin qui constituera pour l'avenir, notre seule salle de travaux et de recherches. L'ensemble du groupe de travail effectue de nouveau des recherches sur le sujet posé et commence ainsi à véritablement cerner ce qui va lui être demandé. Au terme de cette séance, les premières idées émergent et un aperçu de la future structure organisationnelle est obtenu. Enfin, une première approche de la lampe UV est réalisée à l'aide du logiciel Matlab.

Quatrième séance *(17/03/2011)*

Il est clair que le logiciel en notre possession, Matlab, ne correspond pas tout à fait à nos travaux. Par conséquent, il est décidé, à l'aide de notre professeur encadrant, de travailler avec un tout autre logiciel, Visual Basic, avec lequel la programmation de la lampe devrait être beaucoup moins compliquée. L'équipe se divise pour la première fois, certains continuent d'avancer la programmation avec Matlab tandis que d'autres étudient le futur logiciel, Visual Basic.

Cinquième séance *(24/03/2011)*

Présentation du logiciel Visual Basics et première approche de l'ensemble des possibilités que peut offrir ce langage. L'ensemble du programme qui avait été créé sous Matlab est traduit en Visual Basics et ainsi, pour la première, nous avons un résultat concret en fin de séance puisque nous arrivons à communiquer en donnant des ordres simples. La programmation est réalisée sur l'ordinateur du responsable de projet.

A partir de cette séance, le groupe va désormais se scinder principalement en deux sous groupes qui auront pour responsabilité, d'une part, la rédaction des différents algorithmes nécessaires au programme et d'autre part, leur traduction en langage Visual Basic (jusqu'alors inconnu).

Sixième séance *(31/03/2011)*

La configuration de travail adoptée la semaine précédente est maintenue et on a une réelle avancée dans le projet, avec notamment, la mise en place de l'interface qui sera utilisée jusqu'à la fin du projet. A cet instant, le programme est capable d'allumer et d'éteindre la lampe.

Septième séance *(07/04/2011)*

Le but de cette réunion est la création des fonctions qui vont permettre de régler l'intensité de la lampe UV. Après traduction des algorithmes réalisés, nous pouvons mettre l'intensité à 0% ou à 100% et il est possible de l'augmenter ou de la baisser de 1% en fonction des choix de l'utilisateur. Cependant, il subsiste le problème d'un « blocage » à partir d'un certain seuil. En effet, l'intensité ne peut pas descendre manuellement sous les 33% et parallèlement, elle ne peut dépasser les 66%, et la plupart du temps, le programme ne répond plus sans que nous réussissions à déterminer pourquoi.

Huitième séance *(05/04/2011)*

Le groupe est une nouvelle fois scindé en deux afin de travailler simultanément sur l'algorithme et sur la programmation. Le sous-groupe chargé de programmer se concentre sur la résolution du problème résultant de la semaine précédente (problème qui va persister jusqu'à la fin) tandis que l'autre fait une grande avancée en écrivant l'algorithme complet de la variation de l'intensité en fonction du temps (fonction linéaire, polynôme du second degré et sigmoïde).

Neuvième séance *(12/05/2011)*

P*as de réunion car indisponibilité du responsable projet.*

Dixième séance *(19/05/2011)*

Algorithmiquement, toutes les fonctions sont complètes. Nous avons donc une nouvelle répartition des tâches. Certains s'occupent de chercher les erreurs présentes dans le programme afin de les corriger tandis que d'autres s'attaquent à la rédaction du rapport de projet. Enfin, il y a traduction de la fonction linéaire et il subsiste le même problème que pour le réglage de l'intensité (<33% et >66% impossibles).

Onzième séance *(26/05/2011)*

Le programme est terminé. La grande majorité de l'équipe s'atèle aux problèmes persistant au niveau de l'intensité et de la fonction du temps, cependant aucune solution n'est trouvée. Les autres élèves avancent dans la réalisation du rapport.

Douzième séance *(09/06/2011)*

P*as de véritable séance car indisponibilité du responsable projet mais réunion des élèves à la bibliothèque afin d'avancer dans la rédaction du projet. Il y a aussi recherche de solution concernant le problème de synchronisation de montée et descente; mais en vain.*

Séance intermédiaire *(14/06/2011)*

Nous sommes retournés au laboratoire durant la semaine pour tester les modifications que nous avons apporté au programme depuis la dernière séance. Ces modifications devaient permettre au programme de mieux fonctionner mais la connexion avec la lampe est impossible même lors de manipulations élémentaires. La situation est pire qu'auparavant. Il y a définitivement un problème avec la liaison lampe-ordinateur.

Treizième séance *(16/06/2011)*

Dernière séance avant de rendre le rapport et le PowerPoint concernant le projet (le lendemain). Les détails sont peaufinés et une dernière relecture a lieu afin de supprimer d'éventuels fautes d'orthographe. L'algorithme est complet et le programme aussi, bien que le problème concernant la lampe n'est malheureusement pas résolu.

2.3. Problèmes matériels et logistique rencontrés

Dans le cadre de ce projet, il faut reconnaître que nous n'avons eu besoin que de peu de matériel et d'aucune logistique particulière : blouses et lunettes de protection, indispensable dans les locaux de chimie et pendant l'utilisation de la lampe, et d'un ordinateur sur lequel était installé Visual Basic. Néanmoins, il est vrai que nous avons été confrontés à quelques contretemps.

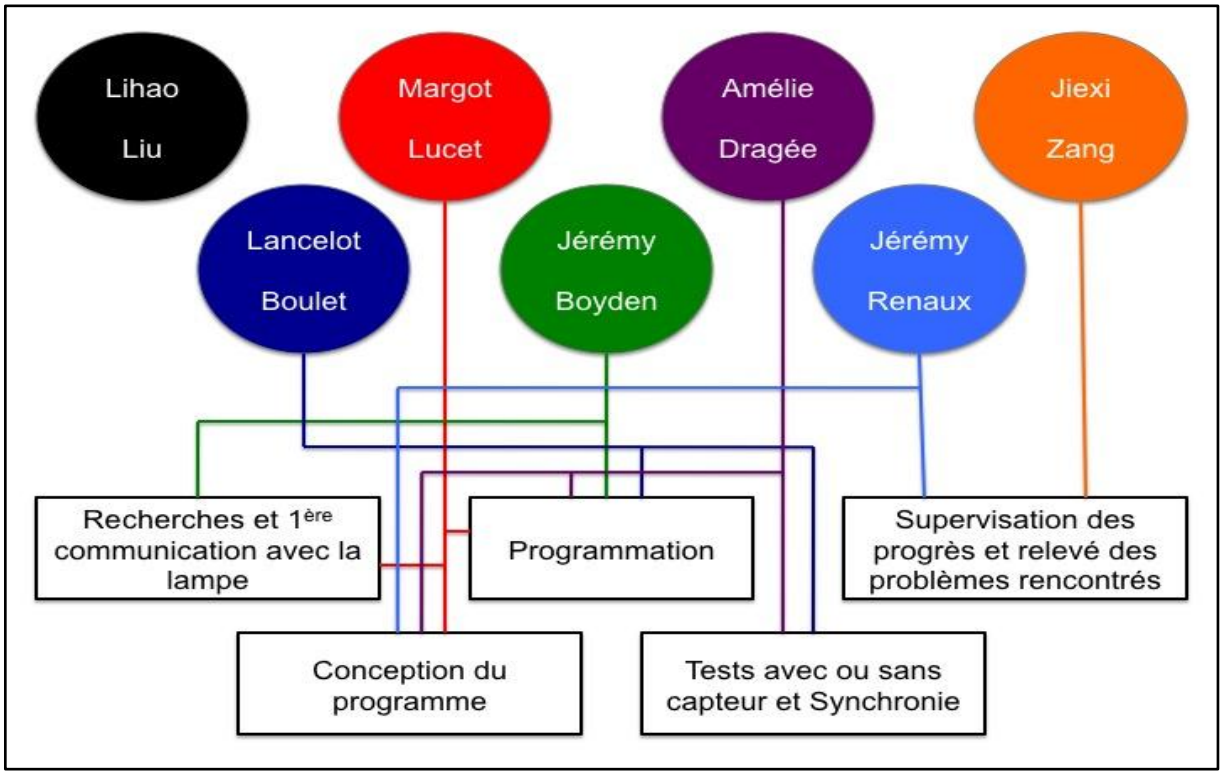
Tout d'abord, en semaine 3, nous avons essayé de communiquer avec la lampe grâce au logiciel Matlab. Cela fonctionnait mais il s'est avéré très peu pratique pour plusieurs raisons. La programmation y est tout d'abord relativement complexe et permettait principalement d'effectuer des manipulations en direct. Ensuite, les possibilités de faire un logiciel interactif et facile d'utilisation étaient limitées, l'interface est plutôt brute et il faut déjà des notions avancées en informatique pour l'utiliser (ce qui n'était pas forcément le cas des potentiels utilisateurs de notre programme).

C'est pourquoi, Mr Lebaudy nous a conseillé l'utilisation de Visual Basic, logiciel que nous avons commencé à utiliser en semaine 4 sur l'ordinateur prêté par notre encadrant.

L'un des gros points noirs du projet résidait dans le fait que Visual Basic n'était installé que sur un unique ordinateur ; il aurait été préférable de pouvoir programmer sur plusieurs ordinateurs simultanément dans un souci d'efficacité.

Enfin, l'installation du capteur optique pour modéliser l'intensité de la lampe au cours du temps sous Synchronie était un peu archaïque puisque l'on utilisait du papier aluminium pour cacher le capteur de la lumière et qu'on le surélevait comme l'on pouvait avec ce que l'on avait. Il apparaissait donc beaucoup de bruit à l'écran, mais cela a quand même permis de tirer des observations intéressantes.

Ainsi, malgré une ou deux semaines perdues, les problèmes d'ordre logistiques et matériels n'auront pas été majeurs lors de ce projet.



Organigramme des tâches réalisées et des étudiants concernés

3. TRAVAIL REALISE ET RESULTATS

3.1. Conception du programme de pilotage de la lampe

3.1.1. Objectif du programme

Comme nous l'avons dit précédemment, le pilotage de la lampe était possible en utilisant directement les commandes manuelles. Il y a plusieurs fonctions : 1 bouton permet d'allumer et d'éteindre la lampe, un autre permet l'ouverture et la fermeture du volet, deux boutons permettent d'augmenter et de diminuer l'intensité de la lampe et enfin, un bouton permet de programmer une durée automatique de fonctionnement de la lampe. Cependant, ces fonctions étaient loin d'être suffisantes pour que l'équipe de notre encadrant puisse mener ses expériences de polymérisation.

Tout d'abord, grâce aux boutons, on peut seulement baisser ou augmenter d'une petite valeur. Nous avons obtenu cette valeur en testant le nombre d'appuis nécessaires pour augmenter de 1% pourcent. Finalement, nous nous sommes rendu compte qu'il en fallait 7 pour augmenter de 1% et 8 pour diminuer de 1%, ces valeurs connaissent cependant quelques petites variations selon l'intensité, cela a d'ailleurs été notre premier indice sur le manque de précision de la lampe.

Ensuite, il n'y a pas du tout de gestion de l'intensité par rapport au temps, ainsi, il est très difficile de piloter la lampe manuellement pour que l'intensité varie selon une fonction. Par exemple si on veut qu'elle suive une fonction linéaire, il faudrait que l'utilisateur appuie à intervalles réguliers sur le bouton pendant toute la durée de l'expérience, ce qui est, évidemment un inconvénient important. Il suffit de s'imaginer alors de la difficulté de suivre une fonction telle que la sigmoïde que notre encadrant voulait utiliser.

La lampe pouvait aussi être commandée par ordinateur grâce à un port série RS-232. Depuis l'ordinateur, nous pouvions utiliser divers commandes (voir annexe documentation technique).

A l'aide de ces instructions, nous avons devions créer un programme ayant les fonctionnalités suivantes :

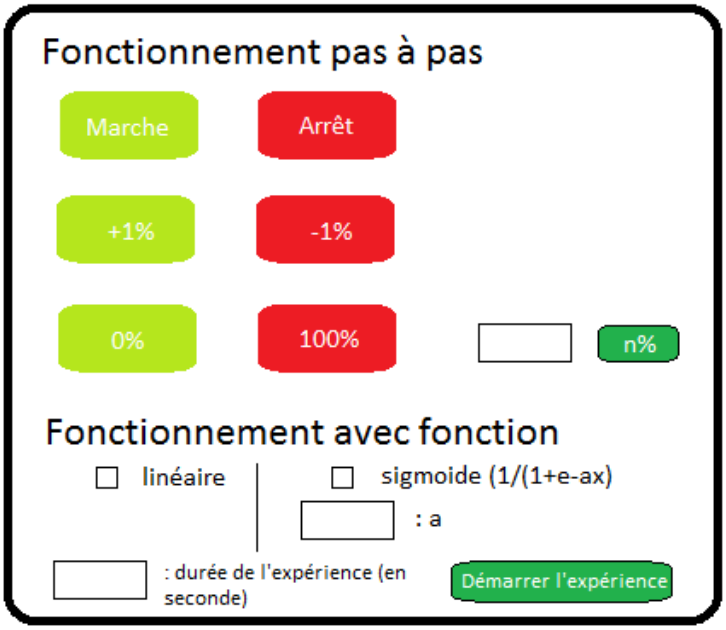
Logique métier

- Allumer et éteindre la lampe
- Mettre la lampe à 0%
- Mettre la lampe à 100%
- Mettre la lampe à n%
- Augmenter et baisser de 1%
- Faire suivre à la lampe une fonction linéaire
- Faire suivre à la lampe une fonction sigmoïde

Interface Homme-Machine

- Interface facile d'utilisation, claire et concise
- Bouton marche et arrêt
- Bouton 0 et 100%
- Bouton +1% et -1%
- Champ permettant à l'utilisateur de choisir une valeur de l'intensité à atteindre
- Champ permettant à l'utilisateur de choisir la durée de l'expérience
- Champ permettant à l'utilisateur de choisir le paramètre de la fonction sigmoïde
- Bouton permettant de démarrer l'expérience

Voilà un schéma de l'aspect que nous voulions donner à notre programme :



Le langage que nous avons choisi, le Visual Basic, est un bon compromis nous permettant à la fois de donner des instructions à la lampe mais aussi de facilement créer une interface comme celle que nous imaginions.

3.1.2. Algorithme

Après avoir testé les différentes fonctionnalités de la lampe avec le logiciel MatLab (voir partie B), il a fallu écrire l'algorithme des différentes fonctions dont nous aurions besoin pour commander notre lampe.

Les instructions marche, arrêt, mettre à 0 et à 100 % sont directement programmées dans la lampe, les boutons associés sont donc directs à programmer : il suffit de donner l'instruction à la lampe.

Pour la fonction +1% et -1%, il suffit de répéter respectivement 7 ou 8 fois l'instruction augmenter de 1/7 % ou baisser de 1/8 %. Il faut donc faire une boucle déterministe pour répéter l'instruction

```
Fonction +1%
Pour i ← 1 à 7 faire
        L3      // Instruction +1/7%
Finpour
```

```
Fonction -1%
Pour i ← 1 à 8 faire
        L4      // Instruction -1/8%
Finpour
```

La fonction aller à n% n'est pas tellement différente, il suffit de récupérer le n pour retrouver le nombre d'itérations à effectuer. Il ne faut pas non plus oublier de remettre la lampe à 0% pour pouvoir atteindre le bon pourcentage.

```
Fonction n%
lire(n)
L0      // Instruction remettre à 0
Pour i ← 1 à n*7
        L3
Finpour
```

Si les fonctions telles que marche/arrêt ou augmenter de n% étaient facile à mettre en algorithme, nous avons mis plus de temps à trouver comment gérer les fonctions par rapport au temps. En effet, il fallait que trouver un moyen que l'intensité suive une fonction du temps alors que nous n'avons à notre disposition que des instructions permettant d'augmenter ou de diminuer l'intensité de 1/7 %. Nous avons finalement décidé de calculer toutes les secondes le nombre d'itérations pour atteindre la valeur de la fonction. Il fallait aussi veiller à ce que le nombre d'itérations total atteigne environ 700, pour que l'intensité varie entre 0 et 100% tout en suivant la fonction. Il fallait donc calculer un coefficient de normalisation. Pour cela, il fallait diviser 100 par la valeur maximale. Cette valeur est en fait la valeur à tmax vu que toutes les fonctions que nous utilisions étaient croissantes. Ainsi, en multipliant la fonction par ce coefficient, les valeurs sont comprises entre 0 et 100, ce qui nous permet de calculer le nombre d'itérations à chaque seconde. Voici l'algorithme auquel nous avons abouti (en utilisant f(x,a,b,c) représentant n'importe quelle fonction croissante de paramètre a,b,c) :

```
Fonction f(x)
lire(a)
lire(c)
lire(tempsExp)
max ← f(tempsExp,a,b,c)
coeffNorma ← 100/max
total ← 0


Pour i ← 1 à tempsExp
```

```
            faire
            valeur ← f(i,a,b,c)*coeffNorma - total
            total ←f(i,a,b,c)*coeffNorma
            nombreIteration ←Trunc(7*valeur);  // Fonction troncature
            Pour j ←1 à nombreIteration
                  faire
                  L3
            Finpour
Finpour
```

Il suffit ensuite d'adapter cet algorithme aux fonctions que nous voulons utiliser comme la fonction linéaire ou sigmoïde (voir algorithme des fonctions en annexe).

3.1.3. Codage en Visual basic 6

Une fois les différents algorithmes définis, nous sommes passé à la programmation de l'application. Nous avons choisis le Visual basic 6 car notre responsable de projet avait déjà travaillé sur la programmation d'une application similaire dans ce langage.

Dans la documentation de la lampe, les instructions pour réaliser différentes actions sont fournies (voir annexe : Documentation technique), nous devions donc apprendre comment programmer notre application pour pouvoir envoyer des instructions à la lampe. A partir de livre prêté par notre professeur Margot et Amélie se sont donc plongées dans l'apprentissage de ce langage.

Nous avons ensuite pu coder les fonctions marche et arrêt de la lampe. Ces fonctions basiques nécessitant avant toute chose la programmation de la communication avec le port COM 1 sur lequel est branché la lampe. Il fallait réaliser les réglages suivants afin de pouvoir communiquer correctement avec la lampe:

```
With MSComm1
        CommPort = 1
        .Handshaking = 2
        RThreshold = 1
        RTSEnable = True
        Settings = "9600,n,8,1"
        SThreshold = 1
        PortOpen = True
        Output = "CTN1" & Chr$(13)
        .Output = "CTN1" & Chr$(13)
        .Output = "INTQ" & Chr$(13)
End With
```

Après avoir effectué ces réglages, nous avons pu programmer la réactivité des boutons pour que lorsque l'on clique dessus il envoi la bonne instruction a la lampe. Nous

avons ensuite ajouté deux autres boutons pour mettre l'intensité à 0% et à 100%. Ces quatre boutons constituent le mode basique de fonctionnement de l'application.

Nous avons ensuite codé 4 autres boutons : marche, arrêt, +1% et -1% qui constitue le mode de fonctionnement pas à pas.

Le troisième mode de fonctionnement que nous avons codé est le mode fonction. Le but étant que l'intensité suive une fonction sigmoïde d'équation du type :

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

Comme nous l'avons dit précédemment, l'algorithme permettait de calculer toutes les secondes la valeur de la fonction pour calculer ensuite le nombre d'instructions nécessaires pour atteindre l'intensité correspondante.

Pour effectuer l'instruction toutes les secondes, il fallait faire une boucle qui tournait tant que le temps depuis le début n'avait atteint une seconde de plus. Cette méthode est codée en Visual à l'aide d'un « timer ». Ce timer est lancé par l'utilisateur lorsqu'il appuie sur le bouton « démarrer » et les instructions contentant dans la programmation de la fonction timer sont exécutées toute les secondes (cette valeur pouvant être modifiée par le programmeur comme il le souhaite mais pas par l'utilisateur).

Bien sûr quelque soit la manière choisie il nous a fallut pauser des conditions d'arrêt car l'intensité et la réaction sont régies par certaines contraintes. Ces conditions étaient donc : si l'intensité atteint 100% le programme s'arrête et si le temps écoulé depuis le début de la réaction atteint la durée de la réaction là aussi le programme s'arrête.

Au final nous avons obtenu l'interface utilisateur suivante :

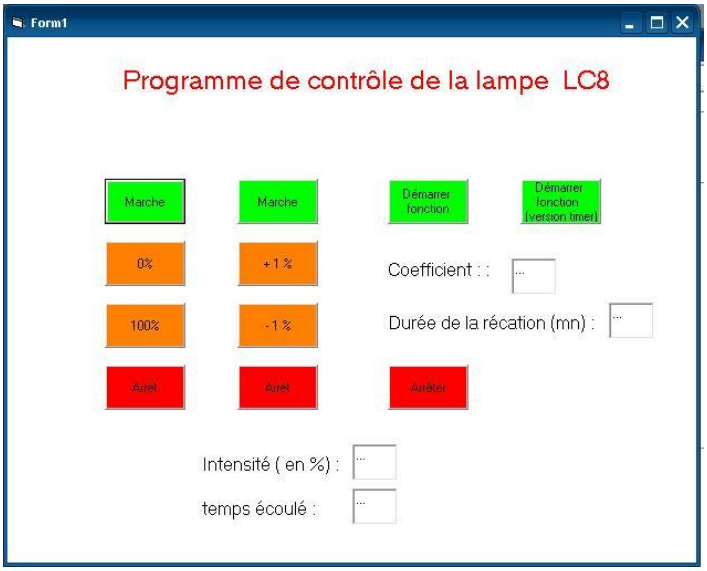


Figure 1 : Interface utilisateur du programme

A chaque fois que nous codions une fonctionnalité de notre programme, nous la testions. C'est pourquoi nous allons maintenant passer à la seconde partie et vous parler des différents expériences et tests que nous avons menés et de leur résultats.

3.2. Résultats

3.2.1. Expériences avec Matlab

Les expériences avec le logiciel Matlab sont les premières que nous avons effectuées. Elles nous ont permis de comprendre la communication entre l'ordinateur et la lampe et de tester la plupart des instructions. Ces tests étaient plutôt faciles à mener étant donné que grâce à MatLab, nous pouvions communiquer directement avec la lampe en la configurant au début.

Dès les premières séances, nous avons donc pu communiquer avec la lampe grâce à des instructions simple. Nous sommes parvenu à éteindre, allumer, mettre à 0% et à 100% la lampe, ainsi qu'augmenter ou baisser son intensité, tout ça à distance.

Ces premières expériences ont été très utiles pour comprendre le fonctionnement de la lampe et de ses instructions.

3.2.2. Tests du programme

Lors du développement de l'algorithme en Visual Basic, nous avons été confrontés à de nombreux problèmes. En effet, lorsque nous lancions le programme certaines fonctionnalités faisaient planter le logiciel et nous n'arrivions pas à comprendre pourquoi. Par exemple, pour le fonctionnement grâce à une fonction linéaire, le programme ne répondait plus au bout d'une ou deux minutes. Nous avons donc parcouru entièrement notre code mais ne réussissant pas à trouver les failles nous avons décidé d'effectuer des tests sur notre algorithme en Pascal. Le Pascal est un langage que certain d'entre nous apprenaient en cours donc que nous maîtrisions mieux et c'était aussi un langage plus adapté pour faire rapidement des tests. De cette façon, nous pouvions voir si, sans la lampe, notre programme effectuait les bonnes instructions.

Voici donc le code (partiel) de notre test :

```
begin

a:=2;                                          // valeurs test de a et de b
b:=0;
tempsExp:=60;                                  // calcul du maximum
max:=a*tempsExp*tempsExp+b*tempsExp;
total:=0;
```

```
                                                    // calcul du coefficient
                                                    de normalisation
coeffNorma:=100/max;

                                                    // affichage du coeff

writeln ('CoeffNorma = ',CoeffNorma);
for i:=1 to tempsExp
        do begin                                    // calcul de la valeur
                                                    normalisée
        valeur:= (a*i*i+b*i)*coeffNorma-total;

                                                    // calcul du nombre
        total:=(a*i*i+b*i )*coeffNorma;             d'itérations nécessaires
        nombreIteration:=Trunc(7*valeur);           // affichage du nombre
                                                    d'itération et de valeurs

        writeln('valeur:', valeur);
        writeln('nombre     des     iterations',
nombreIteration);
        end;

end.
```

Ce code a été obtenu après plusieurs essais qui nous ont permis de corriger certaines erreurs que nous avions commises. Voici maintenant le résultat de notre test (c'est ici la simulation d'une fonction linéaire, le temps n'est pas géré mais simulé de façon faire comme si la fonction était calculée une fois toutes les dix secondes sur 60 secondes) :

CoeffNorma = 5.55555555555556E-001

valeur: 1.66666666666667E+001

nombre des iterations116

valeur: 1.66666666666667E+001

nombre des iterations116

valeur: 1.66666666666667E+001

nombre des iterations116

valeur: 1.66666666666667E+001

nombre des iterations116

valeur: 1.66666666666667E+001

nombre des iterations116

valeur: 1.66666666666667E+001

nombre d'itérations 116

valeur: 1.66666666666667E+001

nombre d'itérations 116

C'est une fonction linéaire, il est donc logique que la valeur dont la fonction augmente à chaque période de temps soit constante, ainsi que les itérations. Si on additionne toutes les valeurs on trouve un nombre total d'itérations de 696 (ce n'est pas exactement 700, mais cette approximation est amplement suffisante au vu de la précision nécessaire à l'usage de notre programme)

Nous avons ensuite effectué une simulation beaucoup plus précise sur un polynôme du second degré. Cette fois, nous avons simulé un calcul toutes les secondes pendant 1 minute, voici le résultat de la modélisation de notre fonction.

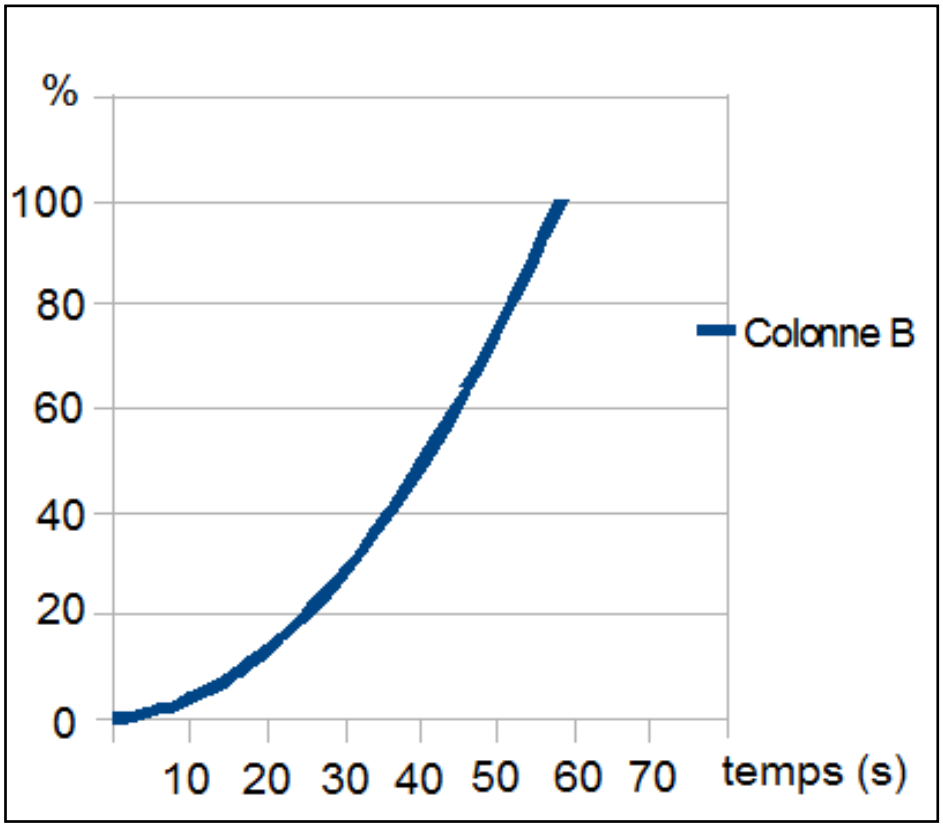


Figure 2 : résultat de la modélisation en pascal

On voit ici que le résultat est effectivement une parabole variant de 0 à 100 sur 1 minute. Une fois ces tests effectués, nous avons recommencé les tests en Visual Basic.

Une fois que nous avions codé une fonction nous la testions sur la lampe. Pour cela il suffisait de réaliser le montage Lampe-ordinateur (voir annexe : Montage) et d'exécuter le programme.

Le test du mode basique a été un succès. La lampe a exécuté les instructions demandées dès qu'on appuyait sur le bouton pour les quatre fonctions : marche, arrêt, 0% et 100%.

Le test du mode pas à pas était un succès de première abord : la mise en marche et l'arrêt fonctionnait car il s'agissait des mêmes que pour le mode basique, et les fonctions -1% et +1% permettais bien de diminuer ou d'augmenter l'intensité de 1%. Cependant lorsque nous avons voulu passer de 0 à 100 % (ou inversement) avec ces deux boutons, nous avons remarqué que le programme plantait et ne répondait plus vers les 60% (30% dans l'autre sens). Nous n'avons pas réussi à résoudre ce problème, en effet ce bouton était codé de manière très simple à l'aide d'une boucle déterminisme (« pour ») s'exécutant 7 fois et qui donc n'expliquait pas le plantage du programme.

Malgré ce problème décelé vers la mi-projet nous avons continué quand même la programmation car il nous restait le mode fonction (le plus important) à coder. Nous l'avons tout d'abord testé avec une fonction linéaire puis une fonction parabole, mais dans les deux cas le programme plantait arrivé vers 60%, de la même manière que dans le mode pas à pas. De plus, nous avions remarqué, en chronométrant le temps de la réaction déclenché par le programme, que l'exécution était trop rapide et donc la durée ne correspondait pas à celle entrée par l'utilisateur.

Ce problème étant similaire à celui rencontré avec les fonctions -1% et +1% nous étions là aussi incapables de le résoudre.

Nous avons décidé de programmer tout de même le mode fonction pour la fonction sigmoïde car c'était la finalité de notre projet. Nous l'avons fait selon deux méthodes différentes (voir partie : codage en Visual basic 6.0). La méthode avec le Timer ayant pour but de résoudre le problème de non respect de la durée rentrée par l'utilisateur, la boucle « pour » de la première méthode mettant moins d'une seconde à s'exécuter.

Cependant lorsque nous avons voulu tester ces nouvelles fonctions, peu de temps avant la fin de notre projet, nous avons été confrontés à un problème de communication avec la lampe. En effet nous recevions les informations sur l'intensité envoyée par la lampe, mais nous ne pouvions pas la piloter car elle ne réagissait pas aux instructions envoyées par le programme. Nous avons donc tenté de la piloter de manière instantanée à l'aide du logiciel MatLab et nous nous somme alors rendu compte lorsque l'on demandait à la lampe de passer en mode pilotage par ordinateur, au lieu de renvoyer « CTN1OK » elle renvoyait « ERR ». Nous avons donc compris qu'il y avait un réel problème pour établir la communication avec la lampe mais nous n'avons pas réussi à le résoudre dans le peu de temps qu'il nous restait. De ce fait nous n'avons pas pu tester les deux algorithmes de la fonction sigmoïde.

3.2.3. Résultats

Au terme de notre projet, nous sommes arrivés à un résultat satisfaisant mais où tout ne fonctionnait pas comme nous l'aurions souhaité. En effet, nous sommes parvenus à coder à l'aide du Visual Basics 6.0 un programme qui correspond bien à nos attentes et, qui, en termes de compilation ne pose aucun problème mais qui n'offre pas toutes les possibilités souhaitées à causes des plantages dont nous n'avons pas trouvé l'origine et de la panne de la lampe lors des tests du programme final.

En réalité, comme il a été dit précédemment, nous avons été confrontés à un certain nombre de problèmes tout au long du projet. Chaque fois, c'est par de multiples réflexions et par de multiples variations de notre programme informatique que nous les surmontions. Cependant l'un d'eux a persisté et malgré de nombreuses efforts. Nous avons effectué des recherches personnelles (internet, documentation, livret sur Visual Basics, consultation du responsable de projet) et avons demandé conseil à notre professeur d'informatique mais nous ne sommes pas parvenus à résoudre ce problème de plantage et d'absence de réponse du programme. Nous avons même été en possession d'un document où apparaissait le programme fonctionnel d'une personne ayant travaillé sur le même projet dans un IUT et après comparaison avec notre propre programme, il n'apparaissait pas de divergence qui aurait expliqué ce problème.

Par conséquent, à l'heure actuelle, notre programme informatique est compatible avec la lampe Hamamatsu et il nous permet une certaine utilisation de celle-ci. Cependant toutes les fonctionnalités souhaitées au début du projet ne sont pas disponibles. Le programme permet bien de régler l'intensité de la lampe UV à 0% et à 100% ainsi que de la modifier en ajoutant ou en soustrayant un pourcentage. Cependant, il subsiste un gros problème au niveau de la synchronisation de la montée et descente. La fonction sigmoïde est elle aussi présente mais du fait de la panne de la lampe nous n'avons pas pu tester cette fonctionnalité.

C'est pourquoi, au terme de treize semaines de recherches, de travaux, de programmation et de problèmes, nous sommes arrivés à construire un programme qui permet de dialoguer avec la lampe bien qu'il ne puisse pas fonctionner de manière à suivre une fonction.

4. CONCLUSIONS ET PERSPECTIVES

La mission qui nous avait été confiée était de développer un programme permettant de contrôler une lampe UV, et plus particulièrement que son intensité suive une fonction précise au cours du temps. Tout cela afin de solidifier un polymère transparent en l'éclairant du rayonnement de la lampe.

D'apprendre que le résultat final du projet allait servir plus tard à notre professeur dans le cadre de ces réactions de polymérisation (ou aux élèves du département CFI) nous a énormément motivé. Il est donc clair que le fait que la lampe ne répondait finalement plus à aucune de nos commandes a été vécu comme une déception pour tous.

Cette expérience a cependant été très enrichissante. Le projet nous a ainsi permis d'utiliser pour la première fois le logiciel Matlab ou de l'utiliser concrètement pour les futurs ASI et, pour tout le monde, d'apprendre à écrire et à se servir de Visual Basic. Le projet physique ayant pour principal objectif de nous faire travailler à plusieurs, il nous a permis de développer notre expérience de groupe, tant sur le plan individuel que sur le plan collectif.

Nous avons réussi à écrire un programme dont les fonctions basiques (marche, arrêt, mise à 100%...) fonctionnent même si, lors des dernières semaines, la lampe ne communiquait plus avec l'ordinateur. Cependant, nous ne sommes pas parvenus à programmer un mode fonction tournant correctement. Finalement, comme l'algorithme fonctionnait sans la lampe, nous pensons que nos problèmes venaient de la communication avec la lampe.

Nous aimerions terminer ce rapport en expliquant ce que l'on aurait pu apporter en plus au programme si la communication avec la lampe n'avait pas été si problématique :

Dans un premier temps, nous avions pensé à intégrer tout un tas de fonctions différentes organisées dans un menu déroulant. Si l'une des fonctions avait été sélectionnée, les paramètres à entrer seraient apparus automatiquement sur la page principale. Ensuite, l'amélioration de l'interface graphique, relativement sommaire, est un point non négligeable mais sur lequel nous ne nous sommes pas attardés, la programmation de la lampe en elle-même étant bien plus importante.

Enfin, nous pourrions intégrer de nouvelles fonctionnalités pour personnaliser le programme, tels que la sélection des molécules à polymériser, pointant sur une fonction où les paramètres sont entrés par défaut, ou l'intégration d'onglets de favoris.

5. BIBLIOGRAPHIE

Livres et documentation :

- ✓ *Microsoft Visual Basics 6*
  Éditeur : Micro application, PC poche
  Description :
  -Complet (toutes les informations et les nouveautés de Visual Basics 6)
  -Pratique (les contrôles, les objets, les événements, les propriétés)
  -Efficace (des réponses concrètes et immédiates)
  Auteur : M. Kirstein
  Date de parution : 1998

- ✓ *Lightningcure Series LC8*
  Editeur : Hamamatsu
  Description : mode d'emploi de la lampe UV
  Date de parution : février 2006

- ✓ *Spot light Source Instruction Manual*
  Editeur : Hamamatsu
  Description : explique le fonctionnement de la lampe UV, le mode d'emploi.
  Date de parution : novembre 2005

Sites internet :

- ✓ **Christophe Darmangeat**, *Explication du langage informatique Visual basics* : *http://www.pise.info/vb/*

- ✓ **Philipe Baquer**, *Apport d'informations sur l'utilisation de Visual Basics*, *http://bbil.developpez.com/tutoriel/vb/debuter-vb6/*

- ✓ **Philippe Galez**, *Explication et aide sur le fonctionnement de la lampe UV* ; *http://www.iut-acy.univ-savoie.fr/fileadmin/DUT/MPH/fichiers/semestre3/techniques-spectroscopiques/Spectrophotometrie-UV-visible.pdf*

- ✓ **UV industrie**, *Fonctionnement de la lampe Hamamatsu*, *http://www.uv-industrie.com/*

6. ANNEXES (NON OBLIGATOIRE – EXEMPLES CI-DESSOUS)

6.1. Documentation technique

Table 2 : List of commands

Command	Description	Character string returned from light source side	Handled by 1STEP	Handled by 7STEP	Remarks
CNT0 CNT1	Control mode setting 0: Front panel control 1: Command control	CNT0OK CNT1OK	Yes	Yes	When using the front panel control, ERR is output for all commands other than this command. The front panel control is used after the power is turned on.
CNTQ	Inquiry of control mode	CNT*	Yes	Yes	* Control mode (0: from front, 1: from communication port)
VER	Version information	VER**.**.**	Yes	Yes	
L1	Lamp ON	L1OK L1NGxx	Yes	Yes	NG trigger being applied (01) Shutter is also closed at the same time if it is open. If auto cure operation is at work, it is terminated (closed).
L0	Lamp OFF	L0OK L0NGxx	Yes	Yes	NG trigger being applied (01)
S1	Shutter OPEN	S1OK S1NGxx	Yes	Yes	NG memory cure operation at work (02)
S0	Shutter CLOSE	S0OK --	Yes	Yes	If memory cure operation is at work, it is terminated (closed). If auto cure operation is at work, it is terminated (closed).
AT***.*	Shutter auto time setting ***.*(000.0 – 999.9 s)	ATOK ATNG03	Yes	No	NG auto function being executed (03)
ATQ	Shutter auto time check	AT***.*	Yes	No	
ATS	Shutter auto operation started	ATSOK ATSNGxx	Yes	No	NG auto function being executed (03)
ATT	Shutter auto operation remaining time check	ATT***.*	Yes	No	ATTOK00000.0 other than when being executed
INT*	Diaphragm control : * is 0 to 4 numbers, meaning 0: Stop, 1: UP, 2: DOWN 3: UP (1point), 4: DOWN (1point)	INT*OK INT*NG	Yes	Yes	NG memory cure operation at work (02)
INTQ	Output intensity check	INT***	Yes	Yes	*** Output intensity (%)
CURE α β γ : nnn,mmm.m,... or CURE α β γ : nn.n,mmm.m,...	The memory cure program of the program No. α is set by the unit of % if γ=1 when using the shutter mode β and by the unit of W if γ=2 when using the shutter mode β. α: Program number (1 – 9) β: Shutter mode (1 or 2) γ: Unit of output intensity being set (1: %, 2: W) nnn: Output intensity (%) n.nn: Output intensity (W) mmm.m: Time(s) Comma division is used to link 7 steps.	CURE α β γ OK CURE α β γ NG	No	Yes	NG memory cure operation at work (02)
CUREQ α	Program number α memory cure program check	CURE α β γ : nnn,mmm.m,... or CURE α β γ : n.nn,mmm.m,...	No	Yes	α: Program number (1 – 9), β: Shutter mode (1 or 2) γ: Unit of output intensity being set (1: %, 2: W) nnn: Output intensity (%), n.nn: Output intensity (W), mmm.m: Time(s) Comma division is used to link 7 steps.
START*	Program ** memory cure (setting in %) started	START*OK START*NGxx	No	Yes	NG memory cure operation at work (02)
STPQ	Memory cure progress state	STP*	No		* Current step (1 – 7, 0 when unexecuted)
STOP	Memory cure forced to end	STOPOK	No	Yes	Shutter is closed and diaphragm is also stopped.
INP	Input pin data transmission	INP**	Yes	Yes	** represent hexadecimal characters and each bit corresponds with the status described below. Bit0: 1 Lamp ON, 0 Lamp OFF Bit1: 1 Lamp stable, 0 Lamp unstable Bit2: 1 Shutter open, 0 Shutter closed Bit3: 1 Diaphragm opening Min, 0 Otherwise Bit4: 1 Diaphragm opening Max, 0 Otherwise Bit5: Error Transmitted as soon as the signal changes
LIF	Lamp operation time check	LIFOK****.*	Yes	Yes	**** *: Lamp operation time (hour), minimum resolution 1 h
LAMPCLR	Accumulated lamp operation time clear	LAMPCLROK	Yes	Yes	
PROG_1STEP	Program mode set to 1STEP (Automatic recognition of Optical Feedback Unit)	PROG_1STEPOK	Yes	Yes	
PROG_7STEP	Program mode set to 7STEP (Automatic recognition of Optical Feedback Unit)	PROG_7STEPOK	Yes	Yes	
PROG_N1STEP	% display program mode setting forced to 1STEP	PROG_N1STEPOK	Yes	Yes	
PROG_N7STEP	% display program mode setting forced to 7STEP	PROG_N7STEPOK	Yes	Yes	
PROG?	Inquiry of program mode	*STEP	Yes	Yes	* represents the current program mode 1STEP: 1 % forced display 1STEP: N1 7STEP: 7 % forced display 7STEP: N7

Explanation of NG ..... XX represent the NG content.
01: Lamp trigger being applied, 02: Memory cure being executed, 03: Shutter auto function being executed.
04: Optical feedback unit unconnected, 05: Error not reaching memory cure setting of output intensity (W)

*** and X represent 1 character of ASCII** ..... Example: when xx=01, the data being communicated is H'30H'31.

6.2. Algorithme des fonctions

Fonction lineaire (a : Entier, x : Réel)
Déclaration : resultat : Réel
Début
resultat ← a*x
retourner résultat
Fin

Fonction polynome (a,b : Entier, x : Réel)
Déclaration : resultat : Réel
Début
resultat ← a*x*x+b*x
retourner resultat
Fin

Fonction sigmoide (a : Entier, x :Réel)
Déclaration : resultat : Réel
Début
Resultat ←1/(1+exp(-a*x))
Retourner résultat
Fin

6.3. Code source

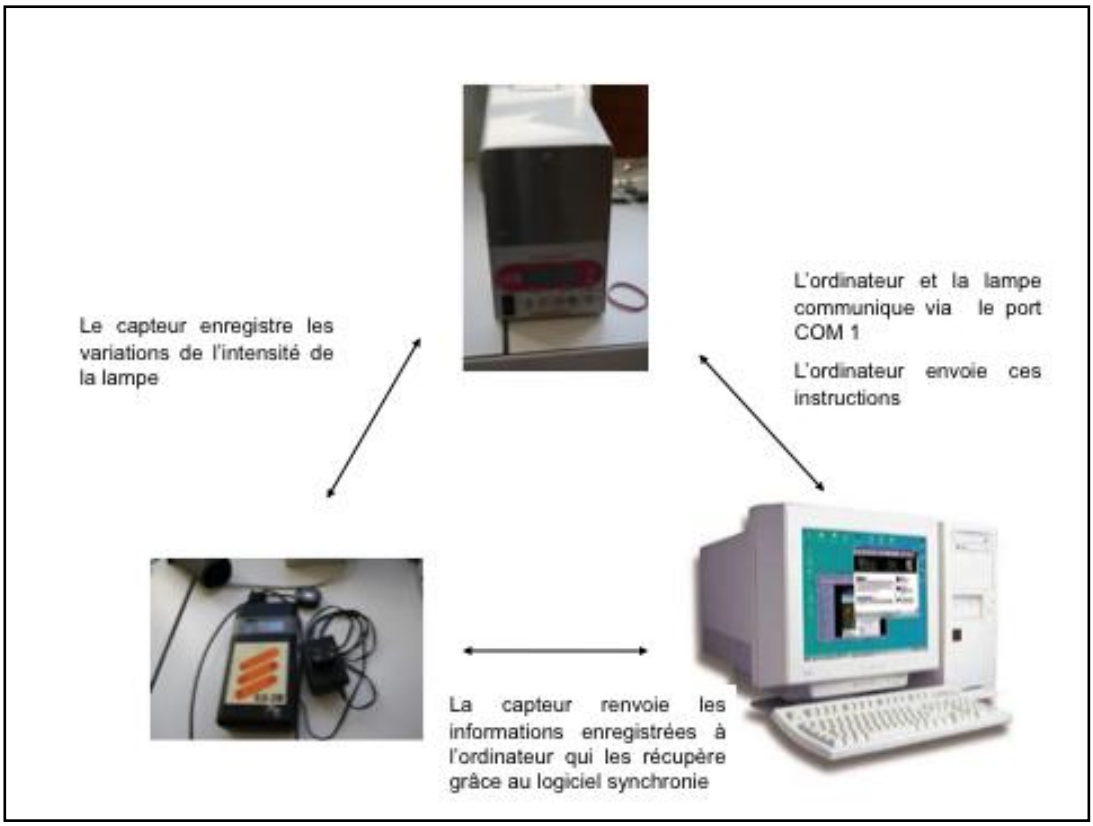
```
Option Explicit 'déclaration des variables
Dim tampon As String
Dim valint As String
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim vduree As Double
Dim coeff As Double
Dim tempsR As Double
Dim valeur As Double
Dim nombreboucle As Double
Dim tempsecoule As Double

Private Sub btn0pourcent_Click() 'fonction pour mettre l'intensité à 0%
MSComm1.Output = "INT2" & Chr$(13) 'commande pour mettre à 0%
MSComm1.Output = "INTQ" & Chr$(13) ' Demande la valeur de l'intensité
End Sub

Private Sub btn100pourcent_Click() 'fonction pour mettre l'intensité à 100%
MSComm1.Output = "INT1" & Chr$(13) 'commande pour mettre à 100%
MSComm1.Output = "INTQ" & Chr$(13) ' Demande la valeur de l'intensité
End Sub

Private Sub btnarret_Click() 'fonction bouton d'arrêt du mode basique
```

```
With MSComm1
    .Output = "S0" & Chr$(13) 'commande pour la fermeture du shutter
    .Output = "L0" & Chr$(13) 'commande pour l'extinction de la lampe
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub btnarreterfonction_Click()
With MSComm1
    .Output = "S0" & Chr$(13) 'commande pour la fermeture du shutter
    .Output = "L0" & Chr$(13) 'commande pour l'extinction de la lampe
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub btnarretpap_Click()
With MSComm1
    .Output = "S0" & Chr$(13) 'commande pour la fermeture du shutter
    .Output = "L0" & Chr$(13) 'commande pour l'extinction de la lampe
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub btndemarrerfonction_Click()
vduree = duree.Text 'on récupère la valeur de la durée rentrée par l'utilisateur


coeff = coef.Text 'on récupère la valeur du coefficient rentré par l'utilisateur
tempsR = vduree * 60 'on convertie la durée en minute en une durée en seconde
For i = 1 To tempsR 'on fait une boucle déterministe qui s'exécutera autant de fois qu'il y a
de seconde dans la durée
   MSComm1.Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
   If (afficherintesite < 100) Then 'on pose la condition que l'intensité soit inférieur à 100%
      valeur = 100 * (1 - Exp(-i / coeff)) ' on calcul la valeur de la fonction à l'instant i
      nombreboucle = valeur - Val(afficherintensite.Text) ' on en déduit le nombre d'itérations
      For j = 1 To nombreboucle ' on exécute les itérations déterminées précédemment
         For k = 0 To 6 ' on réalise 7 fois l'instruction pour augmenter l'intensité
            MSComm1.Output = "INT3" & Chr$(13) 'commande pour augmenter l'intensité
d'1/7%
         Next k
      Next j
      temps.Text = i ' on affiche i correspondant au temps écoulé depuis le début de la
réaction
   End If
Next i
End Sub

Private Sub btndemarrrerfonctiontimer_Click()
MSComm1.Output = "S1" & Chr$(13) 'commande pour l'ouverture du shutter de la lampe
MSComm1.Output = "INT2" & Chr$(13) 'Commande pour mettre l'intensité à 0%
tempsecoulé = 1 ' initialisation du temps de la réaction à 1
Timer1.Enabled = True ' on déclenche le Timer
End Sub
```

```
Private Sub btnmarche_Click()
With MSComm1
    .Output = "L1" & Chr$(13) 'commande pour l'allumage de la lampe
    .Output = "S1" & Chr$(13) 'commande pour l'ouverture du shutter de la lampe
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub btnmarchepap_Click()
With MSComm1
    .Output = "L1" & Chr$(13) 'commande pour l'allumage de la lampe
    .Output = "S1" & Chr$(13) 'commande pour l'ouverture du shutter de la lampe
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub btnmoins1_Click()
For i = 0 To 6
MSComm1.Output = "INT4" & Chr$(13) 'commande pour la diminution de l'intensité de la
lampe de 1/7%
Next i
MSComm1.Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End Sub



Private Sub btnplus1_Click()
For i = 0 To 6
MSComm1.Output = "INT3" & Chr$(13) 'commande pour l'augmentation de l'intensité de la
lampe de 1/7%
Next i
MSComm1.Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End Sub

Private Sub Form_Load() 'cette sous fonction s'exécute lorsque le programme s'ouvre
With MSComm1 'configuration du port de communication COM
    .CommPort = 1 'utilisation du port 1
    .Handshaking = 2 'choix du mode communication
    .RThreshold = 1 'permet de gérer l'évènement : réception de données
    .RTSEnable = True
    .Settings = "9600,n,8,1" 'réglages : 9600bds, pas de bit de parité, 8 bits de données et 1
bit de stop
    .SThreshold = 1 'permet de gérer l'événement : envoi de donnée
    .PortOpen = True ' on ouvre le port de communication
    .Output = "CTN1" & Chr$(13) ‘on demande a la lampe de passer en pilotage par
ordinateur
    .Output = "CTN1" & Chr$(13) ‘ on renvois la même commande car si on l’envoie qu’une
seule fois ça ne marche pas
    .Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
End With
End Sub

Private Sub Form_Unload(Cancel As Integer)
```

```
With MSComm1
    .Output = "CTN0" & Chr$(13) 'commande pour l'extinction de la lampe
    .PortOpen = False ' fermeture du port de communication
End With
End Sub

Sub affichage(tampon As String) 'procédure pour afficher l'intensité
valint = Right(tampon, 4) ' sauvegarde les 4 derniers octets de la variables 'tampon' dans la variable 'valint'
If (valint <= "101") Then ' test avec la condition valint inférieure à 101%
    afficherintensite.Text = valint 'affichage de valint dans l'élément afficherintesite
End If
End Sub

Private Sub MSComm1_OnComm() 'configuration du traitement des données reçues lorsque l'on demande la valeur de l'intensité
Select Case MSComm1.CommEvent
    Case comEvReceive
        tampon = MSComm1.Input 'sauvegarde de la donnée reçu dans la variable tampon
        Call affichage((tampon)) 'appel à la procédure afficher
End Select


End Sub

Private Sub Timer1_Timer() ' instructions à exécuter toute les seconde lorsque le timer est en route
vduree = Val(duree.Text) 'on récupère la valeur de la durée rentrée par l'utilisateur
coeff = Val(coef.Text) 'on récupère la valeur du coefficient rentré par l'utilisateur
tempsR = vduree * 60 'on convertie la durée en minute en une durée en seconde
tempsecoule = tempsecoule + 1 ' on augmente le temps écoulé de 1 seconde
valeur = 100 * (1 - Exp(-tempsecoule / coeff)) ' on calcul la valeur de la fonction à l'instant tempsecoule
nombreboucle = valeur - Val(afficherintensite.Text) ' on calcule le nombre de boucles a exécuter
For i = 1 To nombreboucle ' on exécute le nombre de boucle déterminé précédemment
    For j = 0 To 6 ' on réalise 7 fois l'instruction pour augmenter l'intensité
        MSComm1.Output = "INT3" & Chr$(13) 'commande pour augmenter l'intensité d'1/7%
    Next j
Next i
MSComm1.Output = "INTQ" & Chr$(13) 'Demande la valeur de l'intensité
If (afficherintensite = 100) Then ' on vérifie que l'intensité n'est pas égale à 100%
    Timer1.Enabled = False ' si elle est égale à 100% alors on stoppe le timer
End If
If (afficherintensite > 100) Then 'on vérifie que l'intensité n'est pas supérieur à 100%
    Timer1.Enabled = False ' si elle est supérieur à 100% alors on stoppe le timer
End If
If (tempsecoule = tempsR) Then ' on regarde si le temps écoulé a atteint la même valeur que la durée
    Timer1.Enabled = False ' si les deux durées sont égale alors on stoppe le timer
End If
temps.Text = tempsecoule 'on affiche le temps écoulé
```

6.4. Montage


Le capteur enregistre les variations de l'intensité de la lampe
L'ordinateur et la lampe communique via le port COM 1
L'ordinateur envoie ces instructions
La capteur renvoie les informations enregistrées à l'ordinateur qui les récupère grâce au logiciel synchronie