Fabien ROMEROWSKI
Papa Momar MBOUP

IMA 4

Rapport de projet

# Système d'interaction dans un musée

M. Yvan PETER

M. Thomas VANTROYS

Mai 2013

# Système d’interaction dans un musée

Fabien ROMEROWSKI

Papa Momar MBOUP

# Remerciements :

Merci à monsieur Peter de nous avoir obtenu ce projet, ainsi que de nous avoir guidés dans sa réalisation.

Merci à monsieur Vantroys pour ses conseils, pour les idées qu'il nous a données, ainsi que pour son accompagnement dans la réalisation de ce projet.

Merci à monsieur Redon, qui lui aussi nous a aidés dans la réalisation de ce projet.

# Résumé :

Notre projet porte sur l'étude du développement d'un système interactif destiné à permettre aux visiteurs du Musée des Beaux Arts de Tourcoing de répondre à un très bref sondage, tout en ayant une vue générale des réponses des utilisateurs précédents. Dans cette étude, nous utilisons différents dispositifs d'affichage, mais aussi de saisie et de transmission de données. Nous avons donc créé des programmes permettant de gérer les différents dispositifs, et ce à partir d'un Arduino UNO.

Il était intéressant de travailler sur ce projet, car cela nous a permis d'approfondir nos connaissances en programmation, ainsi que de participer à la réalisation d'une animation qui sera exploitée dans un musée.

# Sommaire :

# Introduction :

Etant en deuxième année d'Ecole d'ingénieur à Polytech'Lille, nous avons eu un projet à réaliser en binôme. Ce projet a été réalisé de février à avril 2013, à raison d'environ huit heures par semaine, hors vacances scolaires.

Notre projet consiste à participer au développement d'un système d'interactions pour un musée. Les dispositifs de saisie, d'affichage, et de transmission de données seront gérés par un Arduino UNO. Notre projet est donc, d'abord, de saisir des données puis de les afficher et de les transmettre à un site et éventuellement à des réseaux sociaux comme Facebook. Tout ceci afin de rendre la visite du Musée plus interactive, en y faisant participer les visiteurs. Nous sommes encadrés dans la réalisation de ce projet par M. Peter et M. Vantroys.

Dans une première partie nous présenterons le projet de façon plus détaillée. Deuxièmement nous parlerons de la saisie et de l'affichage des données. Une troisième partie nous permettra d'étudier la transmission des données, et enfin nous nous intéresserons à ce projet avec plus de recul.

# 1. Présentation du projet :

## 1.1. Présentation du contexte :

Le projet Interaction Musée implique la société Anaxa-vida, le musée Muba de Tourcoing et le Laboratoire d'Informatique Fondamentale de Lille (Université Lille 1). Il consiste à combiner une solution d'analyse vidéo permettant une analyse des flux de visiteurs et des mécanismes d'interaction visant à rendre actifs les visiteurs. Ceci permettra de contribuer à la médiation culturelle en offrant la possibilité de comprendre et d'influencer les comportements de visite.

Le projet Interaction Musée a pour objectif de rendre le visiteur actif par l'expression du ressenti face aux œuvres qui lui sont proposées et les relations établies entre elles par leur mise en place.

Cette interaction s'adresse plutôt à des visiteurs individuels en leur offrant la possibilité d'exprimer leur compréhension de la mise en relation des œuvres ou du ressenti et de visualiser les choix antérieurs (telle ou telle relation ou sentiment est plus exprimé qu'un autre…). Cette expression pourra également être rendue visible à travers le compte Facebook du musée.


**Figure 1 : Interaction autour de la mise en relation de deux œuvres**

L'objectif de ce dispositif est de proposer au visiteur plusieurs pistes concernant la mise en relation de plusieurs œuvres, et de lui offrir la possibilité de confronter son point de vue avec celui des visiteurs précédents via un retour visuel sur l'ensemble des choix.

## 1.2. Système global :

Pour la réalisation de ce projet, nous utiliserons un microprocesseur chargé du traitement des réponses, de leur affichage, et de leur transfert. Nous utiliserons donc aussi un dispositif d'affichage, et un système de saisie des réponses. Afin de transférer les données

vers notre serveur, nous utiliserons le module WiFly Shield, ainsi qu'une borne Wifi. Le système sera alors composé de :

- 1 ArduinoUno
- 1 module WiFly Shield
- 1 borne Wifi
- 1 Tapis de danse (saisie des réponses)
- 1 Ecran LCD
- 1 Matrice 8x8

Nous utiliserons aussi d'autres composants, comme des fils pour réaliser les connexions, et des boutons poussoirs, pour faire des tests et simuler les entrées.

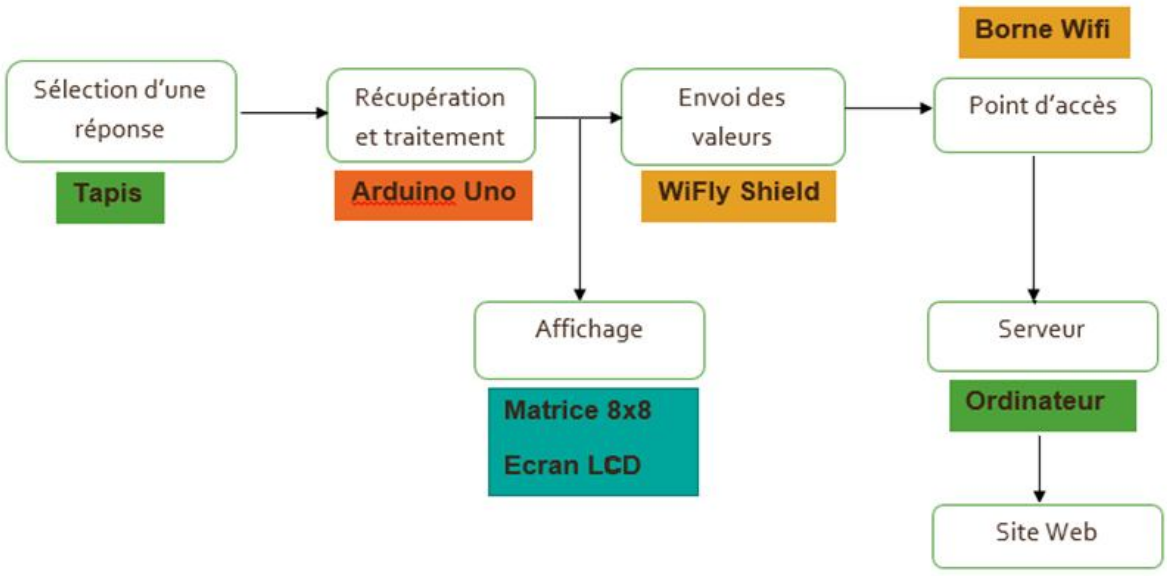


**Figure 2 : Système global**

### 1.3. Organisation du groupe :

Pour la réalisation de ce projet, nous nous sommes par tagé le travail en deux : Fabien a travaillé sur la saisie et l'affichage des données, tandis que Papa s'est chargé de la transmission des données.

# 2. Saisie et affichage des réponses :

Cette partie du travail consistait à programmer l'Arduino de façon à avoir en entrées les différents choix de réponse autorisés, et en sortie un affichage de ces réponses. Etant donné le matériel disponible à Polytech, nous avons choisi d'utiliser en sortie de l'Arduino une matrice de LED et un écran LCD pour la représentation des réponses. De plus, un tapis de danse pour Xbox a été acheté, afin de rendre la saisie des réponses plus attractive.

## 2.1. Réalisation :

#### a) Débuts :

Après avoir fait quelques tests avec l'Arduino, nous avons d'abord conçu un programme permettant de saisir les réponses, avec seulement deux choix possibles au début. Nous avons branché des boutons poussoirs en entrée de l'Arduino, pour la sélection des choix. Le tableau d'entiers cpt du programme contient une variable de comptage pour chaque choix. Lorsqu'une pression est exercée sur l'un des boutons, la variable correspondante est incrémentée. La variable NbVotes compte l'ensemble des réponses, elle correspond donc à

la somme des variables du tableau cpt. La variable ValeurMax a toujours une valeur égale à celle de la variable la plus haute dans le tableau.

Ces variables nous permettent de calculer le pourcentage de chaque réponse par rapport au nombre total de réponses, ainsi que par rapport à la réponse la plus validée.

### b) Matrice de LEDs :

A partir de ce moment, nous avons pu commencer à travailler sur la programmation de la matrice de LEDs. Il s'agit d'une matrice 8x8 de chez SparkFunElectronics. Nous avons décidé de représenter les réponses sur cette matrice sous forme de barre-graphe. La matrice ayant une largeur de 8 LEDs, nous aurions pu représenter 8 sorties sur cette matrice, mais nous avons pensé que la représentation de 4 sorties, chaque sortie étant représentée par une barre de largeur 2 LEDs, serait suffisante, plus visible et plus esthétique. Nous avons donc réalisé un programme permettant cette représentation.

Pour ce faire, nous nous sommes aidés d'un programme exemple trouvé sur internet, affichant une phrase lettre par lettre sur la matrice. Nous avons dû modifier ce programme pour afficher le graphe voulu, et non des lettres, sur la matrice.

En adaptant le programme de départ pour avoir 4 entrées, nous avons pu afficher de façon permanente, côte à côte, des barres représentant les pourcentages de chaque réponse par rapport à la réponse revenant le plus souvent. Etant donné que la matrice fait une hauteur de 8 LEDs, on peut considérer qu'en hauteur une LED correspond à 12,5% de la valeur de comptage du choix le plus populaire.

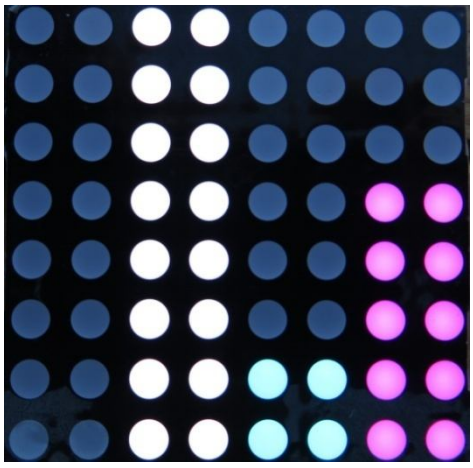

**Figure 3 : Barre-graphe sur matrice de LEDs**

Sur l'affichage de la figure 3, on voit que :

- La réponse 2 est celle qui a eu le plus de succès, elle correspond donc à ValeurMax.
- Le nombre de fois où la réponse 1 a été choisie est inférieur à 12,5% de ValeurMax.
- Le nombre de fois où la réponse 3 a été choisie est compris entre 25% et 37,5% de ValeurMax.
- Le nombre de fois où la réponse 4 a été choisie est compris entre 62,5% et 75% de ValeurMax.

En réalité, on a cpt = { 0, 3, 1, 2 }

Soit cpt[0] = 0% de ValeurMax, cpt[2] = 33% de ValeurMax, et cpt[3] = 66% de ValeurMax.

### c) Ecran LCD :

Le programme faisant fonctionner la matrice de LEDs, nous avons pu commencer la programmation de l'écran LCD. Il s'agit ici d'un écran E-Paper 10x2 de chez SparkFun. Nous nous sommes là encore aidés d'exemples de programmes trouvés sur internet. Nous avons pu récupérer notamment la bibliothèque de définitions des caractères, dans laquelle l'affichage de chaque caractère est défini. Nous avons aussi dû ajouter certains caractères à cette bibliothèque, comme le '%' qui ne s'y trouvait pas. Nous avions au début réalisé un programme affichant pour chaque choix possible son nombre de validation, ainsi que son pourcentage par rapport au nombre de votes total NbVotes. Mais sur le conseil de M. Peter, nous avons modifié le programme pour rendre l'affichage moins lourd, en ne gardant que le pourcentage. Lorsque le programme s'exécute, les pourcentages de chaque réponse s'affichent successivement en boucle. On peut modifier la durée d'affichage d'une réponse en modifiant la valeur de la variable

Timer dans le programme. Sa valeur correspond à une durée en millisecondes.

Avec les résultats de l'exemple précédent, on obtient les affichages de la figure 4.


**Figure 4 : Affichages écran LCD**

Avec cpt={0,3,1,2}, on a NbVotes=6.

On a donc bien : cpt[0] = 0% de NbVotes,

cpt[1] = 50% de NbVotes,

cpt[2] = 16,6% de NbVotes,

et cpt[3] = 33,3% de NbVotes.

Nous avons modifié le programme de façon à avoir l'affichage suivant au démarrage:


**Figure 5 : Initialisation**

### d) Le tapis :

Ayant reçu le tapis dans les dernières semaines, nous avons commencé à réfléchir à son utilisation. Nous avons notamment créé un programme permettant de reconnaitre le tapis lorsqu'il est branché, et de détecter ses changements d'états. Cependant ce programme ne s'exécute que sur un ordinateur. Le but aurait été de le rendre fonctionnel sur l'Arduino, en reliant le tapis à l'Arduino par l'intermédiaire d'une Foxboard par exemple, mais nous n'avons pas eu le temps d'aller jusqu'au bout de cette étape.


**Figure 6 : Tapis**

## 2.2. Manuel d'utilisation :

Le programme de l'Arduino est simple à utiliser. Il suffit d'ouvrir le fichier 'AffichageArduino.ino', accompagné de la bibliothèque 'ePaperDriver.h' dans le logiciel dédié à l'Arduino, puis de le transférer à l'Arduino.

Pour les connexions à réaliser entre l'Arduino et ses périphériques, tout est précisé au début du programme.

Lors de son initialisation, l'Arduino allume entièrement la matrice de LED et affiche sur l'écran "MUSEE ART TOURCOING", et ceci pendant une durée fixée à l'aide de la variable 'Timer' (initialement fixée à 5000) se trouvant dans le fichier 'AffichageArduino.ino'. Une fois l'initialisation terminée, le cycle d'affichage des

réponses se lance. Pour valider une réponse, il suffit d'actionner l'entrée associée. Là encore, pendant une durée modifiable avec la variable ‘Timer’, l'écran et la matrice vont être animés de quelques clignotements.

Lorsque l'un des compteurs de réponses dépasse 9999, les quatre compteurs sont réinitialisés à 0.

Le programme ‘TestTapis.c’ s'exécute comme n'importe quel autre programme c. Il suffit de le compiler dans un terminal à l'aide de la commande 'gcc -o Test TestTapis.c', puis de l'exécuter (en mode administrateur) grâce à la commande ‘./Test’ dans le même répertoire. Ce programme, une fois lancé, cherche le périphérique dont le nom se trouve dans la variable MAT parmi les périphériques USB connectés à la machine. Si ce périphérique n'est pas détecté, le programme se ferme automatiquement. Sinon, le programme surveillera les messages envoyés par le périphérique. Dans notre cas, le nom du périphérique est “ga451-usb device”. Il s'agit de l'identifiant du tapis utilisé lors de la réalisation du projet. Cet identifiant peut être récupéré à l'aide de la commande 'evtest' dans un terminal (sous Linux du moins).

## 2.3. Difficultés :

Nous avons eu des difficultés pour gérer les calculs. L'Arduino n'étant pas optimisé pour manipuler des grands nombres, les types de variables basiques n'ont pas été suffisants pour calculer les pourcentages. Nous ne comprenions pas pourquoi les résultats des calculs n'étaient pas ceux attendus, nous avons donc dû faire de nombreux essais. Finalement, nous avons compris qu’il fallait faire attention aux types de variables et préciser leurs tailles (uint16_t, uint32_t...).

Un autre problème a été de comprendre comment paramétrer le registre pour pouvoir faire fonctionner en même temps la matrice de LEDs et l'écran LCD. Les deux ne pouvant pas

être gérés en même temps, il est nécessaire de configurer le registre pour la matrice à chaque fois qu'on actualise son affichage, puis ceci étant fait, de revenir à l'ancienne configuration.

De plus, n'ayant reçu le tapis qu'en avril, nous n'avons pas pu réfléchir à son fonctionnement avant, et nous avons dû le simuler à l'aide d'un simple bouton poussoir branché sur l'entrée voulue de l'Ardino.

En outre, un problème est resté non résolu avec l'affichage de la matrice : il arrive parfois qu'une LED qui devrait être éteinte s'allume en rouge, de façon tout à fait inopinée.

# 3. Transmission et partage des données :

Pour cette partie du projet, nous avons eu recours au WiFly Shield de Sparkfun, et à une borne Wifi Cisco. Le but est de transférer les données (valeurs des réponses) à une certaine période ou lorsqu'on appuie sur l'un des boutons, vers notre serveur. Nous utilisions un ordinateur de la salle de projet (ZABETH14) comme serveur. Le WiFly Shield nous permet de faire une requête URL en mettant en paramètre les données, ainsi que de les envoyer par Wifi à la borne Wifi Cisco qui les transite ensuite vers le serveur PC. Pour pouvoir communiquer ensemble, il a fallu tout d'abord mettre l'ensemble sous le même sous-réseau. Etant donné que le PC a une adresse IP fixée, nous avons configuré les adresses IP du WiFly Shield et de la borne Wifi de telle sorte qu'ils aient le même masque de sous-réseau. De ce fait, ils arrivent à communiquer et échanger des données entre eux (tests ping fonctionnel).

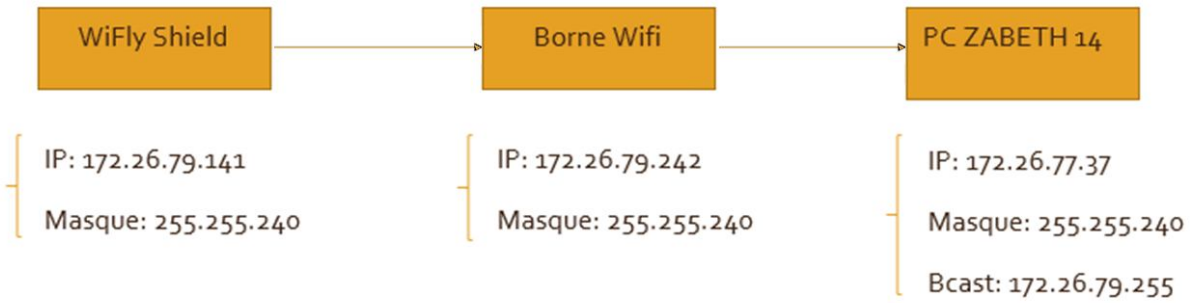


**Figure 7 : Organisation sous-réseau**

## 3.1. Réalisation :

### a) Module WiFly Shield :

Le module WiFly constitue une pièce maitresse du projet. Il nous permet d'envoyer les valeurs des réponses vers la borne Wifi. Afin de le configurer, nous avons réalisé un programme sous Arduino, permettant de visualiser les réponses venant du module WiFly lorsqu'on lui envoie des commandes. Tout d'abord, il faut se mettre en mode commande en envoyant '$$$' via le monitor, le module répond en affichant « CMD ». Ensuite, on fait sa configuration en mettant l'adresse IP qu'on lui a attribuée, le masque, le ssid, le type de clé…:

- set ipaddress172.26.79.141
- set wlanauth 0 // pas d'authentification, donc pas de demande de clé
- set wlanssidprojet_musee
- set ipnetmask 255.255.255.240
- set ipprotocol 0 // utilisation du protocoleUDP

Après ces étapes, et après avoir configuré la borne Wifi, on a réussi à envoyer en mode commande des requêtes URL à un fichier save.php, les sauvegarder dans un fichier 'data'

et vérifier leur bonne réception en visualisant ‘data’ dans un terminal. Les commandes utilisées sont les suivantes :

- set ip address 172.26.77.37 // addresse du serveur (ZABETH 14)
- set ipremote 80 // port pour le serveur
- set com remote GET$/wifly/save.php ?val=X // requête URL
- set sys auto 3 // optionnel pour envoyer tous les 3 secondes
- set option format 1 // envoi automatique dès l’ouverture
- set ip proto 18 // UDP+TCP
- open

Cependant, le système doit être autonome. On a donc créé un autre programme, en utilisant des fonctions de la librairie ‘WiFly.h’, permettant d’envoyer la requête URL à l’appel de la fonction envoi commande, qui prend en paramètre les valeurs des réponses à envoyer.

### b) Borne Wifi :

La borne Wifi fait la liaison entre le WiFly Shield et le PC. Elle nous sert donc de point d’accès. Elle a été configurée grâce à Minicom sous le terminal Unix, et branchée en filaire.


**Figure 8 : Borne Wifi Cisco**

### c) Page Web:

Les données réceptionnées par le PC sont ensuite stockées dans un fichier nommé ici data. Pour réaliser cela, nous avons créé un fichier save.php permettant de récupérer et stocker les données reçues par le PC dans data. A chaque nouvelle arrivée de données, les anciennes sont automatiquement supprimées. Un nouveau fichier 'projet_musee.php' a été créé et constitue ainsi notre page web. Celle-ci exploite ainsi le fichier data et permet de visualiser les données mais aussi de les partager sur Facebook.

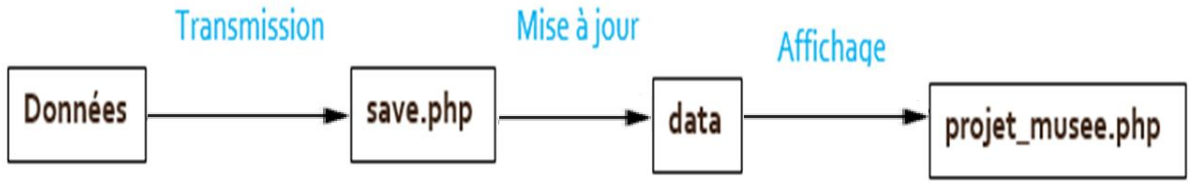


**Figure 9 : Partage des données**

## 3.2. Manuel d'utilisation :

Pour cette partie, il faut :

- Tout d'abord choisir le réseau Wifi auquel on veut connecter le module WiFly.

Pour cela, il faut modifier dans le programme Arduino le ssid et y mettre le nom du Wifi.

- Choisir le serveur qui stockera les données en mettant l'adresse IP correspondant au serveur dans le programme Arduino (au début dans la déclaration de Client client ("X.Y.Z.V",80) avec X, Y, Z et V les valeurs constituant l'adresse IP).
- Configurer le module WiFly, la borne Wifi et le serveur de telle sorte qu'ils soient

dans le même sous-réseau et donc avoir le même masque de sous-réseau en mettant des adresses IP appropriées durant leurs configurations.

- Pour un éventuel debug ou une amélioration du système avec Arduino, il faut veiller à installer la librairie du WiFly au bon endroit c'est-à-dire dans l'emplacement des autres librairies du logiciel Arduino.

## 3.3. Difficultés :

Durant le projet et pour cette partie, on a rencontré certains problèmes. Tout d'abord avec le module WiFly, nous ne savions pas comment passer en mode commande pour pouvoir le configurer. Le problème était que ce dernier répond via le port SPIserial alors que les commandes se font par le port série via le monitor. Ainsi, nous avons créé un programme afin de voir les messages provenant du module WiFly.

La deuxième difficulté a été de faire le programme qui réalise la requête URL et l'envoie au serveur. Au début nous pensions qu'il fallait le faire comme en mode commande avec des 'println' qui se suivent mais nous nous sommes rendu compte que cela ne pouvait pas marcher et qu'il existait déjà des fonctions de bibliothèques téléchargées permettant de réaliser les fonctions nécessaires. Enfin, la compréhension des bibliothèques qui était importante pour pouvoir avancer dans cette partie, a été laborieuse.

Nous avons résolu ces problèmes un par un et au bout du terme, cette partie est fonctionnelle.

# 4. Assemblage des parties :

Pour cette étape, nous avons essayé de rassembler les codes des deux parties en un seul code. Cependant nous avons eu des problèmes sur l'envoi des données mais aussi sur l'écran, à première vue liés aux Cheap Select des modules WiFly et de la matrice de LEDs. Nous avons essayé de nombreuses configurations de connexions de ces périphériques, mais en vain. Apparemment, le même problème serait déjà apparu quelques années auparavant lors d'un projet, et serait sans solution. Cela pousse à se demander s'il existe vraiment une solution. De plus, nous n'avons pas trouvé d'exemple de ce genre de combinaison, malgré les recherches faites sur internet.

Nous n'avons donc pas pu combiner les codes, malgré le fait que les deux parties du projet fonctionnent très bien séparément.

# Conclusion :

Nous pouvons dire que notre participation à ce projet était très intéressante, même si le temps qui nous était imparti pour sa réalisation ne nous a pas permis de le mener à son terme. Cela nous a permis d'approfondir nos connaissances ainsi que notre expérience en programmation.

Par contre, si nous avions eu plus de temps nous aurions pu aller beaucoup plus loin. Après avoir finalisé l'ensemble de ce projet, nous aurions pu, par exemple, avoir un contact direct avec le client, et même éventuellement faire l'installation chez le client, ce qui aurait été d'un grand intérêt pour nous.

Cependant, ce ne sera pas à nous d'aller jusque-là, et c'est pour cette raison que nous avons tenu à laisser un code propre, commenté, compréhensible et modifiable le plus facilement possible, afin qu'il puisse être adapté aux besoins au moment opportun.

# Liste des figures :