Rapport de projet 2014/2015

Visualisation Open Data

Remerciements

Nous souhaiterions remercier M. Boé, M. Redon et M. Vantroys pour leurs conseils et leur aide quand nous les sollicitions.

Introduction

Une donnée ouverte est une donnée numérique qui peut être d'origine publique ou privée. L'Open Data (ou « l'ouverture des données » en français) a pour but de rendre les données numériques accessibles à tous, ainsi que de pouvoir les réutiliser librement, sans restriction juridique, financière ou technique. La SNCF ayant un grand nombre de voyageurs chaque jour (10 millions), l'Open Data lui permet d'optimiser et de valoriser le temps de voyage, de gérer l'affluence dans les trains, ainsi que de s'adapter aux besoins de chaque voyageur.

Ce projet a pour but d'exploiter les données publiées par un site Open Data en les récupérant et en les affichant. Notre choix s'est porté sur le site Open Data de la SNCF afin de répondre aux objectifs du projet. L'idée est de renseigner l'utilisateur le plus facilement et le plus rapidement possible en utilisant les données du site data.sncf.com.

Dans un premier temps, nous avons pensé à recréer une interface web qui reprend les fonctionnalités du site infolignes.com. Ce dernier permet à un client de sélectionner la gare de son choix et de visualiser les trains au départ de cette gare à une heure et une date choisies. Le but était également de mettre en évidence l'éventuel retard des trains.

Dans un second temps, nous avons eu l'ambition de rendre cette application, qui était initialement prévue sur un site web, portable. En effet, l'utilisateur pourrait alors accéder directement à la gare et aux trains de son choix à l'aide d'un Arduino, d'un Shield Ethernet et d'un écran tactile.

I. Présentation du projet

I-1. Les objectifs

Comme spécifié dans l'introduction, le principal objectif du projet est de réutiliser les données fournies par le site Open Data SNCF afin d'informer l'utilisateur de la meilleure manière possible des horaires et éventuels retards des trains d'une gare qu'il a au préalable choisie.

De plus, le projet présentait un autre objectif qui était d'effectuer une connexion avec le projet de fin d'études IMA5 intitulé « Assistance globale pour aide au parking ». Néanmoins, le manque de temps ne nous a pas permis de réaliser cette partie du projet qui restait difficilement réalisable malgré des discussions avec les encadrants et les étudiants de cinquième année.

Cela a fait que nous nous sommes focalisées sur la première partie du projet, sans réfléchir à la seconde partie qui aurait pu être intéressante mais moins importante pour le thème de notre projet.

I-2. Cahier des charges initial

I-2-1. Le matériel utilisé

Pour réaliser ce cahier des charges notre projet, nous avions prévu d'utiliser une matrice de LED et une LED RGB afin d'afficher les données récupérées et un Arduino Mega pour contrôler la matrice et la LED via la liaison série à l'aide d'un code écrit en C.

I-2-2. Les étapes du projet

Dans un premier temps, notre intention était de réaliser un site web basé sur le site infolignes.com qui permettait à un utilisateur de sélectionner une gare, un horaire et un jour afin d'observer les trains au départ de cette gare, leur heure de départ et l'éventuel retard. Pour ce faire, nous avons établi plusieurs étapes pour notre projet.

Tout d'abord, nous devions traiter la récupération des horaires théoriques et en temps réel via les sites Open Data SNCF et infolignes.com. Pour cela, nous avions prévu d'utiliser une base de données afin de stocker les données réelles et théoriques des gares, par exemple l'horaire de départ et la destination d'un train. Pour les données réelles, l'idée était de concevoir un code PHP permettant la récupération de ces données pour une gare. Ces dernières seraient ensuite insérées dans la base de données en SQL. Pour les données théoriques, nous avons pensé à récupérer directement l'archive contenant le fichier texte nous intéressant sur le site Open Data SNCF puis à les insérer dans la base de données à l'aide d'un code SQL. Seuls les numéros et noms des gares existantes dans ce fichier texte.

Ensuite, il nous fallait afficher ces données, c'est-à-dire afficher l'horaire sur la matrice de LED fournie et faire clignoter la LED RGB selon le retard du train. Pour ce faire, nous avons pensé à écrire dans un fichier texte les données souhaitées à l'aide d'un script PHP, puis un code C qui récupère les données de ce fichier pour ensuite les exploiter. Un code C permettant la liaison série avec l'Arduino pour contrôler la LED et la matrice et afficher les données était également nécessaire.

Enfin, le choix des paramètres par l'utilisateur, c'est-à-dire le gare de départ, celle d'arrivée, l'horaire et le jour devait être traité. Pour cela, nous devions créer un site web permettant la sélection des données voulues à l'aide d'un formulaire. Ces dernières seraient récupérées avec un script PHP puis exploitées dans le code PHP créé précédemment afin d'entrer ces données dans la base de données.

I-3. Cahier des charges final

Le cahier des charges a été changé au cours du projet afin de réaliser un prototype plus intéressant et qui n'est pas une simple copie d'un site web déjà existant.

I-3-1. Le matériel utilisé

Afin de répondre aux besoins du nouveau cahier des charges, nous avons prévu un écran Adafruit TFT LCD Shield pour afficher les données, un Shield Ethernet pour assurer la communication entre l'Arduino et le réseau. De plus, un joystick était prévu pour permettre à l'utilisateur d'effectuer ses choix. Enfin un Arduino Uno était nécessaire pour la configuration des différents Shields et joystick.

I-3-2. Les étapes du projet

Après la modification du cahier des charges, l'objectif était à présent de permettre à l'utilisateur de sélectionner de d'afficher les données souhaitées sur un écran et non pas sur un site web.

Pour ce faire, nous devions, dans un premier temps, récupérer toutes les bibliothèques nécessaires au bon fonctionnement du projet, c'est-à-dire les bibliothèques permettant l'affichage sur l'écran, celle gérant la liaison SPI, la communication Ethernet, le joystick, la carte SD et la bibliothèque pour les sockets. Ces dernières étant en C++, il était nécessaire de les « traduire » en C afin d'utiliser le langage C pour coder notre application.

Dans un second temps, le but était de concevoir un programme principal afin de récupérer les informations sur les gares situées dans le fichier texte sur le site Open Data SNCF à l'aide de la bibliothèque socket, puis de les stocker dans la carte SD. Il fallait également penser à trier les informations récupérées pour ne garder que les noms des gares et les codes qui leur sont associés. Ensuite, nous devions gérer l'écran TFT et le joystick via les bibliothèques correspondantes dans le but de permettre à l'utilisateur de choisir une gare parmi celles stockées dans la carte SD. Il nous fallait ensuite récupérer les informations liées à cette gare par requête HTTP en utilisant la bibliothèque socket. Suite à cela, l'utilisateur peut alors choisir le train dont il souhaite avoir de plus amples informations, puis il nous fallait les afficher sur l'écran.

II. Réalisation du projet

II-1. Première solution

Pour la première version du cahier des charges, nous avions trois principales parties à gérer : la récupération et le traitement des données réelles et théoriques, l'affichage de ces données, puis l'interaction avec le client à l'aide du site web.

II-1-1. La récupération des données

Nous avions deux types d'informations à récupérer. En effet, il nous fallait les données théoriques correspondant aux différentes gares et les données réelles qui sont récupérées suite à la demande de l'utilisateur.

Tout d'abord, nous devions stocker ces informations récupérées. Pour cela, nous avons créé une base de données nommée ProjetOpenData sous MySQL et implémenté les tables « Villes » et « Gares ». La table « Gares » est utilisée pour stocker les informations concernant la gare choisie par l'utilisateur. Ces données sont les trains au départ de cette gare, leur horaire de départ ainsi que leur éventuel retard. La gare « Villes » est quant à elle utilisée pour entrer les informations en relation avec les données théoriques. Ces données sont les codes et les noms des gares.

Les données relatives aux gares sont dans un fichier texte nommé « stops.txt » situé dans une archive sur le site Open Data de la SNCF. Après téléchargement et extraction des fichiers de cette archive, le fichier était alors accessible. Nous avons ensuite traité puis entré les données contenues dans ce fichier à l'aide d'un code SQL. Pour ce faire, nous avons tout d'abord scindé ce fichier en deux car nous avons remarqué que chaque gare apparait deux fois mais sous deux formes différentes.

Nous avons alors gardé les lignes qui sont de la forme suivante : StopArea:stop_id,stop_name,,stop_desc,stop_lat,,,stop_lon,

Les champs que nous souhaitions récupérer sont « stop_id » et « stop_name » correspondant respectivement au code et au nom des différentes gares. De plus, le code de la gare

est à la suite d'une chaine de caractères qui ne changent jamais selon les lignes qui est : « StopArea: » et deux virgules se situent après nom de la gare. Les deux champs sont séparés par une virgule. Cela était donc facilement traitable en SQL et les données étaient bien chargées dans la table « Villes » de la base de données.

Pour récupérer les données du site infolignes.com, nous avons utilisé un script PHP. En effet, le but était de récupérer le code source HTML de la page du site infolignes.com suite à la requête formulée par l'utilisateur. Nous avons alors créé un DOMDocument qui nous permettait de récupérer le code HTML de la page à l'aide de la fonction loadHTMLFile. Cette dernière prend en paramètre l'url de la page que nous avons entré en dur en attendant les données de la requête. Ensuite, un tableau était créé afin de stocker tous les éléments situés entre les balises <td> dans le code source récupéré. En effet, ces éléments correspondaient parfaitement aux données que l'on cherche à extraire du site.

Cependant, le traitement de ces données n'a pas été effectué même si nous avons réussi à les insérer dans la table « Gares » de la base de données à la suite de ce script SQL.

II-1-2. L'affichage des données

Le code permettant l'affichage sur l'Arduino a été fait exclusivement en C et ressemble à ce que l'on a vu en TP ordonnancement de système. Seule la partie pour contrôler la LED a été codée mais n'a pas été testée.

Pour faire le lien entre les données à afficher situées dans la table « Gares » de la base de données, nous avions pensé à écrire ces données dans un fichier texte nommé « destination.txt » à l'aide d'un script PHP. Nous pensions ensuite pouvoir les récupérer avec l'ouverture de ce fichier en C, afin de les récupérer et les afficher. Cependant, nous n'avions pas pensé au fait que c'était l'Arduino que l'on codait et donc que ce que l'on souhaitait faire n'était pas possible.

II-1-3. Interaction avec le client à l'aide d'un site web

Malheureusement, nous n'avons pas traité cette partie pour cause d'un manque de temps et d'un changement de cahier des charges au bout de 6 semaines de projet. Etant donné que le

site web ne serait pas utilisé dans la suite du projet, il ne nous semblait pas nécessaire de le concevoir.

II-2. Deuxième solution

Initialement, l'écriture du code devait s'effectuer en C. Cependant, nous avons décidé de coder à l'aide de l'IDE Arduino afin de simplifier le travail qui pouvait être important.

II-2-1. Les bibliothèques nécessaires

Tout d'abord, comme spécifié dans le cahier des charges, il nous fallait récupérer les bibliothèques nous permettant de contrôler l'écran TFT qui sont « Adafruit_ILI9341.h » et « Adafruit_GFX.h », le Shield Ethernet ainsi que le joystick. Il nous fallait également gérer la liaison SPI, l'enregistrement des données dans la carte SD ainsi les requêtes HTTP pouvant être effectuées à l'aide de la bibliothèque socket.

Les bibliothèques précédentes ont été codées en C dans le but de ne pas utiliser l'IDE Arduino. Cependant, nous avons utilisé les bibliothèques en C++ dans la suite du projet étant donné l'écriture du code à l'aide de l'IDE Arduino comme spécifié ci-dessus.

Suite à ce choix, nous avons également décidé de ne pas utiliser le joystick et donc de profiter de l'écran tactile que nous avions à notre disposition pour en exploiter ses fonctionnalités, donc d'utiliser la bibliothèque « Adafruit_STMPE610.h » prévue à cet effet.

De plus, nous avons utilisé la bibliothèque « SM.h » qui nous permettait de construire une machine d'états utile pour la conception de notre programme.

II-2-2. Structure du programme principal

Comme pour la solution précédente, il nous fallait récupérer les données situées sur les sites Open Data SNCF et infolignes.com pour ensuite les traiter.

Tout d'abord, on récupère l'archive située sur le site Open Data SNCF contenant le fichier texte qui inclut les gares existantes en France et dans des pays étrangers situés autour de la France. L'archive est extraite, puis, un traitement en C est effectué sur ce fichier texte afin de ne garder que les codes et les noms des gares séparés par des virgules. Nous créons ensuite 26 fichiers texte pour chaque lettre de l'alphabet. Par exemple, tous les noms de gares commençant par 'A' seront dans le fichier « gares_0.txt ». Ces nouveaux fichiers texte sont alors stockés sur un serveur web Apache. Ce traitement est pour l'instant réalisé à la main, cependant une solution optimale est envisagée. En effet, nous avions pensé à écrire un fichier Shell afin de récupérer l'archive située sur le site web. Ce Shell ferait de plus l'extraction de l'archive et enverrait le fichier texte « stopstxt » sur le serveur web.

Nous avons ensuite écrit un script Arduino qui nous permet d'accéder à ce serveur web à l'aide d'une requête HTTP, puis stocke les fichiers texte dans la carte SD située sur le Shield TFT. Ce traitement est pour le moment réalisé une seule fois.

Puis, nous avons géré les fonctions permettant l'affichage sur l'écran, la sélection de données par l'utilisateur à l'aide de l'écran tactile, la requête HTTP pour récupérer les données sur le site infolignes.com ainsi que le traitement de ces données. De plus, nous avons créé une machine d'états qui nous permet de passer d'un écran à un autre à la suite d'événements précis.

II-2-3. L'application en détails

Nous avons choisi d'utiliser une machine d'états dans le but d'afficher plusieurs écrans différents dans notre application.

En effet, il en existe cinq différents. Le premier permet de sélectionner la première lettre de la gare à l'aide d'un clavier, le second affiche toutes les gares correspondantes et permet d'en sélectionner une. Le troisième laisse choisir à l'utilisateur l'heure et la date du départ, tandis que le quatrième affiche tous les trains correspondant à la requête transmise par le client et permet la sélection d'un train en particulier pour avoir de plus amples informations. Enfin, le dernier écran affiche les informations du train demandé par l'utilisateur.

II-2-3-1. L'écran d'accueil

Pour le premier écran, nous avons créé deux fonctions SAccueilH() et SAccueilB() qui correspondent aux fonctions « head » et « body » de la machine d'état.

La fonction « head » nous permet l'affichage en général sur l'écran. Sur cet écran, nous affichons un clavier pour sélectionner la lettre, un bouton « OK » et un bouton « MAJ Gares ». Elle nous permet également de mettre en couleur la touche de la lettre du clavier sélectionnée par l'utilisateur.

La fonction « body » nous permet d'effectuer les différentes actions suivant les événements qui se déroulent. En effet, si le clavier est touché par l'utilisateur, une fonction détecte automatiquement la lettre sélectionnée selon l'emplacement du doigt sur l'écran. De plus, une variable globale stocke la lettre sélectionnée. Puis, si une lettre est sélectionnée et que l'utilisateur appuie sur le bouton « OK », le fichier contenant les noms et codes des gares stocké dans la carte SD s'ouvre et l'ensemble des gares commençant par la lettre sélectionnée est stocké dans une structure qui contient le nom de la gare et le code de la gare en paramètres. Enfin, nous passons à l'écran suivant à l'aide de la fonction « Set » de la bibliothèque de la machine d'états. De plus, un bouton « MAJ Gares » est disponible mais n'est pas réactif car la fonction correspondante n'a pas été codée. Ce bouton aurait permis le lancement du script PHP énoncé plus haut, c'est-à-dire, le traitement du fichier texte « stops.txt » pour stocker les codes et noms des gares dans 26 fichiers texte selon la première lettre du nom de la gare, la récupération immédiate de ces fichiers ainsi que le stockage de ces derniers sur la carte SD.

II-2-3-2. L'écran de sélection de la gare

Comme pour l'écran précédent, nous avons créé une fonction « head » et une fonction « body ».

La fonction « head » permet l'affichage de quatre gares commençant par la lettre sélectionnée et d'un bouton « OK » qui permet de passer à l'écran suivant. De plus, un bouton « Home » permet de revenir instantanément à l'écran d'accueil de l'application. Des boutons « Suiv » et « Prec » ont également été affichés afin de voir les gares suivantes ou précédentes. Enfin, elle permet de colorier la case correspond à la gare sélectionnée.

La fonction « body » de cet écran permet alors de sélectionner la gare voulue. L'utilisateur peut afficher les gares suivantes s'il souhaite afficher les détails d'une gare ne se trouvant pas directement sur l'écran. Lorsqu'il appuie sur ce bouton, les gares suivantes se chargent automatiquement sur l'écran. Cependant, le bouton permettant d'afficher les gares précédentes n'est pas fonctionnel. De plus, un bouton « home » fait son apparition sur cet écran et permet de revenir à l'écran d'accueil en appelant la fonction « Set » qui affiche l'état SAccueil. Lorsqu'une gare est sélectionnée, la variable globale est mise à jour selon la position de cette gare. Nous avons alors le code correspondant à la gare en faisant appel au champ « code » de la structure précédente. Enfin, lorsque l'on appuie sur le bouton « OK » et qu'une gare est bien sélectionnée, nous passons à l'écran suivant, et un tableau de dates est généré automatiquement pour ce dernier.

II-2-3-3. L'écran de sélection de l'horaire et du jour

Cet écran permet le choix de la date et de l'horaire à l'utilisateur. Deux fonctions pour cet état sont alors créées.

La fonction « head » affiche de nouveau les boutons « Home » et « OK », et affiche de plus un tableau contenant les cinq dates possibles ainsi que des boutons de sélection pour les heures et les minutes. Le choix de cinq dates a été effectué selon les requêtes possibles sur le site infolignes.com. Enfin, la date sélectionnée est coloriée si la variable globale correspondante le permet.

La fonction « Body » permet l'incrémentation et la décrémentation des heures et des minutes selon le choix de l'utilisateur. Les minutes sont toujours égales à 0 ou 30. Nous pouvons également sélectionner la date parmi les cinq possibilités possibles. Lorsqu'une gare est sélectionnée, la variable globale correspondante est mise à jour. Enfin, l'appui sur le bouton « OK » provoque immédiatement l'envoi de la requête HTTP pour accéder aux données demandées par l'utilisateur. En effet, nous avions besoin de la gare et donc du code de cette gare, de la date et de l'horaire. A l'aide de cette requête, nous accédons au serveur web décrit précédemment afin d'exécuter un script PHP situé sur ce dernier permettant l'envoi de la requête. Ce script PHP a été utilisé dans la première solution de ce projet. Les données sont ensuite renvoyées à l'Arduino qui stocke les informations dans un fichier texte dans la carte SD. Une fonction est ensuite appelée

permettant le remplissage d'un tableau avec les données récupérées. L'écran suivant est enfin affiché.

II-3-2-4. L'écran d'affichage des trains

Cet écran permet l'affichage des trains au départ de la gare choisie par l'utilisateur. Les données relatives aux trains tels que l'horaire de départ, la destination et l'éventuel retard sont également affichés.

La fonction « Head » affiche quatre trains à l'écran, ainsi que les boutons « Home », « Prec », « Suiv » et « OK ». Si un train est à l'heure, la case correspondante est en vert, sinon, elle est en rouge. Lorsqu'un train est sélectionné par l'utilisateur, la case correspondante est coloriée en bleu.

La fonction « Body » permet de sélectionner un train lorsque l'utilisateur appuie sur une des cases correspondantes. De plus, étant donné que seulement quatre trains sont affichés, le bouton « Suiv » permet un passage aux trains suivants. Nous pouvons remarquer que le bouton « Home » entraine la remise à zéro (ou à -1) de toutes les variables globales que l'on a pu modifier jusqu'à présent. Enfin, l'appui sur le bouton « OK » provoque l'apparition de l'écran final.

II-3-2-5. L'écran final

Cet écran renseigne plus particulièrement l'état du retard, s'il existe. En effet, nous avions stocké dans une structure qui récupère les données (destination, horaire, si retard le nombre de minutes ou d'heures) de chaque train, une variable « onTime » qui indique si le train est à l'heure ou en retard.

Nous affichons alors un rappel des données sélectionnées par l'utilisateur. En effet, nous pouvons observer la gare de départ, la gare d'arrivée, ainsi que la date et l'horaire du jour. De plus, l'horaire de départ du train est affiché. Enfin, si la variable onTime est égale à « true », alors le train sera affiché en retard, sinon il sera affiché à l'heure.

Enfin, lorsque l'utilisateur appuie sur « Home », toutes les variables globales sont remises à zéro pour éviter que les cases restent coloriées. De plus, un retour vers l'écran d'accueil est effectué.

Conclusion

Pour conclure, le sujet de ce projet était la visualisation Open Data et l'objectif était de récupérer des données à partir du site Open Data de la SNCF, effectuer le traitement de ces dernières, et pour finir les afficher.

Pour cela, nous avons tout d'abord pensé à réaliser un site web reprenant le même principe que le site infolignes.com afin de permettre à l'utilisateur de sélectionner un gare de départ, une gare d'arrivée, un horaire et une date et d'afficher à l'aide d'une LED RGB et d'une matrice de LED l'horaire de départ et l'éventuel retard. Cependant, cette solution n'a pas abouti pour cause d'une application web similaire à celle existant déjà et donc non innovatrice.

Nous avons donc effectué un nouveau cahier des charges. Le but était à présent d'effectuer les mêmes requêtes que pour la première solution mais en utilisant cette fois un écran et un joystick associés à un Arduino Uno et à un Shield Ethernet. Cependant, la fonction tactile de l'écran, TFT a remplacé le joystick. Cette solution a pu être codée et implémentée et fonctionne.

Cependant, plusieurs fonctionnalités manquent à notre application comme la possibilité de retourner en arrière dans les menus et dans le choix des gares ou trains. Le fait d'implémenter la fonction de mise à jour automatique des fichiers texte dans la carte SD aurait été intéressante à effectuer et serait une amélioration possible.

Ce projet nous a permis de découvrir des choses, notamment en programmation avec le langage PHP, et surtout le fait de gérer plusieurs fonctionnalités en même temps comme effectuer une requête HTTP ou afficher du texte sur un écran. Il nous a beaucoup appris et sera certainement utile pour la suite de nos études et carrières professionnelles.