Compte rendu Projet UF Dev

Membres du groupe :

Dylan Levraud

Owain Charlon

Paul Bagnis

**1/ Introduction**

Notre projet consistait en la création d’une plante connectée. Ce projet serait mis en place via un programme codé en langage C sur le pc du client, et une partie « connectée » grâce à un Arduino qui serait intégré dans un pot afin de récupérer les informations concernant le bon développement de la plante.

**2/ Déroulé**

2.1 Architecture

Tout d’abord, nous avons conceptualisé le programme grâce à un flowchart représentant les fonctionnalités de l’application et son architecture (comment le client va naviguer dans l’application).

Partant de cette architecture, nous avons commencer à développer les menus et la navigation entre les différentes parties du programme (demande d’e-mail au lancement, menu de choix, menu d’ajout de plantes, menu de sélection de plantes…).

Pour cela nous avons utilisé les librairie « stdio.h », « stdlib.h » et « windows.h », nous permettant de faire appel aux fonctions « printf », « scanf » et « system(« cls ») ». L’idée étant de diviser notre programme avec un fichier de format .c, les autres étant des librairies (au format .h) qui s’appellent en cascade au fur et a mesure de la navigation dans le programme.

La première étape est l’enregistrement du mail de l’utilisateur dans un fichier texte, nous avions pensé cela dans l’optique de pouvoir ensuite notifier l’utilisateur des besoins de la plante par e-mail. Pour cela, nous avons utilisé les fonctions d’ouverture, de fermeture et d’écriture dans les fichiers.

Pour cela, nous avons codé les fonction « login », « save », « load » et « verif\_mail ». Respectivement, la première demande son mail a l’utilisateur, la deuxième sauvegarde le mail dans un fichier texte, la troisième récupère le mail pour le stocker dans une variable si celui-ci est déjà défini et la dernière permet de vérifier si l’utilisateur a déjà entré son mail (donc s’il y a déjà quelque d’écrit sur le fichier texte sur lequel on sauvegarde le mail).

Ensuite il nous fallait créer un menu de navigation au sein du programme. Dans ce but, nous avons principalement utilisé les fonctions « printf » et « scanf » afin de proposer les différents choix à l’utilisateur et récupérer son choix afin de l’emmener sur la partie souhaitée.

2.2 La base de données

A ce stade nous pensions stocker les données des différentes plantes sur un fichier texte, n’ayant pas encore eu les cours de SQL (solution pour laquelle nous avons finalement opté).

Nous avons donc créé un menu référençant les 30 plantes de base du programme avec les différents besoins de chacune en ensoleillement, ph, eau et le substrat optimal pour la plante. Nous avons choisi de les représenter au travers de notes de 1 à 10. L’utilisateur aura donc une liste des plantes de base et pourra en sélectionner une afin d’obtenir les détails la concernant (l’idée étant par la suite de charger les informations sur le matériel connecté au pot).

Il nous fallait ensuite créer une page sur laquelle l’utilisateur pourrait rentrer les informations concernant une plante de son choix, afin de l’ajouter à la base de données. Nous avons donc créé la fonction « ajout\_plante » qui demande à l’utilisateur de saisir les informations nécessaires et les stocker dans des variables afin de les enregistrées en base de données plus tard.

Pour celle-ci, ayant reçu des cours de SQL, nous avons repensé le système de stockage des données. Nous avons donc créé une base de données selon le diagramme Merise que nous avons réalisée. Nous avons créé un script de création de base de données, ne sachant pas comment la créée directement sur l’ordinateur de l’utilisateur via le programme en C. Ce script créé les tables, les référencements de chacune des tables au sein des autres (Clés étrangères) en créant les 30 plantes de base.

Maintenant il nous faut relier la base de données au code en C afin de pouvoir insérer en base de données la/les plante(s) ajoutée(s) par l’utilisateur. Pour cela nous avons utilisé l’API MySQL en C. C’est là que nous avons eu de gros problèmes. Cette librairie et ses lignes de programmations utilisant les structures ou encore l’allocation dynamique de mémoire, notions que nous n’avions pas abordés au sein de notre formation, nous n’avons pas réussi à trouver un moyen d’utiliser cette fonction tel que nous l’envisagions : stocker en base de données les informations, où aller chercher des informations relatives aux plantes afin d’afficher dynamiquement ces dernières. En effet, dans la fonction qui ouvrait la base de données et dans laquelle on faisait les injections SQL (select ou insert), nous n’arrivions pas à la mettre dans une librairie en récupérant les données (grâce au retour de la fonction). Nous avons supposé que nous ne pouvions pas récupérer de données via cette fonction due à l’allocation dynamique de mémoire car, quand nous cherchions à retourner une chaine de caractère contenant les informations nécessaires, nous ne pouvions pas le faire après avoir libérer la mémoire allouée à la structure contenant les informations sur la base de données. La seule solution à laquelle nous avons pensé, mais que nous n’avons pas eu le temps de mettre en place, aurait été de ne pas avoir cette fonctionnalité en librairie mais directement dans le fichier « main.c », encadrant la totalité du programme (ouverture et connexion à la base de données au lancement de celui-ci et fermeture de la base après avoir charger les informations sur l’objet connecté). Cependant cette solution nous obligeait à repenser entièrement l’architecture de notre programme.

C’est à ce moment-là que nous avons vraiment bloqué, car le fait de ne pas réussir à faire ces fonctionnalités dans les fonctions au sein d’une librairie rendait notre programme non adaptable aux différentes options de retour en arrière dans les pages du menu pour sélectionner une plante ou finalement saisir les données d’une nouvelle.

2.3 Objet connecté

Pour finir, il nous restait la partie objet connecté à traiter. Même si nous n’avons pas eu le temps de la réalisée, nous pensions faire communiquer l’Arduino (avec le capteur de température, celui d’ensoleillement et celui d’acidité), avec notre programme via le server Wamp qui administre la base de données. L’idée étant d’envoyer les données sur le serveur Wamp grâce à un site Web qui aurait stocké les données reçues en base de données. Le programme quant à lui aurait récupérer ces données afin de les traiter et renvoyer des messages en fonction des différentes valeurs reçues comparées aux valeurs optimales enregistrées en base de données.

Le programme Arduino aurait consisté en plusieurs parties :

* La première consiste en la déclaration des variables avec les entrées associés (numéro de port de l’Arduino). Donc paramétrer les 3 capteurs différents. Il nous fallait donc faire appel à la librairie « dht.h » contenant toutes les fonctions nécessaires au fonctionnement du capteur de température et d’humidité.
* La deuxième consistant en l’initialisation (setup) du programme grâce aux fonctions « Serial.begin » pour définir le port de sortie des informations, la fonction « dht.Begin » qui initialise le capteur DHT22, la fonction « WiFi.begin » qui permet au matériel de se connecter au wifi local sur lequel serait aussi connecté le serveur Wamp ainsi qu’une partie de code qui vérifie si la connexion s’est bien déroulée.
* La troisième et dernière partie est la fonction « loop » de l’Arduino, ce code sera effectué en boucle sur l’Arduino. Celui-ci aurait récupérer les valeurs des différents capteurs et les auraient envoyés au serveur sous la forme de requêtes http, celui-ci les aurait enregistrés en base de données via les injections SQL en PHP.

Cette partie nous posait différents problèmes. Le premier étant la communication du nom du Wifi et de son mot de passe à l’Arduino afin de le paramétrer. Le deuxième étant la récupération des données en base de données car il aurait fallu un site internet ouvert qui se rafraichisse à chaque « loop » du programme Arduino pour enregistrer les données envoyées, et nous n’avions aucune idée de la façon d’automatiser un tel procédé. Le dernier est que nous n’avons pas réussi, au cours d’un exercice auxiliaire, à envoyer des requêtes au serveur Wamp grâce une requête http.

**3/ Conclusion**

Nous aurions aimé mener à terme ce projet malgré les problèmes rencontrés. Nous avions encore des idées afin d’améliorer les fonctionnalités du programme comme par exemple affiché sur forme de graphique l’évolution des données (comme l’hydrométrie) pour donner toujours plus d’options à l’utilisateur.