BACCALAUREAT TECHNOLOGIQUE

SESSION 2013

SCIENCES ET TECHNOLOGIES DE L'INDUSTRIE ET DU DEVELOPPEMENT DURABLE

Épreuve de SPECIALITE : **SYSTÈMES D'INFORMATION ET NUMERIQUE**

MANUEL Alexandre

Lycée Gustave EIFFEL – Cachan (94)

Station météo autonome pour site isolé

Transmission et affichage des données sur une machine distante

SOMMAIRE :

Préambule – Présentation du projet

A / Cahier des charges

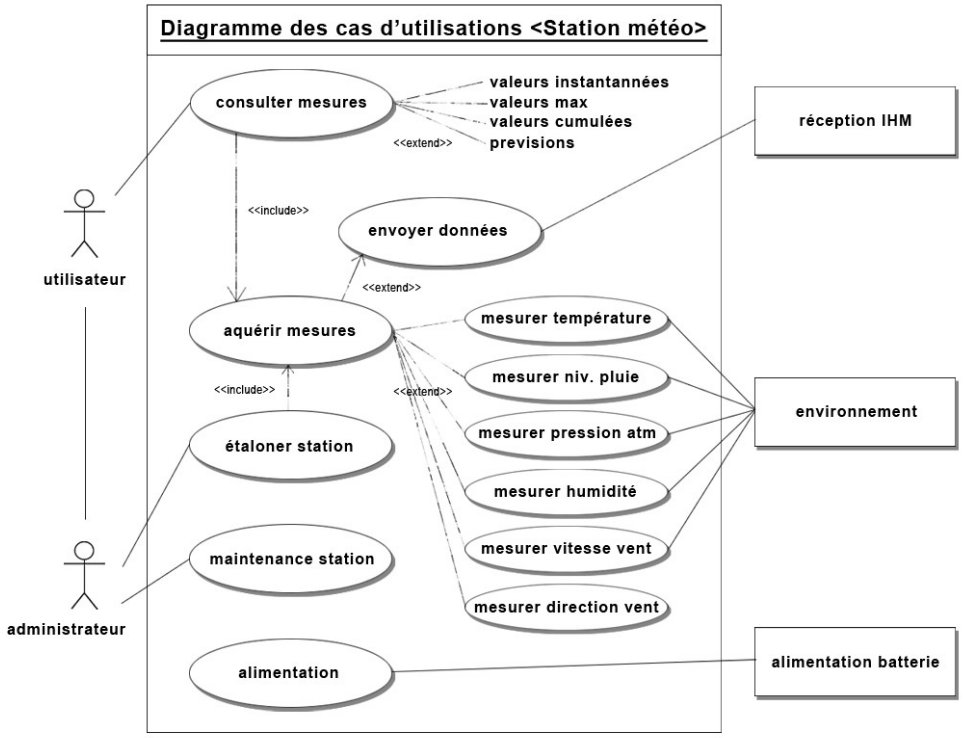
La station météo doit fonctionner en site isolé, c'est à dire qu'elle doit être autonome et ne doit pas nécessiter de câblage avec une installation distante. Elle est donc auto-alimentée. La station devra être capable de recevoir plusieurs grandeurs physiques puis de les afficher. Elle devra également être capable de transmettre les données à un ordinateur distant pour que ce dernier les affiche.

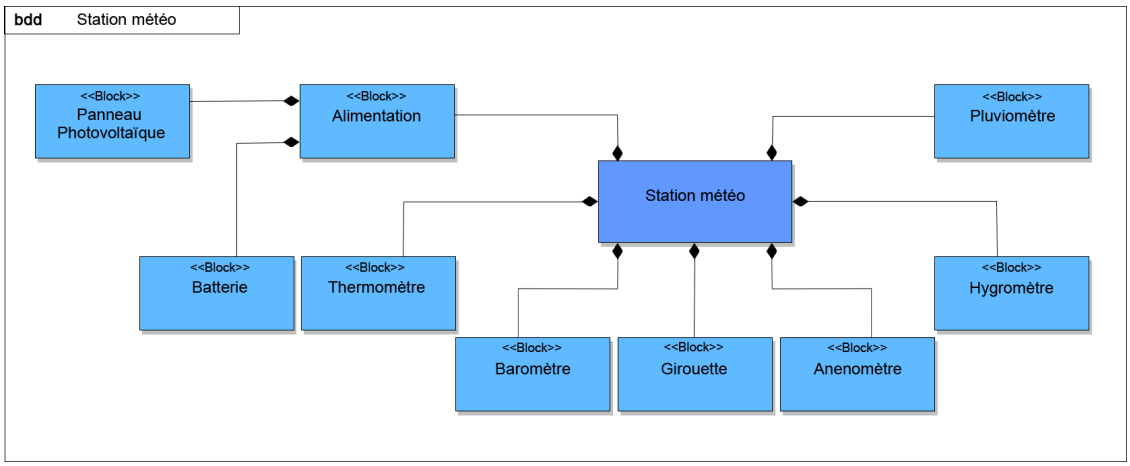
Les grandeurs physiques à mesurer sont :

- La température → De -25 °C à +45 °C au degré près
- Pression atmosphérique → de 900 hPa à 1100 hPa à 1 hPa près
- Précipitations → Mesure de -25 °C à +45 °C. Sensibilité : 0,2mm
- Humidité → De 0 % à 100 %. résolution : 1 %
- Direction du vent → 16 directions possibles
- Vitesse du vent → résolution : 1km/h

A noter que le capteur de température devra délivrer un signal positif à un radiateur positionné près du pluviomètre en cas de température inférieure à 0 °C

B / Diagrammes



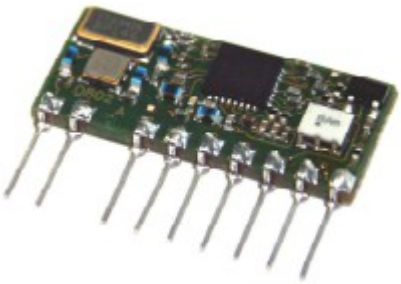


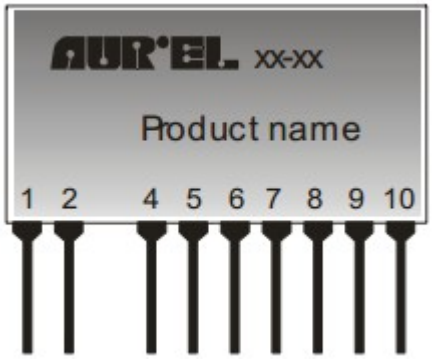
I – Émission

A / Composant utilisé

Pour l'émission des données j'ai choisi d'utiliser le module Aurel RTX MID 5V. Ce composant est un émetteur récepteur radio haute fréquence. Le composant étant déjà en la possession de mon professeur, le coût du prototype à ce niveau est nul. Si la station devait être construite à grande échelle il faut savoir que ce composant coûte 23€30 sur électronique diffusion.

Ce composant comporte 9 broches comme on peut le voir sur la photo et le schéma ci dessous :





Les caractéristiques du composant sont les suivantes :
Tension d'alimentation → 5V
Consommation :
- Pendant émission d'un bit de niveau logique haut → 65mW
- Pendant émission d'un bit de niveau logique bas → 22,5mW
- En veille (broche enable non alimentée) → 6mW
- En réception → 32mW

Fréquence → 433,92MHz
Débit d'information maximal → 9600 bit/sec

B / Longueur de l'antenne

Pour calculer la longueur de l'antenne j'ai calculé la longueur d'onde correspondante à la fréquence 433,92MHz. J'ai donc
longueur d'onde = vitesse lumière / fréquence = 300000000 / 433,92*10^6 = 0,69m
J'ai décidé de faire une antenne quart d'onde pour avoir un excellent compromis entre taille de l'antenne et stabilité d'émission.
Cela m'a donné une antenne de 69cm / 4 = 17,25cm
J'ai ajouté un plan de masse de 15x15cm pour plus d'efficacité.

C / Algorithme de l'émission

Pour l'émission des données j'ai utilisé USART. Depuis le PIC je n'ai donc plus qu'à envoyer bêtement chaque variable. Pour plus de sécurité au niveau de la transmission j'envoie les données 3 fois. Ainsi, si un problème de transmission survient sur un caractère, je peux comparer aux deux autres versions. Voici l'algorithme de l'émission des données :

```
pour i = 0, tant que i < 3 faire
    tant qu'il reste des données à envoyer
        envoyer une donnée avec USART
        attendre que le buffer soit vide
    fin tant que
fin tant que
```

II – Réception

A / Composant utilisé

Pour la réception des données, j'ai utilisé le même composant que pour les envoyer étant donné que le composant fait office d'émetteur – récepteur. Je reprends don mon module Aurel RTX MID 5V.

Le câblage du composant est le suivant :

Numéro de la broche	Description de la broche	Relié à
1	Antenne	Antenne
2	Ground	Masse
4	Data In	Masse → On envoie rien alors on branche l'entrée sur la masse
5	TX / RX	Masse → Mode réception
6	Enable	Vcc → Pas de problèmes énergétique donc on allume tout le temps
7	Ground	Masse
8	Analog out	Broche de débogage uniquement
9	Data out	Câble série – USB
10	Vcc	Vcc

B / Conversion série – USB

Afin de faire la conversion des données séries à l'USB, j'ai utilisé un câble contenant une puce FTDI représenté sur la photo suivante :

Ce câble a coûté **£30.11** soit **€35,60** en euros.

C / Code source de la réception

Pour recevoir les données envoyées par UART, je n'ai pas pu utiliser cette même librairie à cause du câble FTDI. J'ai ici dû inclure ftdi.h et ftdi.c dans mon projet et utiliser les fonctions de ces fichiers. Pour savoir me servir de ces fonctions j'ai suivi cet article explicatif :
http://yosemitefoothills.com/Electronics/
Les données reçues ont chacune été placées dans une variable différente. Le langage C étant un langage à variable typées, il a fallu définir le type de chaque variable. J'ai utilisé les mêmes types que lors de l'envoi soit :

```
char directionDuVent[3];
double hygrometrie;
double pluviometrie;
int pressionAtmospherique;
int temperature;
double vitesseDuVent;
```

D / Archivage des données

Je n'ai pas placé les données dans la base de données en langage C. Pour cela j'ai utilisé un script PHP que je détaillerais dans la partie III. Ici je les ai simplement écrites dans un fichier. L'algorithme d'archivage est le suivant :

ouvrir le fichier de sortie en écriture et création (« w+ »)
écrire la direction du vent puis un « ; » dans le fichier
écrire l'hygrométrie puis un « ; » dans le fichier
écrire la pluviométrie puis un « ; » dans le fichier
écrire la pression atmosphérique puis un « ; » dans le fichier
écrire la température puis un « ; » dans le fichier
écrire la vitesse du vent puis aller à la ligne dans le fichier
fermer le fichier
faire appel au script PHP d'envoi de ces données dans la base de données.

L'implémentation en langage C est la suivante :

```
FILE* output = fopen("output.mdta", "w+");
if (output == NULL)
{
	printf("impossible de creer le fichier output.mdta\n Droits insuffisants");
	exit(-1);
}
else
{
	fprintf(output, "%s;", directionDuVent);
	fprintf(output, "%lf;", hygrometrie);
	fprintf(output, "%lf;", pluviometrie);
	fprintf(output, "%lf;", pressionAtmospherique);
	fprintf(output, "%lf;", temperature);
	fprintf(output, "%lf\n", vitesseDuVent);
	fclose(output);
	system("php saveData.php");
}
return 0;
```

III – Traitement

Une fois arrivées sur le disque dur de l'ordinateur, les données doivent être archivées dans la base de données.

A / Logiciels utilisés

Pour enregistrer les données, j'ai utilisé le langage de script PHP. C'est ce script qui va venir ouvrir le fichier qui contient les données puis les mettre dans la base. Le script chargé de faire l'enregistrement des données dans la base doit s'exécuter toutes les 10 minutes. Pour cela, dans un soucis de portabilité du système, j'ai évité de faire appel aux tâche CRON sous Unix ou au gestionnaire des tâches sous Windows. J'ai préféré exécuter le script PHP directement depuis le programme C qui s'occupait de la réception.

B / Récupération des données archivées

Pour récupérer les données stockées dans le fichier, j'ai fait appel à plusieurs fonctions PHP. D'une part fopen pour créer le flux de fichier. D'autre part j'ai utilisé la fonction fscanf pour lire le fichier en question. L'ouverture du fichier se fait d'abord en lecture uniquement puis le fichier est supprimé grâce à la fonction unlink.
L'algorithme de récupération des données est le suivant :

ouvrir le fichier en lecture (mode « r »)
lire la première ligne du fichier. Sur cette ligne lire 6 données chacune séparées par un point virgule.
Placer les 6 données dans des variables
fermer le fichier
détruire le fichier avec unlink

C / Archivage dans une base de données

Une fois les dernières données de la station en mémoire, il faut les rentrer dans la base de données. Pour ça j'ai utilisé PDO avec PHP pour la connexion à la base de données. PDO permet de se connecter à une base de données sans se soucier de son type (MySQL, PostGreSQL, etc...), une sorte de surcouche en fait.
PDO protège également la base contre les injections SQL, l'attaque informatique la plus répandue de nos jours. Le code d'enregistrement est le suivant :

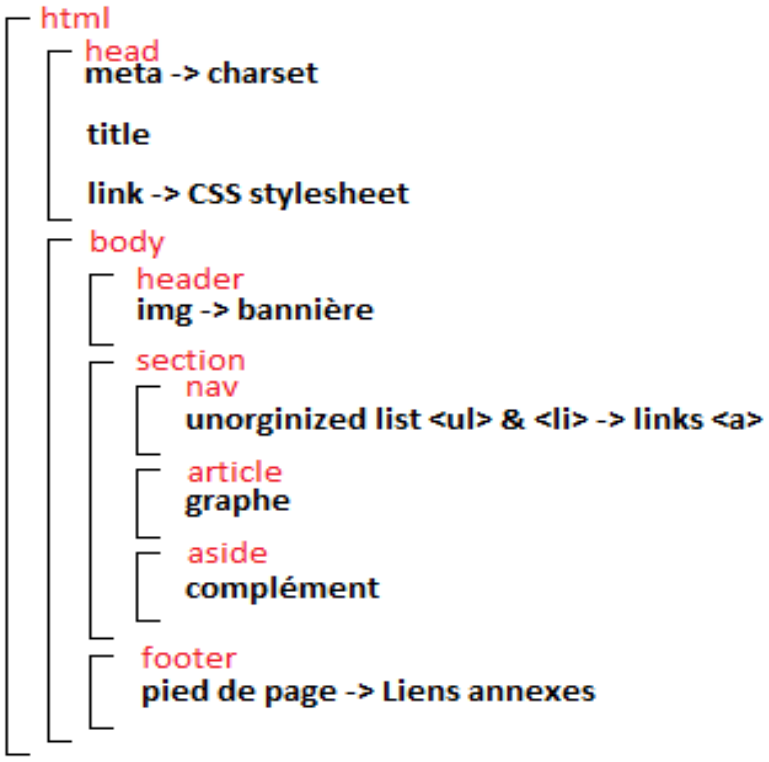
```
try
{
        $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
        $bdd = new PDO('mysql:host=localhost;dbname=proj', 'user', 'pass');
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}
$req = $bdd->prepare("INSERT datas( ?, ?, ?, ?, ?, ?");
$req->execute(array(    $directionDuVent
                        $hygrometrie
                        $pluviometrie
                        $pressionAtmospherique
                        $temperature
                        $vitesseDuVent
                        ));
```

IV – AFFICHAGE

Afin de garantir un accès simple à un utilisateur non informaticien, le support retenu a été celui d'un site web.
En effet, ce support permet d'une part la portabilité du système sur tous types de systèmes d'exploitations mais offre d'autre part la possibilité d'ouvrir l'accès à ces données à un réseau plus ou moins ouvert dont les restrictions d'accès sont définies par l'utilisateur.

A / Langages utilisés :

- HTML5 : La page web est sémantiquement découpée et présentée grâce au HTML. On retrouvera tout le long du site une architecture tout à fait typique du HTML5 de construction du code illustrée sur la figure suivante :



- CSS3 : Pour la mise en page du site web, j'ai utilisé le langage CSS dans sa version la plus récente.

- PHP5 : PHP est un langage de script particulièrement adapté au milieu du web. C'est grâce à lui que j'arrive à sortir les informations de la base de données pour les afficher sur la page web. Il est fortement ressemblant au langage C. Le code PHP s'exécute côté serveur, ainsi, le client n'a aucune trace de ce code là quoi qu'il fasse.

- SQL : Acronyme de « Structured Query Language », le SQL permet de manipuler des bases de données. Dans le cas présent j'ai utilisé une base de données de type MySQL. Le code SQL est exécuté via le script PHP.

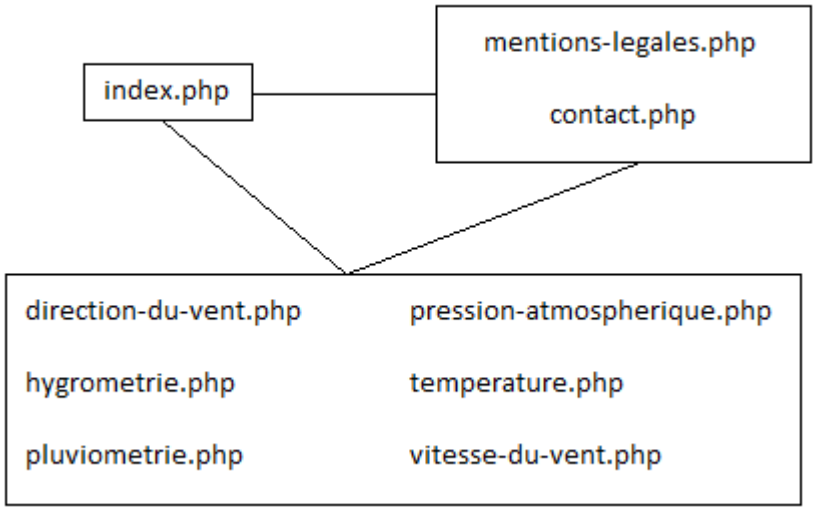
- Javascript avec la librairie jQuery : jQuery va permettre de mettre en place les indicateurs de manière dynamique. jQuery se charge de l'animation des graphes de la page.

- Ajax : Utilisé avec jQuery, Ajax va permettre à jQuery de questionner l'API[1] PHP pour recevoir les données à intégrer dans les graphes et indicateurs. C'est Ajax qui va permettre le chargement de ces données dynamiquement, sans avoir à recharger la page web pour constater un changement.

B / Architecture du site web :

D'un point de vue interne au serveur, on pourra observer l'architecture ci-dessous à gauche. D'un point de vue externe, utilisateur, on préférera la modéliser selon la figure de droite.

```
www/
    css/
        index.css
        other.css
    js/
        jQueryUI/
            UIObjects.js
        dataLoader.js
        jquery.min.js
    images/
        photos/
        banner.png
    includes/
        footer.php
        header.php
        nav.php
        start.php
    contact.php
    direction-du-vent.php
    hygrometrie.php
    index.php
    mentions-legales.php
    pluviometrie.php
    pression-atmospherique.php
    temperature.php
    vitesse-du-vent.php
```



C / API PHP

L'API PHP est constituée d'un fichier PHP. C'est ce script qui va être interrogé pour obtenir les données de la station météo. L'algorithme de l'API est le suivant :

Si la grandeur demandée n'est pas définie
fin
Sinon
demander à la base les mesures de la grandeur demandée les plus récentes
demander à la base les moyennes de la grandeur demandée
envoyer les données récupérées au format JSON
fin

[1] API = Application Programming Interface

D / Widgets jQuery

Pour le site j'ai utilisé deux différents widgets. D'une part, je me suis servi de jqxchart.js pour les graphes et d'autre part de jqxgauge.js pour les indicateurs de mesure instantanée.
Tous les widgets ont été téléchargés sur http://www.jqwidgets.com
J'ai utilisé plusieurs finalités graphiques différentes pour un même widget afin d'avoir de la diversité visuelle sur le site web.

E / Rafraîchissement des données

Au chargement de la page ainsi que toutes les 10 minutes, une fonction jQuery va interroger l'API PHP grâce à Ajax afin de rafraîchir les données en plus de commencer par les demander au chargement de la page. Le système est géré par un système de timer.

Conclusion

Malgré le fait que la station ne soit pas encore fonctionnelle à la fin des délais, le projet fut extrêmement enrichissant. Il nous a appris à travailler avec des contraintes techniques, temporelles et économiques tout en nous apprenant énormément de concepts électroniques et physiques.

Le projet a été pour moi l'occasion de vraiment découvrir ce qu'est le travail en équipe. Nous avons construits notre projet de façon participative et n'avons pas hésité à nous aider les uns les autres tout en n'oubliant pas notre objectif final.

Le projet a surtout été synonyme d'un apprentissage immense. Si nous devions refaire cette station, il nous faudrait beaucoup moins de temps que ce que nous avons pris pour fabriquer ce prototype.

Grâce à un professeur à l'écoute et très investi, ce projet, malgré le fait qu'il ne soit pas fini, peut être considéré comme une réussite au moins d'un point de vue pédagogique.

Si tout ceci était à refaire j'accepterais sans hésiter.