

REALISATION D'UN SERVEUR METEO



Elèves : MAROLLEAU Nicolas - ROGER Olivier	Professeurs : Mr Vermeulen - Mr Moutoussamy
BTS : 2005 - 2006	Lycée : Gustave Eiffel – 21000 Dijon
Projet : Station météo Oregon WMR928N	

SOMMAIRE :

I-CAHIER DES CHARGES	4
I-1 PRESENTATION ET SITUATION DU PROJET DANS SON ENVIRONNEMENT	4
<i>I-1-1 Contexte de réalisation</i>	<i>4</i>
I-2 SITUATION DU PROJET	5
I-3 OBJECTIFS PROFESSIONNELS DU PROJET	5
I-4 PRESENTATION DU PROJET.....	6
<i>I-4-1 Synoptique :.....</i>	<i>6</i>
I-5 EXPRESSION DU BESOIN	7
<i>I-5-1 Cas d'utilisation</i>	<i>7</i>
I-6 MOYENS PRELIMINAIRES DISPONIBLES ET CONTRAINTES DE REALISATION	7
<i>I-6-1 Spécifications</i>	<i>7</i>
<i>I-6-2 Synoptique de l'architecture matérielle</i>	<i>9</i>
<i>I-6-3 Contraintes de l'environnement.....</i>	<i>9</i>
<i>I-6-4 Contrainte économique</i>	<i>9</i>
<i>I-6-5 Documents et moyens technologiques mis à disposition.....</i>	<i>9</i>
<i>I-6-6 Exigence qualité à respecter</i>	<i>9</i>
I-7 REPARTITION DU TRAVAIL PAR ETUDIANT	11
II- PRESENTATION DE LA STATION.....	12
II-1 UNITE PRINCIPALE.....	12
II-2 CAPTEURS.....	12
<i>II-2-1 Thermo-Hygro-Baromètre intérieur</i>	<i>12</i>
<i>II-2-2 Transmetteurs et panneaux solaires</i>	<i>12</i>
<i>II-2-3 Thermo-hygromètre extérieur.....</i>	<i>13</i>
<i>II-2-4 Girouette et anémomètre.....</i>	<i>13</i>
<i>II-2-5 Pluviomètre.....</i>	<i>13</i>
II-3 PRINCIPE DE FONCTIONNEMENT	14
<i>II-3-1 Principe général</i>	<i>14</i>
<i>II-3-2 Mise à l'heure (DCF77).....</i>	<i>15</i>
<i>II-3-3 Principe UART.....</i>	<i>17</i>
II-4 IMPLANTATION DE LA STATION	18
<i>II-4-1 Intérieur</i>	<i>18</i>
<i>II-4-2 Extérieur</i>	<i>18</i>
III- ANALYSE	20
III-1 MODELISATION GENERALE	20
<i>III-1-1 Diagramme d'implantation</i>	<i>20</i>
<i>III-1-2 Diagramme de déploiement.....</i>	<i>21</i>
III-2 MODELISATION DETAILLEE	22
<i>III-2-1 Point de vue station</i>	<i>22</i>
<i>III-2-2 Point de vue serveur</i>	<i>23</i>
<i>III-2-3 Communication Station / Serveur.....</i>	<i>23</i>
<i>III-2-4 Communication Client /Serveur</i>	<i>24</i>
IV- CHOIX EFFECTUES	25
IV-1 STRUCTURE DE LA BASE DE DONNEES	25
IV-2 CABLAGE RESEAU	25
V- PARTIE PERSONNELLE DE CHAQUE CANDIDAT	26
V-1 Mr MAROLLEAU : COMMUNICATION STATION SERVEUR.....	26
V-2 Mr ROGER : COMMUNICATION SERVEUR CLIENT.....	70
VI- TESTS COMMUNS ET MISE EN PRODUCTION	123
VI-1 TESTS EFFECTUES	123
VI-2 MISE EN PRODUCTION	123
VII- CONCLUSION.....	124

TABLE DES ILLUSTRATIONS

<i>Figure 1: Principe général.....</i>	14
<i>Figure 2 : Portée des ondes radio Extrait Science et Avenir</i>	15
<i>Figure 3 : Signal de synchronisation</i>	15
<i>Figure 4 : Trame envoyée par l'émetteur(voir annexe pour détail).....</i>	16
<i>Figure 5 : Principe d'utilisation du DCF77</i>	16
<i>Figure 6 : Principe ligne série</i>	17
<i>Figure 7 : Srialisation/Dé sérialisation</i>	17
<i>Figure 8: Implantation de la station</i>	18
<i>Figure 9 : Schéma implantation.....</i>	20
<i>Figure 10 : Diagramme de déploiement</i>	21
<i>Figure 11 : Cas d'utilisation WMR928.....</i>	22
<i>Figure 12 : Cas d'utilisation DCF77</i>	22
<i>Figure 13 : Cas d'utilisation Serveur</i>	23
<i>Figure 14 : Communication TCP/IP</i>	24

I-Cahier des charges

BTS**IRIS***Informatique et Réseaux pour l'Industrie et les Services techniques***E6 – PROJET INFORMATIQUE****Dossier de présentation et de validation du sujet de projet**

Groupement académique : Besançon-Dijon-Reims	Session : 2006
Lycée ou Centre de formation : Lycée G. Eiffel	
Ville : Dijon	
Nom du projet : Station et Serveur météo	

Récapitulatif des projets du Lycée ou du Centre de Formation :	Nb. d'étudiants concernés
Projet N°1 : Station et serveur météo.	2+2
Projet N°2 : Mini serre	3+3+3
Projet N°3 : Station d'épuration	2+2
Projet N°4 : Centre de perçage	3

I-1 Présentation et situation du projet dans son environnement

I-1-1 Contexte de réalisation

Projet proposé et suivi par :	M : J. Moutoussamy professeur M : G. Vermeulen professeur																								
Statut des étudiants	Candidats scolarisés : en temps plein X																								
Projet développé :	au lycée ou en centre de formation X																								
Si le projet est développé au lycée ou en centre de formation :	<p>Constitution de l'équipe de développement :</p> <table> <tr> <td>Etudiant</td> <td>E11</td> <td>:</td> </tr> <tr> <td>Etudiant</td> <td>E12</td> <td>:</td> </tr> <tr> <td>Etudiant</td> <td>E21</td> <td>:</td> </tr> <tr> <td>Etudiant E22 :</td> <td></td> <td>:</td> </tr> </table> <p>Entreprise partenaire : oui <input type="checkbox"/> non X</p> <table> <tr> <td>Origine</td> <td>du</td> <td>projet</td> <td>:</td> </tr> <tr> <td>- idée :</td> <td>lycée X</td> <td>entreprise <input type="checkbox"/></td> <td>:</td> </tr> <tr> <td>- cahier des charges :</td> <td>lycée X</td> <td>entreprise <input type="checkbox"/></td> <td>:</td> </tr> </table> <p>Suivi du projet : lycée X entreprise <input type="checkbox"/></p>	Etudiant	E11	:	Etudiant	E12	:	Etudiant	E21	:	Etudiant E22 :		:	Origine	du	projet	:	- idée :	lycée X	entreprise <input type="checkbox"/>	:	- cahier des charges :	lycée X	entreprise <input type="checkbox"/>	:
Etudiant	E11	:																							
Etudiant	E12	:																							
Etudiant	E21	:																							
Etudiant E22 :		:																							
Origine	du	projet	:																						
- idée :	lycée X	entreprise <input type="checkbox"/>	:																						
- cahier des charges :	lycée X	entreprise <input type="checkbox"/>	:																						

Budget alloué :	
Autres ...	

I-2 Situation du projet

Dans quelle (s) catégorie (s) de systèmes s'insère le projet à étudier :	
Moyens de production	
Services techniques.	X
Biens d'équipements	X

I-3 Objectifs professionnels du projet

Domaines d'Activités Professionnelles abordés et développés avec le projet :	(cf. le Référentiel des Activités Professionnelles)
Analyser et spécifier le système informatique à développer	X
Réaliser la conception générale et détaillée	X
Coder et réaliser	X
Tester, mettre au point et valider	X
Intégrer et interconnecter des systèmes	X
Installer, exploiter, optimiser et maintenir	X
Assurer l'évolution locale ou la rénovation d'un système informatique	X
Gérer le projet	X
Coopérer et communiquer en langue française et notamment en langue anglaise	X

I-4 Présentation du projet

Ce projet doit permettre de récupérer des informations météorologiques ,tel que la température, le vent, la pluie et la pression, à partir d'une station météorologique autonome, de les stocker, de les archiver sur un poste informatique appelé : serveur météo. Ces informations doivent pouvoir être consultées de manière textuelle et graphique à partir d'un poste client utilisant un navigateur standard. L'accès au serveur météo est possible soit à partir d'un réseau local (Intranet), soit à partir d'Internet.

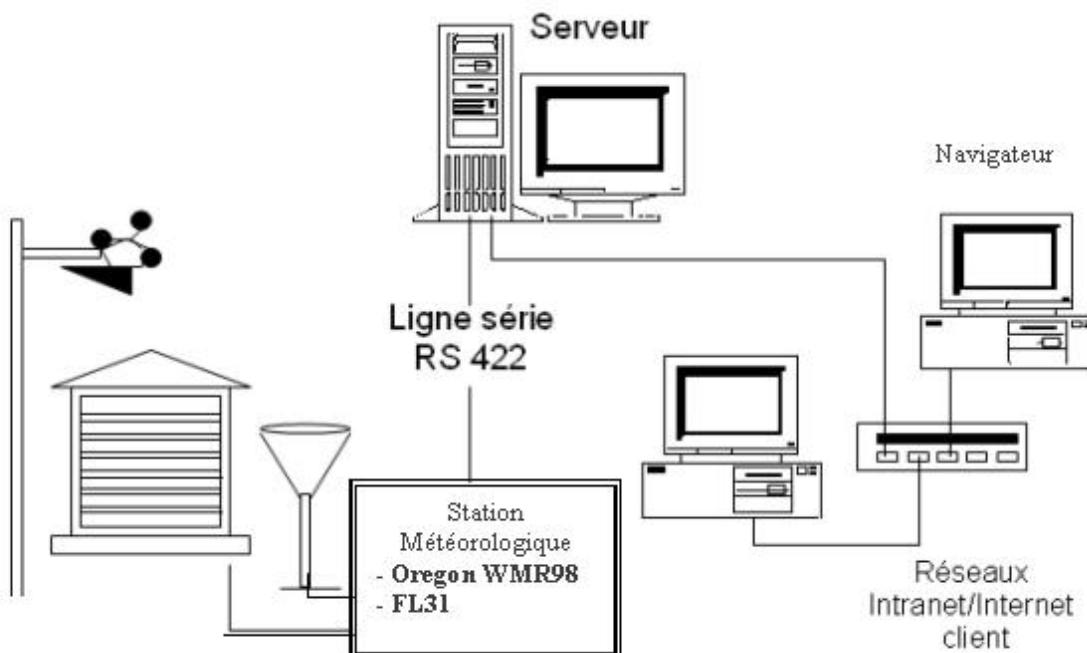
Deux versions seront réalisées :

- **Version Champollion** : dans le cadre d'un partenariat avec le collège Champollion (classe de Troisième), nous sommes chargés de réaliser cette application, la station météorologique Oregon WMR98 est utilisée par ce collège.

La station se comporte comme un serveur de données.

- Version Eiffel : la station météorologique est réalisée à partir de modules développés au lycée par la section électronique.

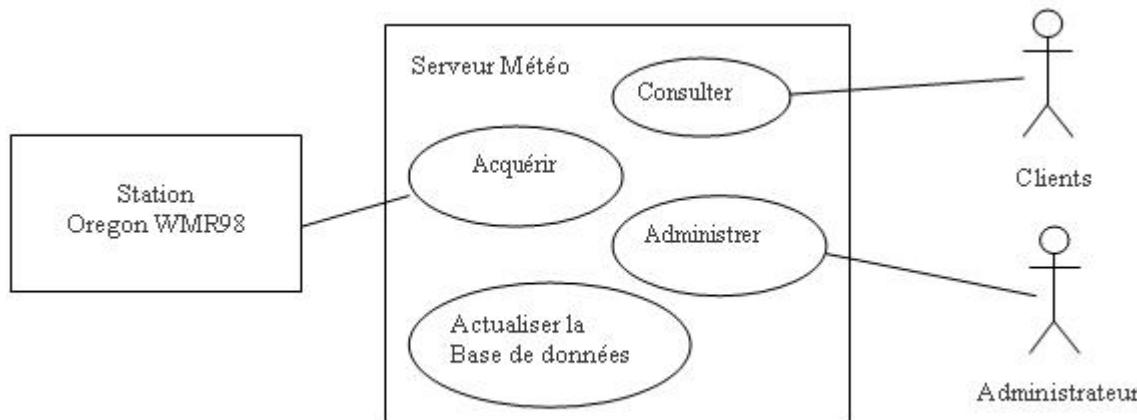
I-4-1 Synoptique :



I-5 Expression du besoin

I-5-1 Cas d'utilisation

Diagramme des cas d'utilisation du système serveur météo Champollion :



La station météo Champollion (Oregon WMR98) possède un module horloge radio synchronisée.

La station météo Eiffel possède une mémoire horodateur (time-keeper 48T08), qu'il est souhaitable de remettre à l'heure système du serveur météo chaque jour. La mise à l'heure du serveur météo Eiffel sera assurée par le module externe Radio Time Date DCF.

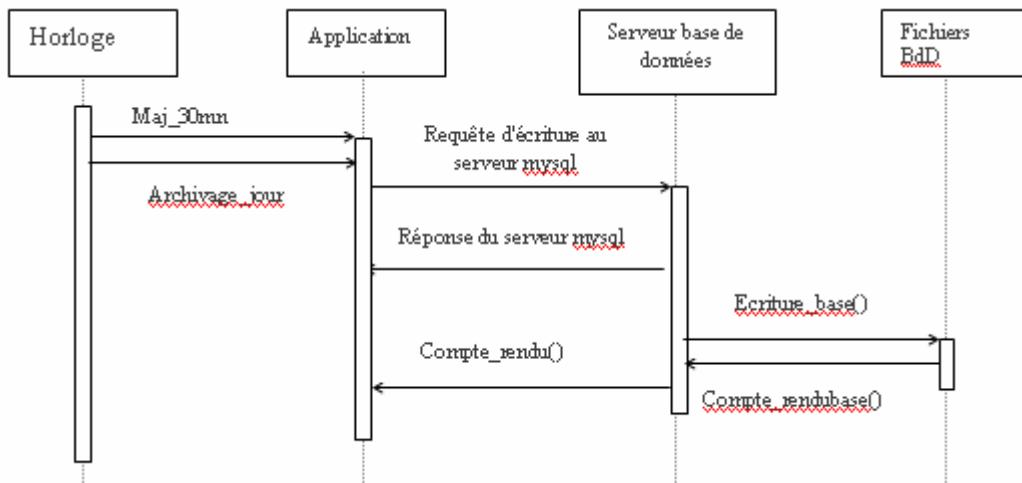
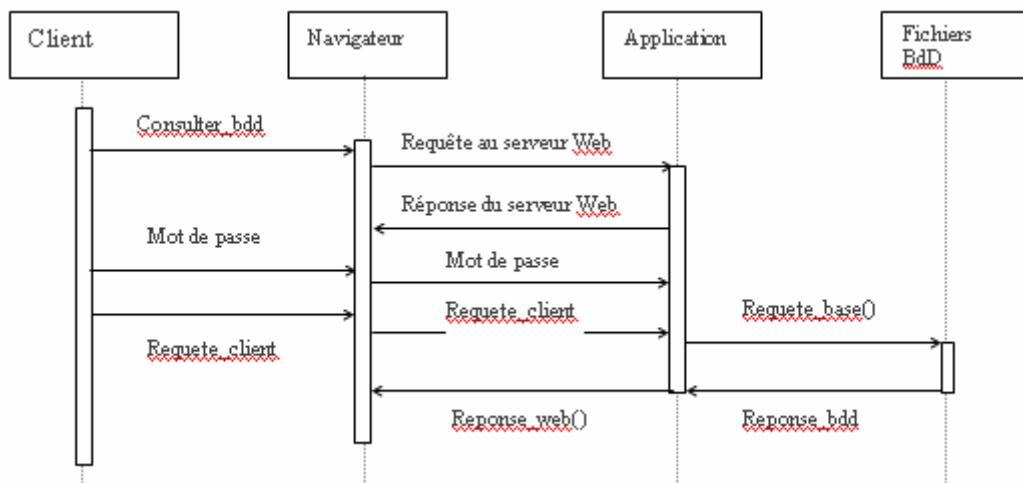
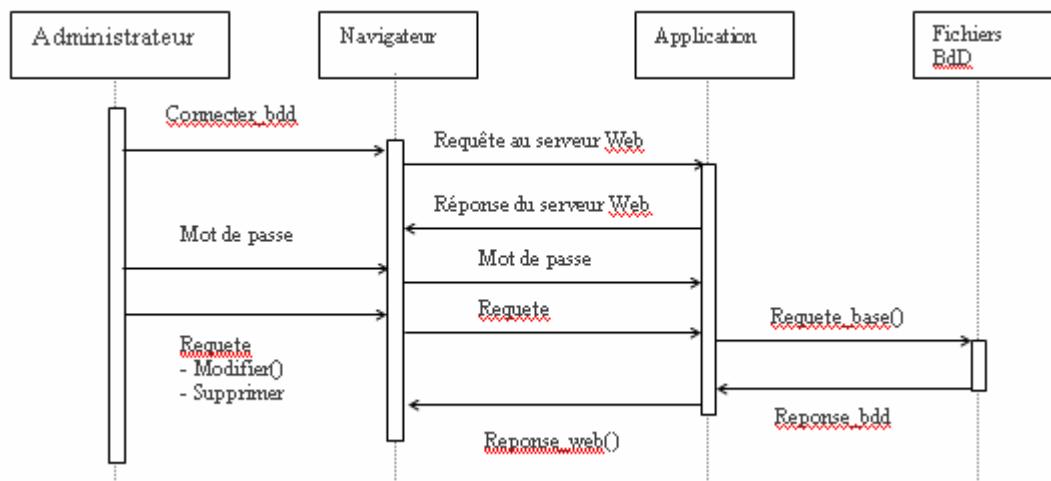
Les clients doivent pouvoir visualiser le résultat de leurs recherches quel que soit le navigateur utilisé : Internet Explorer 6.0, Netscape 7.0, Firefox.

I-6 Moyens préliminaires disponibles et contraintes de réalisation

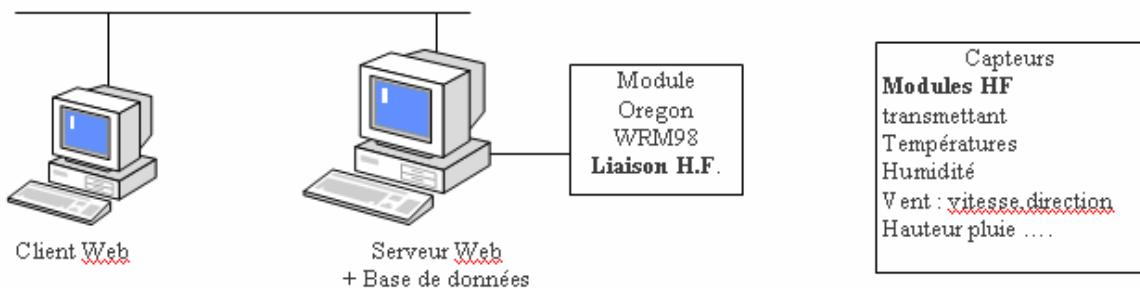
I-6-1 Spécifications

Pour la station météo Champollion (Oregon WMR98) :

- La station se comporte comme un générateur de données autonome.
- La vitesse de transfert, le format des trames voir documentation.

Serveur météo : Scénario Actualiser base de données**Serveur météo : Scénario Consulter la base de données****Serveur météo : Scénario Administrer base de données**

I-6-2 Synoptique de l'architecture matérielle



I-6-3 Contraintes de l'environnement

- Système d'exploitation : Windows 2000 pro - 2000 server
- Chaîne de développement pour la station météo Eiffel : Ezpro31 ; JSIM.
- Environnement de développement : Visual C++
- Environnement de développement pour les pages dynamiques Dev PHP ou autre.
- Wamp5 : Apache, Mysql pour la base de données, PHP 5.0 pour la gestion des pages dynamiques.

I-6-4 Contrainte économique

Assurer la mise en œuvre du système sans dépasser le budget alloué.

I-6-5 Documents et moyens technologiques mis à disposition

Documentation de la station Micrelec

Système météo WMR98

Documentation météo France.

Micro ordinateurs dont un possédant deux voies série.

I-6-6 Exigence qualité à respecter

- **Exigences qualité sur le produit à réaliser**

<i>Facteurs liés à l'environnement d'exploitation et d'utilisation</i>	
<i>Facteur</i>	<i>Signification</i>
• Efficacité	Optimisation de l'utilisation des ressources
• Maniabilité	Facilité d'emploi pour l'utilisateur (interface / homme machine sous la forme de fenêtres d'affichage et de boîtes de dialogue, choix de la langue...)
• Robustesse	Conservation d'un fonctionnement conforme aux besoins exprimés, en présence d'événements non prévus ou non souhaités (arrêt normal, intempestif ou d'urgence)
<i>Facteurs liés à l'environnement de maintenance et de suivi</i>	
<i>Facteur</i>	<i>Signification</i>

• Adaptabilité	Facilité de suppression, d'évolution de fonctionnalités existantes, ou d'ajout de nouvelles fonctionnalités
• Maintenabilité	Facilité de localisation et de correction des erreurs résiduelles
• Portabilité	Minimisation des répercussions d'un changement d'environnement logiciel et matériel

- **Exigences qualité sur le développement**

Modélisation pour la spécification : UML.

Architecture du logiciel.

Type de langage de codage : Asm 8051, C, PHP5, HTLM 4.0, Mysql.

Choix du gestionnaire d'applications pour la chaîne de production des exécutables :

EZPRO31, VisualC.

Choix du standard pour la réalisation des interfaces matérielles

Respect des normes de représentation en vigueur.

- **Exigences qualité sur la documentation**

Les exigences de qualité à respecter, relativement aux documents, sont :

Sur leur forme : respect de normes et de standards de représentation, maniabilité, homogénéité, lisibilité, maintenabilité ;

Sur leur fond : complétude, cohérence, précision, suivi des modifications.

- **Exigences qualité sur la livraison**

Produits à mettre à la disposition du client sous forme papier et informatique :

- Un seul dossier technique pour le projet, comprenant les spécifications communes et, pour chaque étudiant, les spécifications individuelles, la conception détaillée, les tests, etc.,
- Les documentations diverses soit les manuels de mise en œuvre et d'utilisation, les annexes, les codes sources, les exécutables, les interfaces matérielles, etc.).

- **Exigences qualité sur l'environnement d'exploitation**

Le serveur météo est placé dans une salle spécialisée.

La station météo Champollion est placée dans la salle spécialisée, les modules capteur sont placés à l'extérieur.

I-7 Répartition du travail par étudiant

Station Champollion	Fonctions à développer et tâches à effectuer
<u>Elève 1.1 :</u> M Marolleau.	S'approprier de la modélisation du système Finaliser la modélisation du système Caractériser les informations fournies par la station météo, définir la présentation des pages Web Prototypage IHM. Installation des outils de développement Réaliser une émulation de la station météo WRM98. Récupérer les données fournies par la station simulée, puis réelle. Effectuer la mise à l'heure système. Stocker les données. Enregistrer les données dans le serveur base de données. Tester et valider le stockage des données. Interconnecter la station météo avec le serveur météo. Gérer la planification Assurer la traçabilité des travaux Rédiger les documents relatifs au projet
<u>Elève 1.2 :</u> M Roger	S'approprier de la modélisation du système Finaliser la modélisation du système Caractériser les informations fournies par la station météo , définir la présentation des pages Web Installation des outils de développement Réaliser une base de données cohérente de simulation, correspondant à la station météo WRM98. Administre cette base de données. Assurer l'exploitation, l'affichage (graphique et textuel) des données à partir d'un poste client en respectant les critères de choix de celui-ci. Interconnecter la station météo avec le serveur météo. Administre la base de données réelle WRM98. Assurer l'exploitation, l'affichage (graphique et textuel) des données réelles WRM98 à partir d'un poste client en respectant les critères de choix de celui-ci. Gérer la planification Assurer la traçabilité des travaux Rédiger les documents relatifs au projet

II- Présentation de la station

II-1 Unité Principale

L'unité principale est une console qui permet de recevoir et d'afficher des informations de la station. Son large écran tactile rétro éclairé permet de configurer les options et de choisir les valeurs à afficher.

La station est constamment connecter à l'horloge atomique de Francfort par un signal DCF77 ce qui lui permet de rester à l'heure tout au long de l'année.

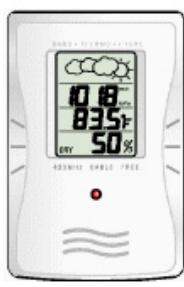
La liaison entre les capteurs et l'unité se fait grâce à un signal radio fréquence de 433Mhz. En plus de toutes les valeurs fournies par les capteurs la console affiche des prévisions météorologiques déterminées en fonction notamment des variations de pression.



II-2 Capteurs

Les capteurs sont au nombre de quatre, cependant la station peut en accueillir jusqu'à 7. Aucun câbles n'est nécessaire pour que la communication s'établisse entre les capteurs et la console d'affichage. Ils doivent simplement être placés dans un rayon de 30 à 50m de l'unité principale. La portée théorique est de 100m mais de nombreuses perturbations peuvent venir diminuer cette distance.

II-2-1 Thermo-Hygro-Baromètre intérieur

**BTHR918**

Le BTHR 918 est un capteur que l'on place en intérieur et qui permet de mesurer la température, l'humidité (hygrométrie) et la pression atmosphérique. Tout comme l'unité principale il dispose d'un écran LCD qui affiche une prévision météorologique par le biais d'un pictogramme ainsi que les derniers relevés effectués.

Les mesures sont transmises à la console principale toutes les 38 secondes qui les affichent à son tour.

Le capteur à une résolution de 1hPa (600 à 1050) en pression 0.1° C en température (-5 à 50) et 1% en humidité (2 à 98).

II-2-2 Transmetteurs et panneaux solaires

Les STR928 et STR938 sont des transmetteurs radio mais également des panneaux solaires. L'énergie générée grâce au soleil permet d'alimenter les transmetteurs ainsi que les capteurs qui leur sont associés. Deux piles de type lr6 sont placé dans chaque transmetteur afin d'assurer la continuité de l'alimentation électrique en cas de manque de luminosité. Chaque transmetteur est relié à son capteur à l'aide d'un câble de 2.4m. La transmission d'une donnée de mesure vers l'unité principale est signalée par une petite DEL rouge qui s'allume un court instant.

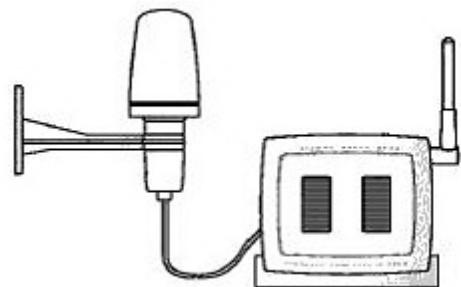
**STR938****STR928****STR928**

II-2-3 Thermo-hygromètre extérieur.

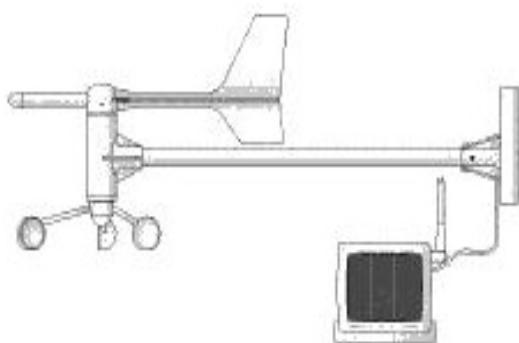
Ce capteur permet de mesurer les valeurs de température et d'humidité extérieures. Il est étanche au ruissellement ce qui permet de le placer dans des conditions difficiles. Il est important de le placer à l'ombre afin de ne pas obtenir des valeurs faussées.

Il transmet ces mesures toutes les 37 secondes à l'unité principale. Les relevés de ce capteur permettent également de calculer le point de rosée.

Le capteur à une résolution de 0.1°C (-20 à +60) en température et 1% en hygrométrie (2 à 98%).



II-2-4 Girouette et anémomètre.



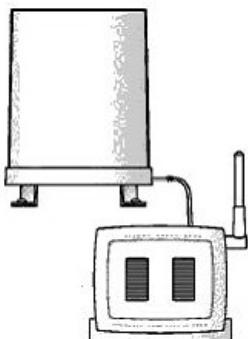
Le WGR918 est un appareil capable de mesurer à la fois la vitesse du vent et sa direction. Pour des mesures optimales il faut placer le capteur sur un mât afin de ne pas être perturbé par l'environnement proche du capteur.

L'anémomètre permet d'obtenir des mesures de vitesse instantanée, moyenne et enregistre la plus forte rafale ainsi que sa direction. Chaque vitesse est consultable en plusieurs unités (m/s ; km/h ; mph ; nœuds). La direction quant à elle est indiquée

numériquement de 1 à 359°C. Les données sont transmises toutes les 17 secondes.

La résolution du capteur est de 0.2m/s (0 à 56) pour la vitesse et de 1° pour la direction.

II-2-5 Pluviomètre



Le PCR918, dernier capteur fourni, permet de mesurer les chutes de pluie ainsi que leur intensité. Le capteur mesure le taux quotidien de chute de pluie, la hauteur des chutes de pluie de la veille ainsi qu'un cumul des chutes depuis la dernière remise à zéro. Le cumul est horodaté.

La résolution du capteur est de 1mm pour les chutes de pluie et 1mm/h pour leur intensité.

II-3 Principe de fonctionnement

II-3-1 Principe général

La station se compose de 5 composants principaux :

- Un microcontrôleur qui est le cœur du système
- Une UART pour gérer la liaison série avec le serveur
- Un Convertisseur Analogique Numérique pour convertir les données envoyées par les capteurs
- Un récepteur radio 433Mhz pour recevoir les données des capteurs
- Un récepteur radio 77Mhz DCF77 pour la mise à l'heure automatique via Francfort.

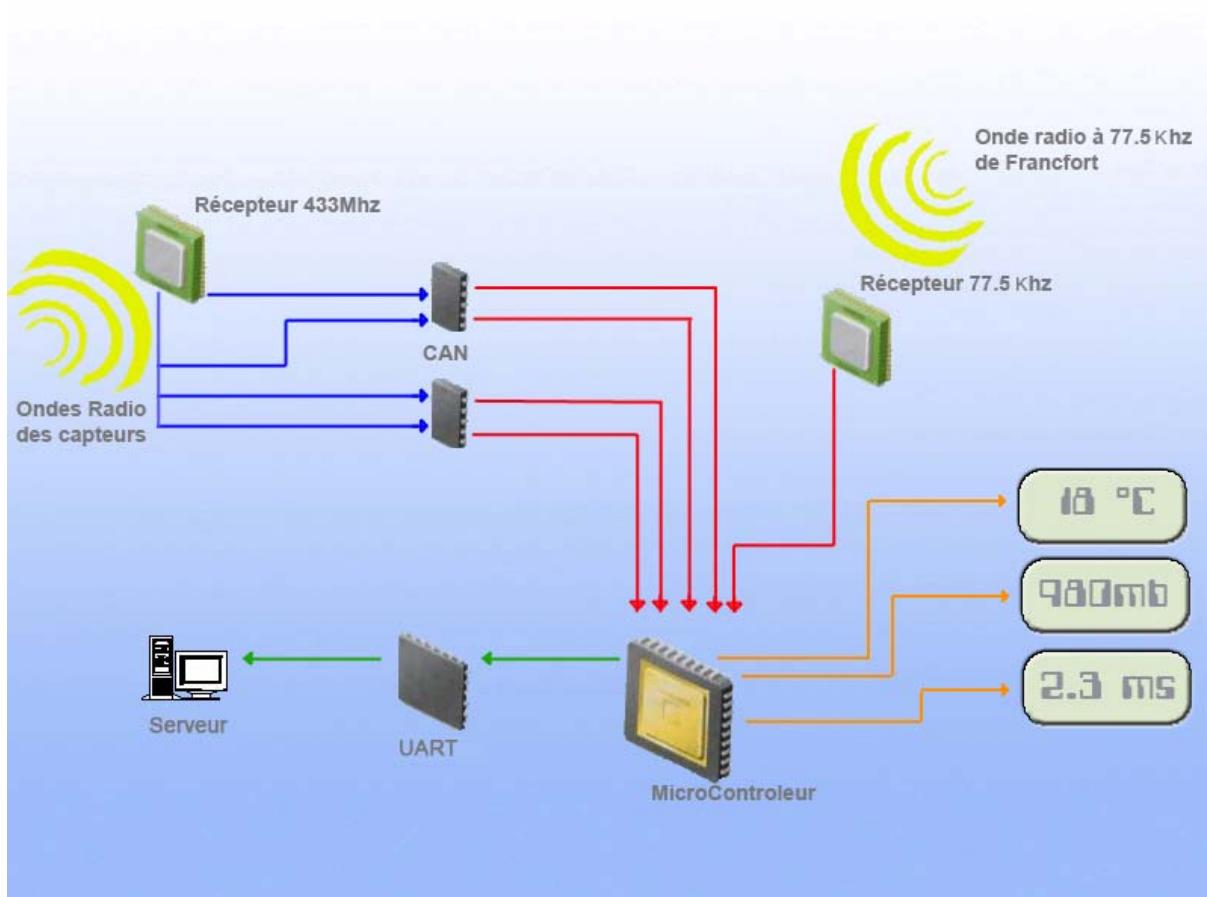


Figure 1: Principe général

Les capteurs envoient leurs données via une onde radio à 433Mhz à l'unité principale. Ces données sont de type analogiques c'est pourquoi l'utilisation d'un CAN (convertisseur analogique numérique) est indispensable pour que les données soit utilisable par le microcontrôleur qui ne travaille qu'avec des valeurs numériques.

A ces relevés s'ajoute-les donnée du DCF77 qui donne l'heure et la date exacte.

Le microcontrôleur quant à lui permet l'affichage des valeurs sur l'écran de l'unité principale mais également l'envoie des informations sur un ordinateur via l'UART et la ligne série.

II-3-2 Mise à l'heure (DCF77)

La mise à l'heure de la station est radio pilotée, c'est-à-dire que l'horloge est automatiquement réglée. Le radio pilotage se base sur un émetteur situé à Mainflingen à côté de Francfort en Allemagne. L'émetteur quant à lui est piloté par une horloge atomique ce qui garantit la précision de l'heure. Sa portée atteint une distance d'environ 2000km ce qui permet de couvrir quasi intégralement la France. La France dispose elle-même d'un tel émetteur, il appartient à France-Inter et fonctionne comme celui de Francfort seul le procédé de modulation est différent.

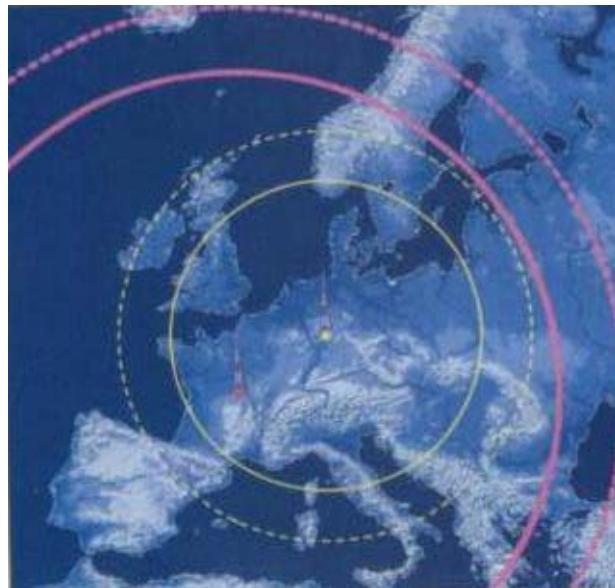


Figure 2 : Portée des ondes radio Extrait Science et Avenir

C'est grâce au récepteur DCF77 que la station capte le signal à 77.5kHz envoyé par l'émetteur et se règle automatiquement. La précision de cette horloge atteint plus ou moins 1 seconde par million d'années.

Le signal de synchronisation capté par la station météo se base sur un codage numérique très bas débit : 1bit par seconde.

Chaque bit est matérialisé par une diminution de 25% de l'amplitude du signal reçu.

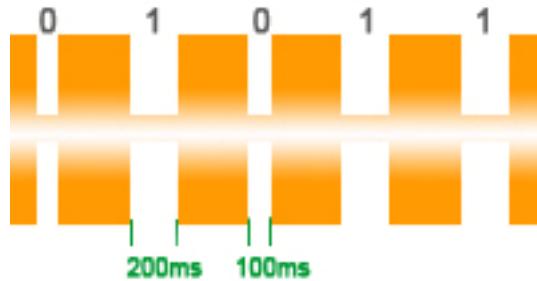


Figure 3 : Signal de synchronisation

Une impulsion de 200ms correspond à un 1 tandis qu'une impulsion de 100ms correspond à un 0.

Le message envoyé par l'émetteur comporte 59bits répartis sur 1 minute et qui se répètent constamment :

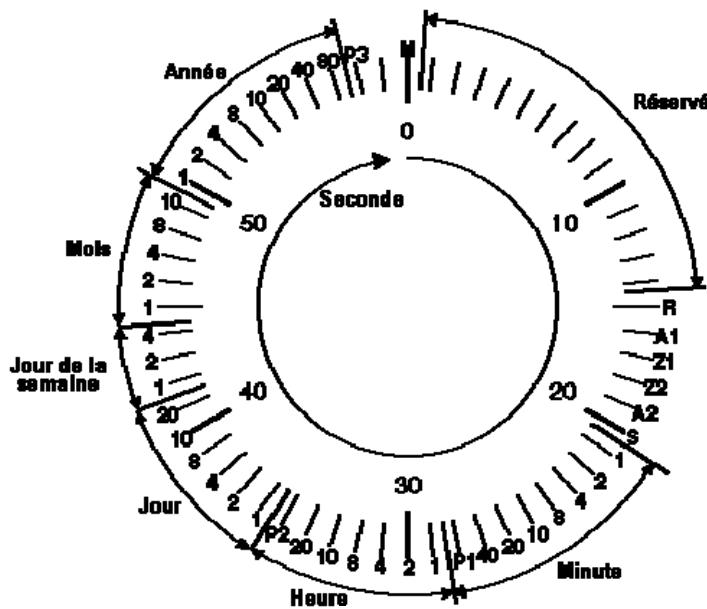


Figure 4 : Trame envoyée par l'émetteur (voir annexe pour détail)

Le début de la trame est détecté par l'absence d'un bit. Lors de ce « trou » dans la transmission le récepteur sait que les données qui vont suivre seront celle à utiliser.

Le premier groupe de bits est réservé pour des données techniques ou des options futures. Le second groupe de bits permet de différencier l'heure d'hiver et l'heure d'été ainsi que spécifier si l'émetteur est de secours est actif ou non. Les autres groupes permettent quant à eux de donner la date complète (seconde, minute, heure, jour, jour de la semaine, mois, année).

Le signal horaire DCF77 parvient donc sous forme de grandes ondes, à une fréquence de 77,5 kHz, sur une antenne puis est démodulé. On obtient alors un signal numérique, correspondant aux impulsions de 100 et 200 ms émises. Ce signal peut ensuite être traité par un décodeur afin d'obtenir les informations de date et heure :

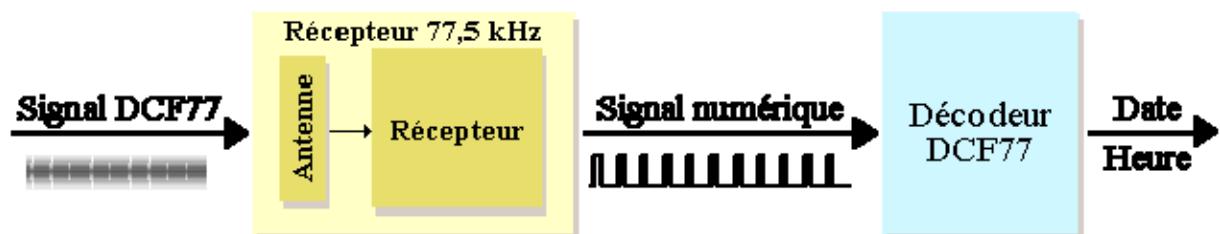


Figure 5 : Principe d'utilisation du DCF77

II-3-3 Principe UART

Dans une liaison en série, les données sont envoyées ou reçues bit par bit sur la voie de transmission. Toutefois, étant donné que la plupart des processeurs traitent les informations de façon parallèle, il s'agit de transformer des données série en parallèle ou inversement.

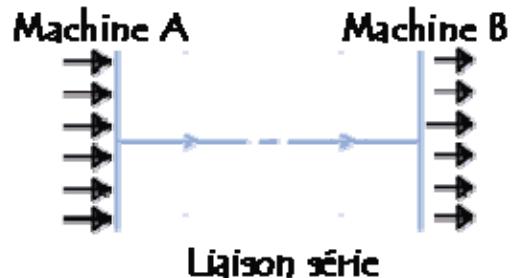


Figure 6 : Principe ligne série

Ces opérations de « transformation » série/parallèle, parallèle/série sont effectuées par une UART (*Universal Asynchronous Receiver Transmitter*).
UART est un registre à décalage. C'est-à-dire qu'il décale les données parallèles qu'il reçoit, au rythme d'une horloge afin de les sérialiser. De la même façon il décale le registre à la réception de chaque bit pour qu'une fois plein il puisse renvoyer les données en parallèles :



Figure 7 : Sérialisation/Dé sérialisation

L'unité principale est donc équipée d'une UART tout comme le PC auquel elle est reliée.

II-4 Implantation de la station

II-4-1 Intérieur

L'installation des équipements d'intérieur n'est pas normalisée mais requiert tout de même certaines attentions.

Il est préférable de placer les équipements (thermohygrobar et unité centrale) dans une pièce sèche qui n'est pas sujette aux variations de températures (véranda) et de pression. Le but étant bien entendu d'obtenir des mesures les plus représentatives possibles.

II-4-2 Extérieur

Les équipements d'extérieur quant à eux doivent être installés selon les normes MF (Météo France) et OMM (Organisation Météorologique Mondiale).

Tout d'abord le boîtier accueillant tous les capteurs sauf pluviomètre et anémomètre doit être placé de manière très précise.

Lorsqu'il est situé près d'un obstacle il doit être à une distance supérieure ou égale à 3x la hauteur de cet obstacle :

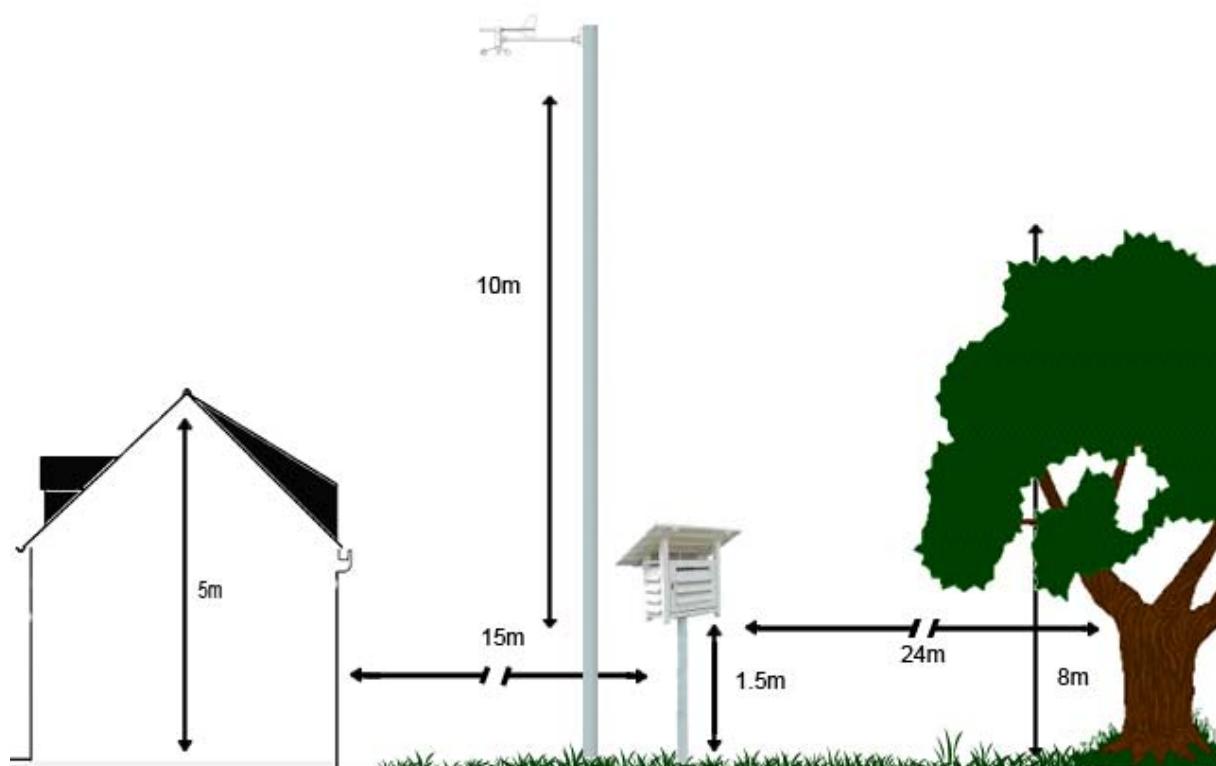


Figure 8: Implantation de la station

Comme le montre le schéma ci-dessus si on souhaite placer la station près d'une maison de 5m il faudra alors la placer à une distance minimale de 15m. De la même manière avec un arbre de 8m la station devra être située à 24m.

L'autre élément important pour le boîtier est le fait qu'il doit être positionné sur un sol de type herbeux afin d'éviter toute altération des mesures dû par exemple à un rayonnement de la chaleur.

L'anémomètre doit être placé sur un mât d'environ 10m. Le but étant de l'exposer au vent dominant et d'éviter toutes perturbations. Contrairement aux idées reçues, il ne faut surtout pas le placer au dessus d'un toit ou d'une cheminée qui génèrent énormément de perturbations.

Le pluviomètre doit être situé dans une zone dégagée comme par exemple le toit de l'abri météo.

Pour finir le boîtier doit se situer à 1.5m du sol, doit être de couleur clair (blanc de préférence) offrir une bonne ventilation et être en bois ou en matière non conductrice de chaleur.

III- Analyse

Afin de comprendre le fonctionnement complet du système il est important de le modéliser sous différentes forme plus ou moins détaillées. Cette modélisation se réalise grâce à UML :

III-1 Modélisation générale

D'après le cahier des charges on sait que l'on dispose d'une station Oregon ainsi que ces différents capteurs et d'un serveur accessible en réseau local ou sur Internet. Cet équipement peut être représenté de la façon suivante :

III-1-1 Diagramme d'implantation

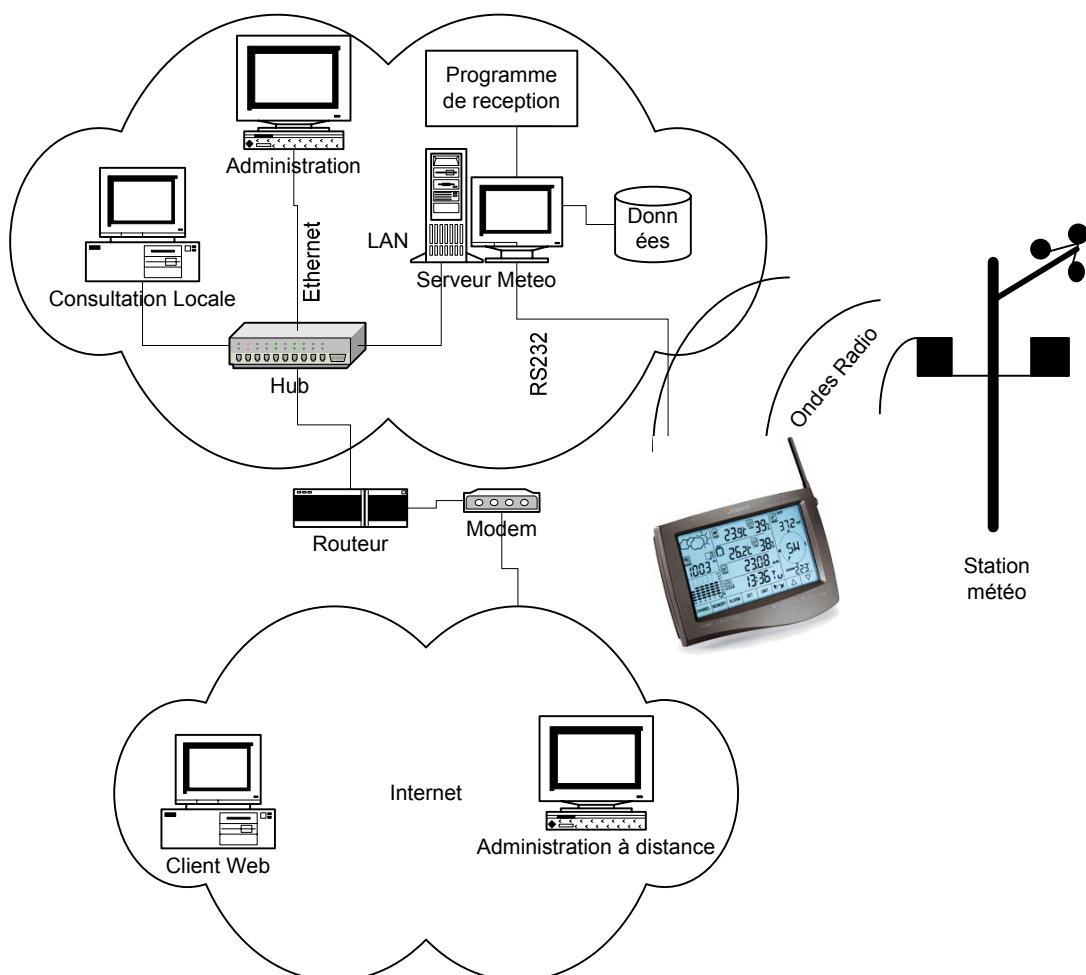


Figure 9 : Schéma implantation

On constate que le serveur météo est relié au réseau local ainsi que à l'Internet par une liaison Ethernet alors que la liaison entre le serveur et la station est de type RS232 (ligne série).

III-1-2 Diagramme de déploiement

Le diagramme de déploiement permet de compléter le diagramme d'implantation en ajoutant quelques détails tel que la multiplicité entre les différents objets ou encore les composant de chaque objet.

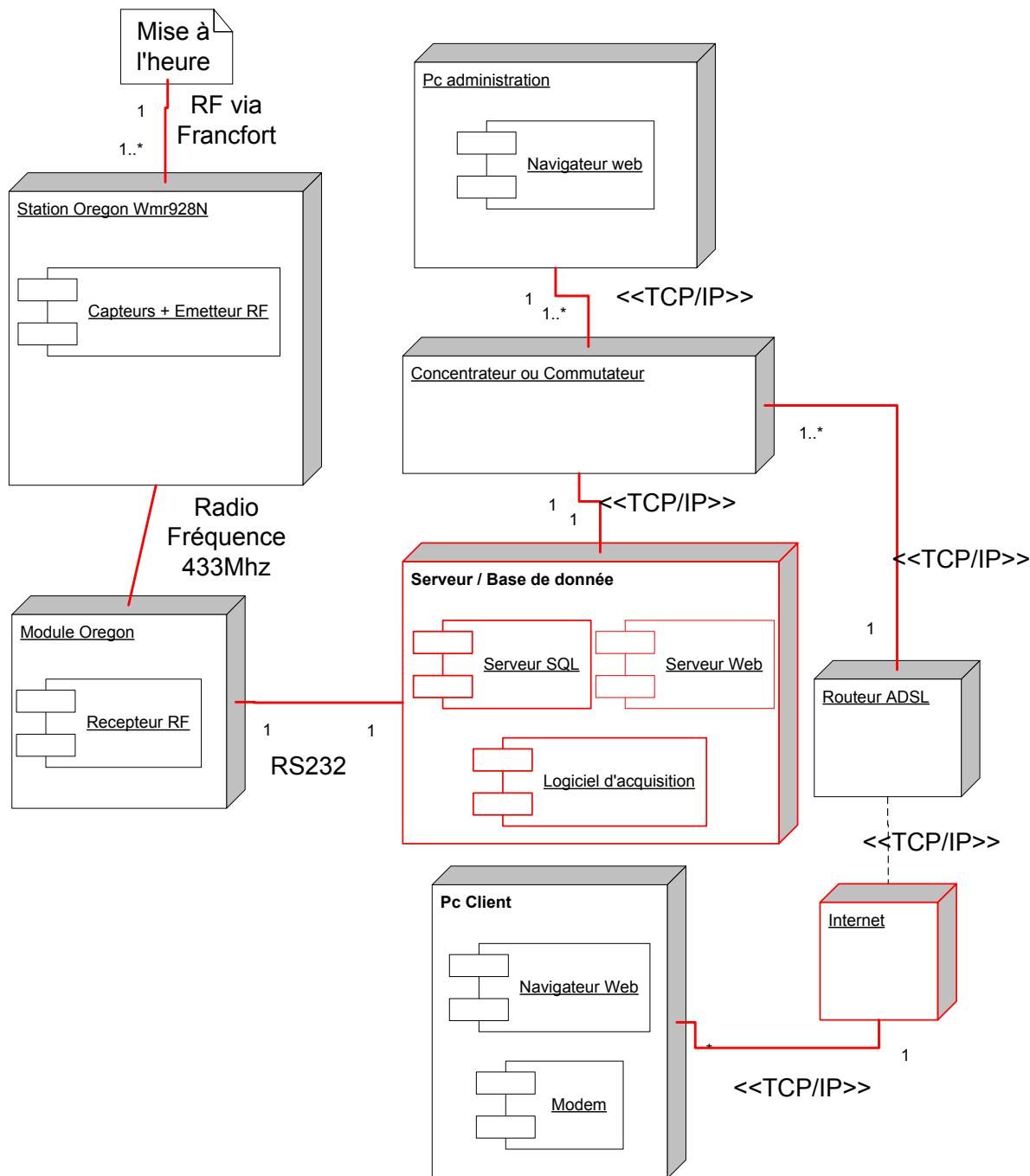


Figure 10 : Diagramme de déploiement

III-2 Modélisation détaillée

III-2-1 Point de vue station

La station météo peut être en interaction avec deux acteurs principaux : Le serveur et le client. En effet la météo peut être consultée via une IHM contenue sur le serveur ou directement sur la console principale de la météo.

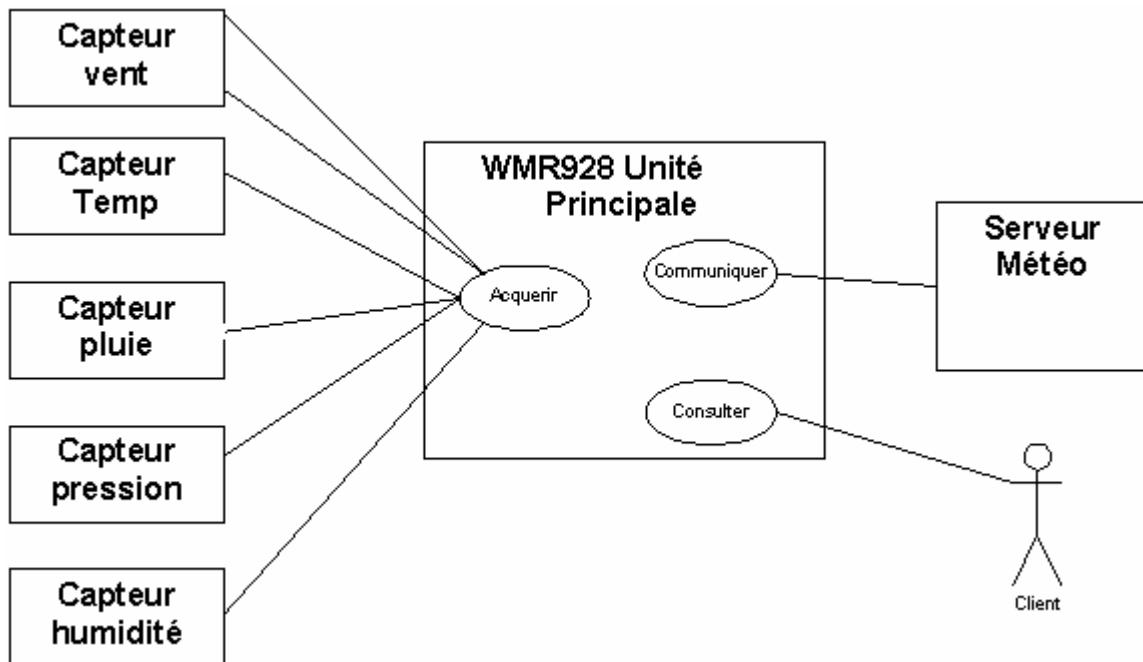


Figure 11 : Cas d'utilisation WMR928

La station permet donc l'acquisition des données mesurées par les capteurs ce qui offre la possibilité à l'utilisateur de consulter les mesures directement sur l'écran de la station. La communication qui peut être établie entre le serveur et l'unité principale permet le stockage des données ainsi qu'une consultation ultérieure.

La station peut être mise à l'heure automatiquement à l'aide du module DCF77 :

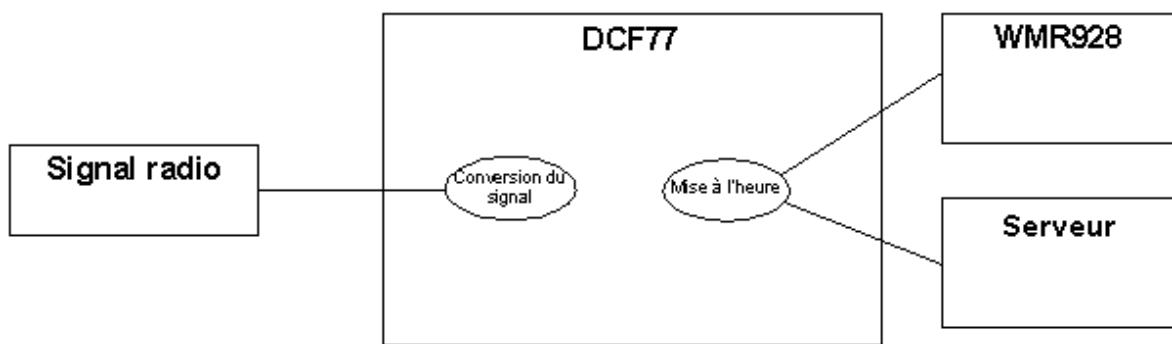


Figure 12 : Cas d'utilisation DCF77

Le module permet de capter et de convertir un signal radio envoyé depuis Francfort. Ce signal une fois convertit permet de mettre à l'heure l'unité principale de la station.

III-2-2 Point de vue serveur.

Le serveur est lié à deux acteurs : Le ou les clients et l'administrateur.

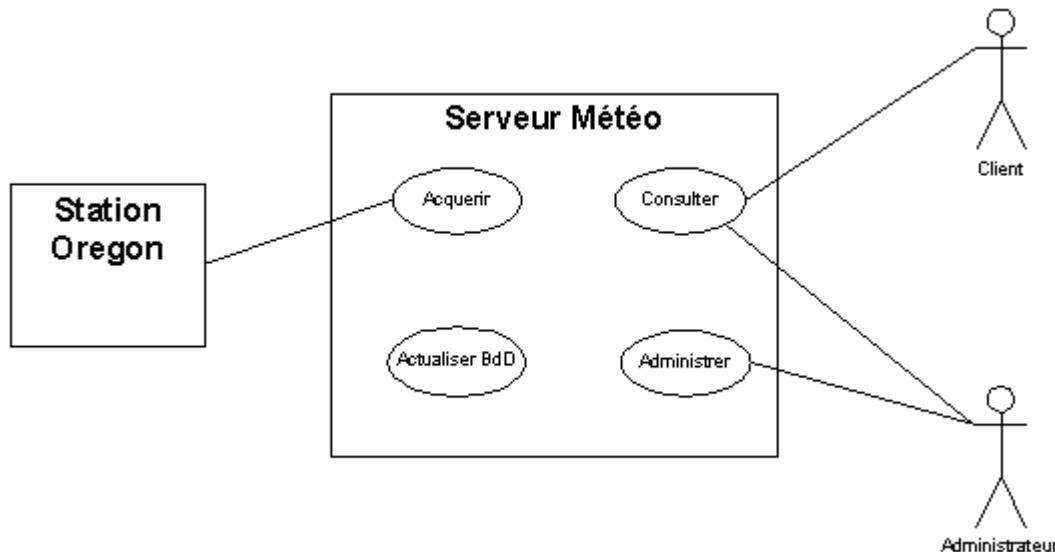


Figure 13 : Cas d'utilisation Serveur

Le serveur permet donc d'acquérir les valeurs envoyées par l'unité principale. Cette acquisition tout comme l'actualisation de la base de données est effectuée grâce au logiciel d'acquisition.

La consultation et l'administration se font via l'interface Web.

III-2-3 Communication Station / Serveur

Avant le début de la conception il est important de connaître le mode de communication de la station.

D'après la documentation Oregon Scientific la station ne fait que envoyer des données en continue et ce sans attendre d'acquittement. Le programme de réception n'aura que pour seul fonction l'écoute. On ne signalera pas à la station que les données sont erronées par exemple. De plus la communication via le port RS232 n'est possible que si la station est reliée au secteur via son transformateur. *Pour plus d'information sur la communication station serveur se reporter au paragraphe « 2.5 » du rapport de Mr Marolleau*

III-2-4 Communication Client /Serveur

La communication entre le client et le serveur s'effectue via le protocole TCP/IP et peut être modélisée comme ceci :

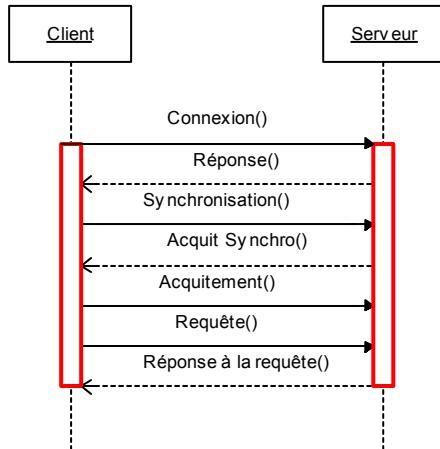


Figure 14 : Communication TCP/IP

On peut donc voir qu'à chaque page que le client voudra consulter il enverra une requête de connexion au serveur. Nous devrons donc disposer d'un système pouvant gérer les connexions multiples afin que les données soit accessibles par tous. *Pour plus d'information sur la communication client serveur se reporter au paragraphe « II-4 » du rapport de Mr Roger.*

IV- Choix effectués

Le point d'interconnexion entre les deux parties du projet est la base de données. Le protocole d'envoi des données de la station Oregon est imposé il s'agit du RS232. De ce fait le programme devra s'adapter à ce protocole.

Le site Internet étant indépendant de la station météo, il était important de choisir la structure de la base de données afin qu'elle convienne au deux parties du projet.

IV-1 Structure de la base de données

- La base de données doit permettre d'accueillir toutes les données de la station. Ces données ont été caractérisées grâce à la documentation technique fournie avec le matériel.
- C'est pourquoi nous avons décidé que la base de données devrait contenir :
- La date complète (année, mois, jour, heure, minute).
- Les températures (extérieur, intérieur)
- L'humidité (extérieur, intérieur)
- La température de rosée (extérieur, intérieur)
- La vitesse du vent
- La direction du vent
- Les données de pluviométrie (totale, hier, date)
- La prévision
- L'état des batteries
- Les erreurs éventuelles de fonctionnement

Une fois cette structure définie il faudra réaliser la base de données et dans la mesure du possible l'optimiser afin que sa taille soit minimale.

IV-2 Câblage réseau

Avant d'installer quelconque système sur les machines nous voulions pouvoir mettre nos pc en réseau sans pour autant les laisser à disposition sur le réseau de l'établissement.

C'est pourquoi nous avons choisis de tirer des câbles réseaux depuis les pc non connectés vers un Hub située dans la salle. Nous avons donc du passer plusieurs câbles dans les faux plafonds afin que ce câblage puisse être définitif et par conséquent être utilisé par d'autres.

V- Partie Personnelle de chaque candidat

V-1 Mr Marolleau : Communication station serveur

COMMUNICATION STATION SERVEUR : PROGRAMME D ACQUISITION Par Mr Marolleau



Sommaire :

1.	CAHIER DES CHARGES.....	28
1.1.	VUE D'ENSEMBLE	28
1.2.	MES TACHES A EFFECTUER.....	28
2.	ANALYSE DU SYSTEME OREGON.....	29
2.1.	MODELISATION DU SYSTEME.....	29
2.2.	DIAGRAMME DE CONTEXTE	30
2.3.	CAS D'UTILISATION STATION.....	31
2.4.	CAS D'UTILISATION SERVEUR.....	31
2.5.	LE PROTOCOLE	32
2.6.	SOUS QUELLE FORME	33
2.7.	INSTALLATION DES OUTILS DE DEVELOPPEMENT	35
3.	REALISATION D'UNE EMULATION DE LA STATION METEO	36
3.1.	PRINCIPE DU LOGICIEL D'EMULATION	36
3.2.	DIAGRAMME D'ACTIVITE DU LOGICIEL D'EMULATION	36
3.3.	MENU PROPOSE PAR LE PROGRAMME D'EMULATION	37
3.4.	EMULATION, CONCEPTION DU PROGRAMME EN C++:.....	37
3.5.	TEST DE FONCTIONNEMENT: ORPHY	41
4.	CREATION DU PROGRAMME OREGON.....	42
4.1.	PROTOTYPAGE IHM	44
4.2.	PROGRAMME DE RECEPTION ET D'ANALYSE DE TRAME.....	45
4.2.1.	<i>Le fonctionnement du programme</i>	45
4.2.2.	<i>La classe acquisition</i>	48
4.2.3.	<i>Mise à l'heure du serveur</i>	49
4.2.4.	<i>Test</i>	49
4.3.	PROGRAMME D'ECRITURE SUR LA BASE MYSQL.....	50
4.3.1.	<i>Le fonctionnement du programme</i>	50
4.3.2.	<i>La classe Sqlmeteo</i>	53
4.3.3.	<i>L'organisation</i>	53
4.3.4.	<i>Test</i>	54
4.4.	PROGRAMME FINAL	54
4.4.1.	<i>Fonctionnement général</i>	55
4.4.2.	<i>Organisation des fichiers</i>	58
4.4.3.	<i>Organisation des classes</i>	59
4.4.4.	<i>Test grandeur nature</i>	60
5.	SUPPLEMENTS.....	61
5.1.	FICHIER AIDE AVEC HTML HELP WORKSHOP	61
5.2.	INSTALLATION AUTOMATIQUE AVEC INBOX 2.0	62
6.	ANNEXES.....	64
6.1.	DIAGRAMME D'ACTIVITE GENERAL DU PROGRAMME	64
6.2.	OREGON PCLINK PROTOCOLE V 0.2	65
6.3.	TABLEAU D'ORGANISATION DES DONNEES	67
6.4.	JOURNAL DE BORD	67
7.	CONCLUSION.....	69

1. Cahier des charges

1.1. Vue d'ensemble

Mon cahier des charges consiste en la réalisation d'un programme permettant le dialogue entre la station wmr98 et le serveur de données. Cette interconnexion permettra le stockage des données fournies par la station pour une consultation future.

1.2. Mes tâches à effectuer

Durant ce projet, il me sera demandé les différentes tâches suivantes :

Analyse :

- M'approprier la modélisation du système,
- Finaliser la modélisation du système,
- Caractériser les informations fournies par la station météo,
- Faire un prototypage IHM.

Conception :

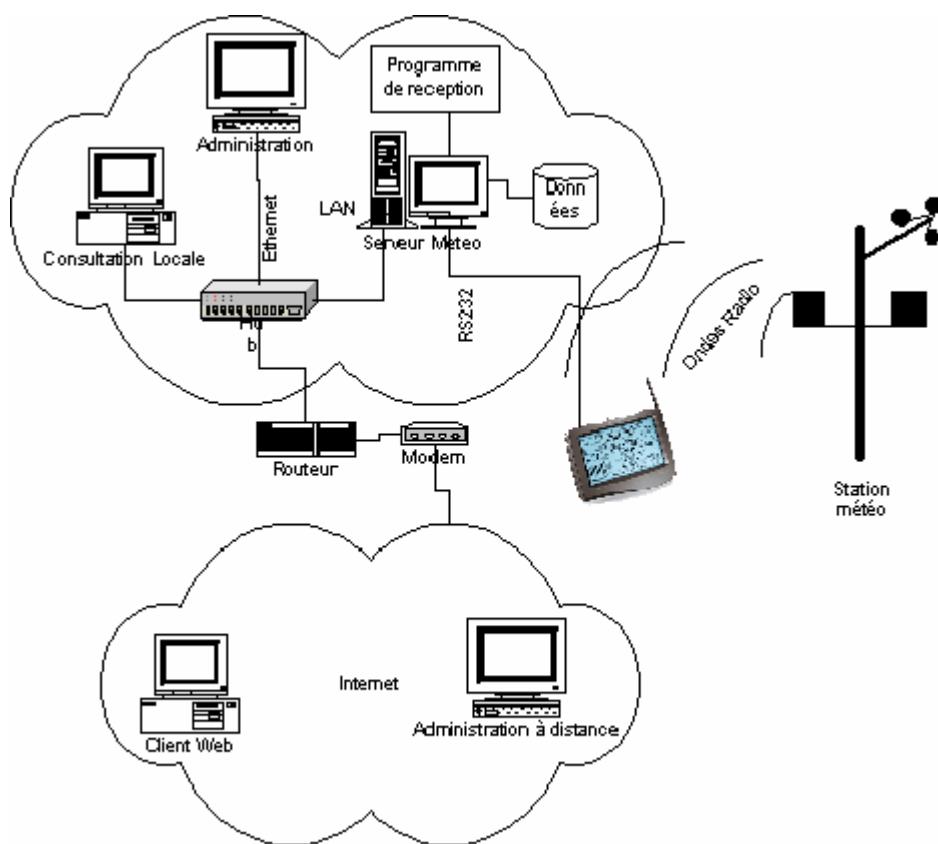
- Installation des outils de développement
- Réaliser une émulation de la station météo WRM98
- Récupérer les données fournies par la station simulée, puis réelle
- Effectuer la mise a l'heure du système
- Stocker les données
- Enregistrer les données sur le serveur météo
- Interconnecter la station météo avec le serveur météo
- Gérer la planification
- Assurer la traçabilité de mes travaux
- Rédiger les documents relatifs a mon projet

2. Analyse du système Oregon

Afin de mieux comprendre le principe de fonctionnement de la station Oregon, j'ai analysé le système via plusieurs modélisations inspirées du cahier des charges du projet.

2.1. Modélisation du système

Voici une modélisation générale du système :



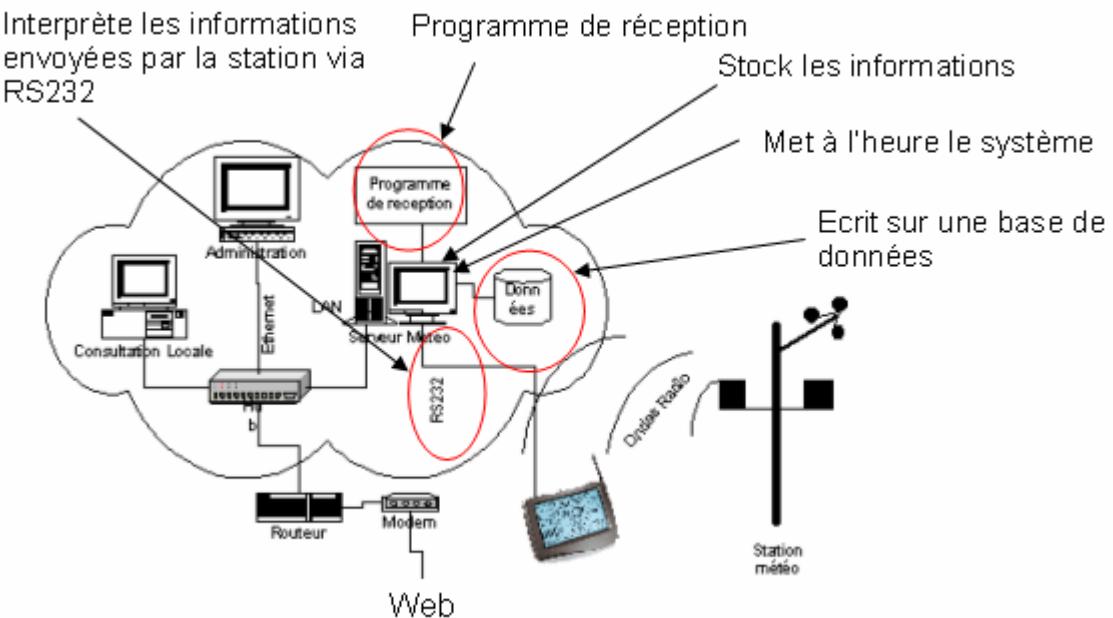
Les capteurs de la station météo envoient les informations météorologiques à l'unité principale via fréquence radio à 433Mhz.

L'unité principale réceptionne les informations, les affiche et les renvoie via RS232 sous forme de trame au serveur météo.

Celui-ci est équipé d'un programme de réception et d'analyse des informations, il analyse les données et les stocks temporairement.

Après 30 minutes, le programme ouvre une liaison avec la base Mysql.

Ensuite, il écrit les mesures dans la base Mysql, puis recommence sans cesse.

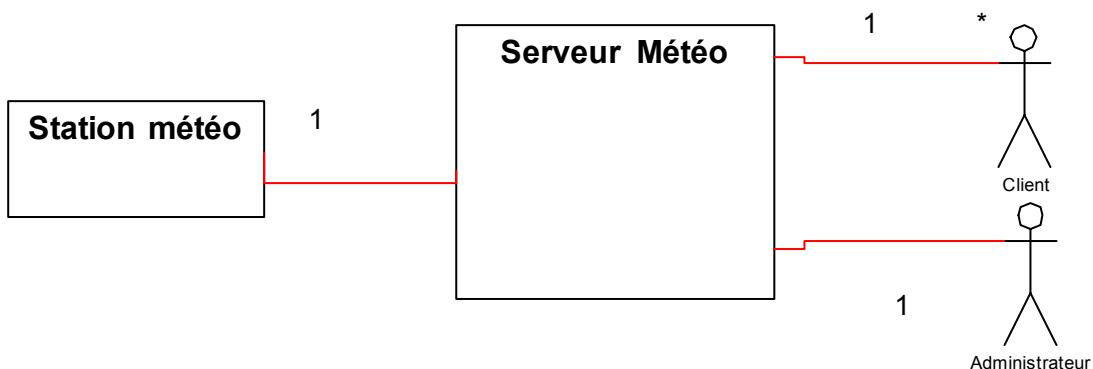


Après analyse de la modélisation du système par rapport à mon cahier des charges, je peux en déduire mon secteur d'activité ainsi que les différentes tâches que doit effectuer mon programme.

Création d'un programme de réception qui:

- Interprète les informations envoyées par la station via RS232,
- Stock les informations,
- Met à l'heure le système,
- Ecrit sur une base de données.

2.2. Diagramme de contexte



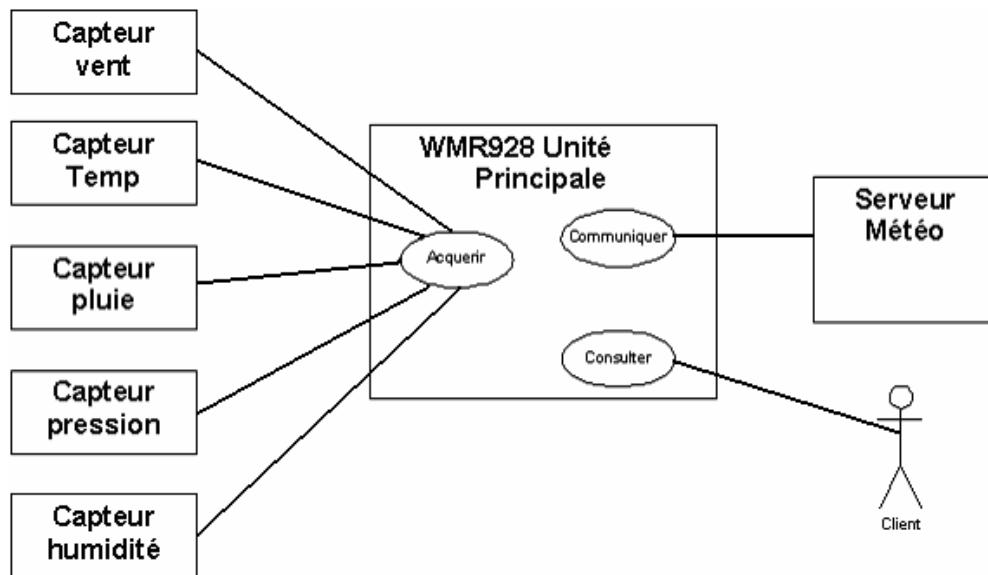
Ici, on peut voir qu'il s'agit d'une station météo dialoguant avec un serveur météo. L'administrateur est lui aussi unique. Les clients, eux sont multiples.

Ce diagramme de contexte conforte l'idée qu'il est nécessaire d'avoir :

- un support multi-connexion pour la liaison serveur météo/clients administrateur.
- un support mono-connexion pour la liaison station météo/serveur.

2.3. Cas d'utilisation station

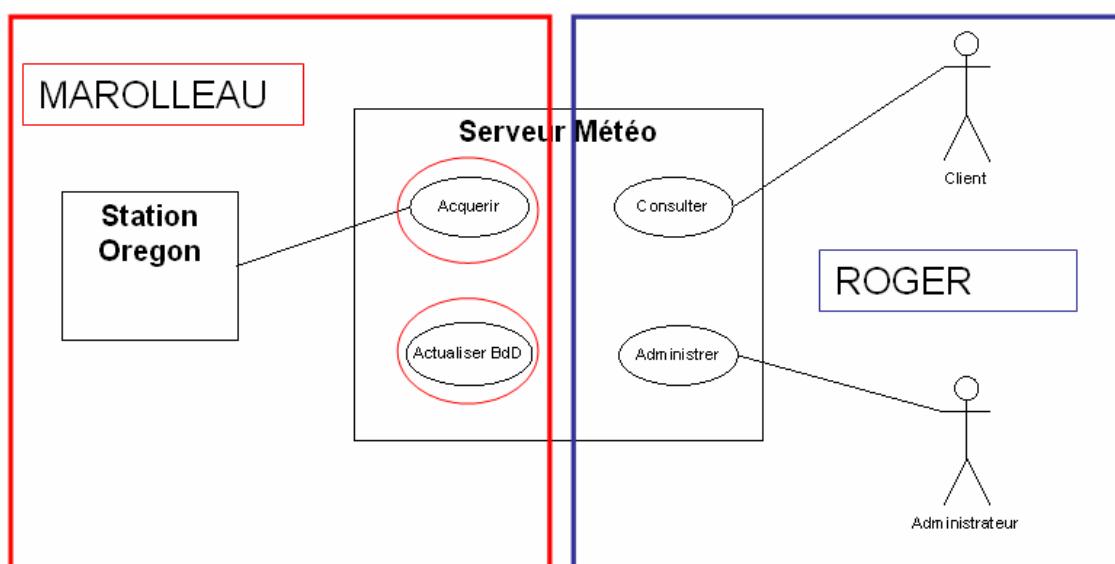
La station météo peut être en interaction avec deux acteurs principaux : le serveur et le client. En effet la météo peut être consultée via une IHM contenue sur le serveur ou directement sur la console principale de la météo.



Ce diagramme de cas d'utilisation montre que la gestion de la diversité des capteurs (donc de la diversité des liaisons) est assurée par l'unité principale.

Il existe donc une unique liaison pour communiquer entre le serveur météo et la station Oregon.

2.4. Cas d'utilisation serveur



Le diagramme de cas d'utilisation serveur permet de ce rendre compte du partage du travail. Mon rôle principal est donc d'assurer l'acquisition des mesures météorologiques et l'actualisation de la base de données.

2.5. Le protocole

La station Oregon transmet ces informations via liaison RS 232 appelé aussi le port série.



- La vitesse d'envoi des données est de 9600 bauds.
- La liaison n'a pas de parité possède 8 bits de données et 1 bit de stop.
- La station Oregon utilise la liaison RTS-CTS (Ready To Send/Clear To Send).
- Pour que le PC puisse recevoir les données de l'Oregon, la broche 'Request to send' du PC doit être activée (RTS état 1), si le cas contraire, celle-ci ne recevra rien.

Chronologie de l'acquisition :

La station Oregon possède des capteurs météorologiques qui envoient leurs données à l'unité principale d'affichage à intervalle régulier:



- Par la suite de ces envois, si l'unité principale n'est pas sollicitée, elle enverra via RS232 les données qu'elle vient de recevoir.
- Elle enverra aussi chaque minute, la donnée 'minute'.
- A toute nouvelle mesure ou à intervalle régulier, la station envoie des informations.
- Le programme d'acquisition doit être en mesure de recevoir sans cesse des données. Il sera toujours à l'écoute.
- Le programme d'acquisition doit pouvoir interpréter ces données et en valider son contenu.
- Celui-ci stockera temporairement les données dans des variables.
- Et toutes les 30 minutes, le programme écrira sur la base de données.

2.6. Sous quelle forme

La station Oregon envoie les données sous forme de trames et celles-ci ont différentes longueurs selon le type de données.

Lorsque la station Oregon envoi une trame, celle-ci envoi en premier une **tête** de reconnaissance : 2 octets égal à **0xFF**.

L'octet suivant correspondra au type d'information que la station veut nous transmettre **0x0E** correspond a la minute.

Ensuite, arrive les données météorologiques. Leurs nombres varient selon le type de la donnée.

Pour finir est envoyé le dernier octet : le Checksum qui est égal à l'octet de poids faible de la somme de la trame (entête comprise).

FF
FF
0E
XX
XX
XX
...



Oregon PCLINK protocole Version 0.2

La documentation du constructeur donne un fichier XLS informant sur la composition de la trame qui est envoyée.

La documentation Oregon PCLINK protocole V 0.2 est consultable dans les annexes.

Prenons l'exemple de la minute 36 :

Data		Minute		
Header 1	Bit 0	1	1	0xFF
	Bit 1	1	1	
	Bit 2	1	1	
	Bit 3	1	1	
	Bit 4	1	1	
	Bit 5	1	1	
	Bit 6	1	1	
	Bit 7	1	1	
Header 2	Bit 0	1	1	0xFF
	Bit 1	1	1	
	Bit 2	1	1	
	Bit 3	1	1	
	Bit 4	1	1	
	Bit 5	1	1	
	Bit 6	1	1	
	Bit 7	1	1	
3rd Byte (DEVICE) TYPE	Bit 0		0	0x0E
	Bit 1		1	
	Bit 2	00001110	1	
	Bit 3	Minute	1	
	Bit 4		0	
	Bit 5		0	
	Bit 6		0	
	Bit 7		0	
4th Byte	Bit 0		0	0x36
	Bit 1	Date 1 digit minute	6	
	Bit 2		1	
	Bit 3		1	
	Bit 4	Date 10 digit minute	3	
	Bit 5		1	
	Bit 6		0	
	Bit 7	Batt. Low	0	
5th Byte	Bit 0		0	0x42
	Bit 1		1	
	Bit 2		0	
	Bit 3		0	
	Bit 4		0	
	Bit 5		0	
	Bit 6		1	
	Bit 7		0	

Les deux premiers octets sont **0xFF** et **0xFF**.

Ici il s'agit de la minute donc du TYPE minute donc le 3ème bit est à **0x0E**.

Ensuite les 4 premiers bits du 4ème octet correspondent à l'unité de minute.

Puis, les 3 bits suivants correspondent à la dizaine de minute.

Après arrive le drapeau indicateur de l'état de batterie (0 : batterie opérationnelle, 1 : batterie faible).

Pour finir arrive le checksum. Il s'agit de l'octet de poids faible de l'addition de toutes les données de la trame.

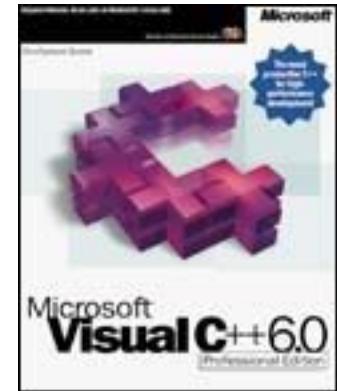
2.7. Installation des outils de développement

Visual C++ ainsi que la librairie MSDN

Entreprise : Microsoft
 Version : 6.0
 Licence : éducative
 Conception : Programme (.exe)



Oregon.exe

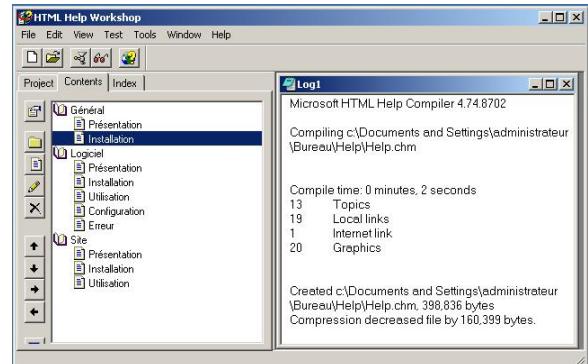


HTML Help Workshop

Entreprise : Microsoft
 Version : Version: 1.1
 Licence : gratuit
 Conception : Fichier aide (.chm)

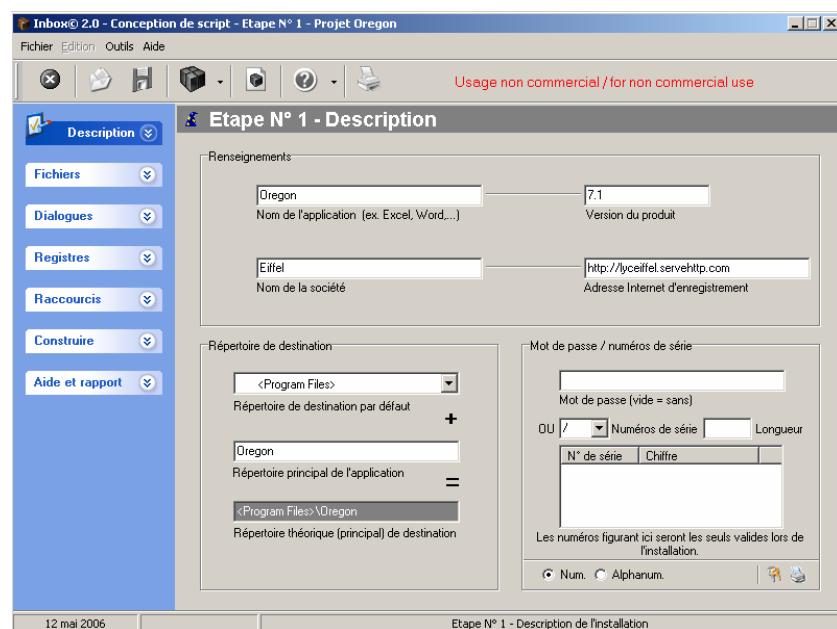


Help.chm



Inbox 2.0 Evaluation

Entreprise : Palmsoft
 Version 2.0
 Licence : Shareware 10 jours d'essai
 Conception : Fichier installation (.exe)



3. Réalisation d'une émulation de la station météo

3.1. Principe du logiciel d'émulation

Le programme d'émulation a un rôle double :

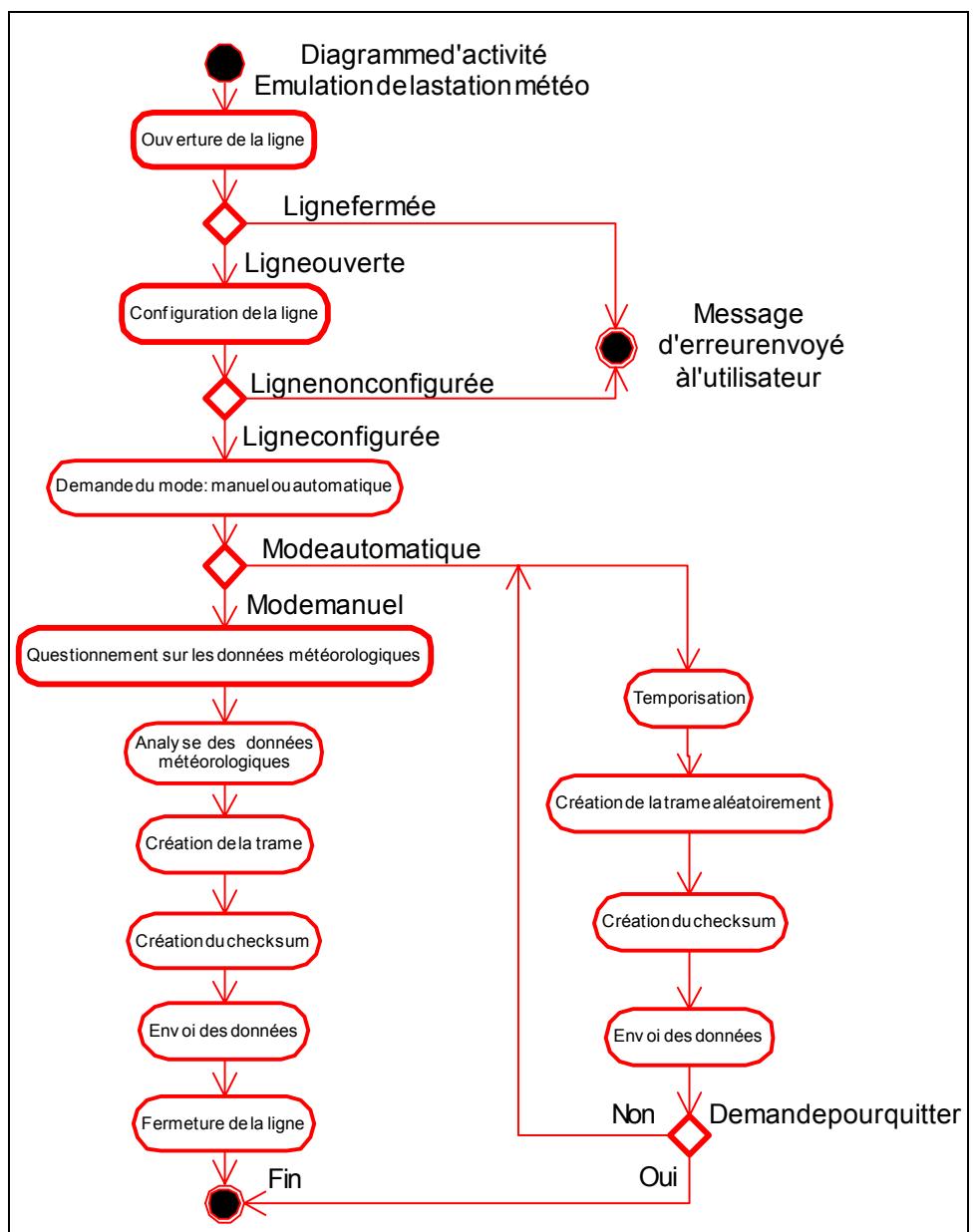
- Il me rassure et m'aide à comprendre comment les informations de la station Oregon sont envoyées.
- Il me permettra par la suite de tester mon programme d'acquisition et en vérifier réellement la bonne fonctionnalité.

Celui-ci construira des trames suivant le même modèle que le protocole Oregon PCLINK.

3.2. Diagramme d'activité du logiciel d'émulation

Ici, la gestion d'erreur n'est pas présente car il s'agit du programme d'émulation il n'est donc pas important de faire part à l'utilisateur du type d'erreur qu'il a rencontré.

La boucle autonome du mode automatique aura comme rôle d'envoyer les données météorologiques à intervalle régulier.



3.3. Menu proposé par le programme d'émulation

Le programme d'émulation propose 2 façons de tester et d'émuler la station : La première façon consiste à poser des **questions** à l'utilisateur afin de créer pas à pas sa propre trame de donnée. (1, 2, 3, 4, 5, 6 dans le menu).

Voici par exemple les questions posées pour la Minute :

```

4
Batterie faible ? 1 oui 0 non
0
signe de la temp 0=>+ 1=>-
0
temperature ?
26
virgule ?
6
humiditee ?
11
temperature de rosee signe de la valeur 0=>+ 1=>-
0
temperature de rosee ?
2
3=>pluie 2=>nuageux 6=>couvert C=>soleil ?
2
Pression atmosphérique ?
987

La trame a envoyer est la suivante:
ff ff 06 00 66 02 11 02 83 21 00 00 06 29
ff ff 06 00 66 02 11 02 83 21 00 00 06 29
Press any key to continue

```

Menu de selection
 1.Minute
 2.Vent
 3.Pluie
 4.Temp interieur
 5.Temp exterieur
 6.Clock
 7.Automatique
 Votre choix ? :

Une trame est générée en fonction des paramètres météorologiques choisis.

La seconde façon consiste à faire une simulation **automatique** qui, comme la station Oregon envoi ces informations à intervalles prédéfinies. (7 dans le menu).

Tout comme l'indique la partie **protocole 2.5** l'envoi des données sur **une minute** devra répondre à ce schémas :

17' Anémomètre
 34' Anémomètre
 37' Extérieur
 38' Intérieur
 47' Pluviomètre
 51' Anémomètre
 60' Minute

3.4. Emulation, conception du programme en C++:

Le programme comporte plusieurs fonctions qui seront exécutées une à une :

```

if(ouvre()==true)
{
    if(config()==true)
    {
        if(prepaparation()==true)
        {
            if(creachecksum()==true)
            {
                if(envoitrame()==true)
                {
                    if(fermserie()==true)
                    {

```

- Ouverture de la ligne,
- Configuration de la ligne,
- Préparation de données,
- Création d'un checksum,
- Envoi de la trame,
- Fermeture de la série.

Etant donné que le programme de simulation possède uniquement un rôle d'émulation de la station Oregon, je n'ai apporté aucune importance à la conception d'une classe.

Le programme de simulation m'a surtout éclairé sur la démarche à adopter lors d'un dialogue RS232.

La fonction ouverture permet d'ouvrir la ligne série :

```
bool ouvre()
{
    IdCom = CreateFile("COM1", GENERIC_READ|GENERIC_WRITE, 0, NULL, OPEN_EXISTING, NULL, 0);
    if (IdCom != INVALID_HANDLE_VALUE) //étape suivante
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Pointe le nom du fichier L'accès Mode de partage Sécurité Attribut(s) du fichier Template
Comment créer?

L'ouverture de la ligne Rs232 se fait via CreateFile car on lit sur la ligne série comme si on lisait dans un fichier.

La fonction configuration permet de configurer la ligne série pour qu'elle corresponde aux demandes techniques.

```
OldDcb = dcb;           // Récupère l'ancienne configuration
dcb.BaudRate = CBR_9600; // Vitesse de transmission
dcb.ByteSize = 8;        // Taille de l'info
dcb.Parity = NOPARITY;   // Parité
dcb.StopBits = ONESTOPBIT; // Nbre de bit de stop
// sans protocole
dcb.fOutxCtsFlow = 0;    // CTS output flow control
dcb.fRtsControl = 0;     // RTS flow control
dcb.fOutX = 0;           // Echange Xon/Xoff
dcb.fInX = 0;            // idem Xon/Xoff
dcb.XonLim = 0;          // permet de sortir le Xon
                        // automatiquement si le buffer est vide
dcb.XoffLim = 0;          // permet de sortir le Xoff automatiquement
```

RTS est mis à 0 car ce programme émule la station Oregon, il faudra le mettre à 1 lorsqu'il s'agira du programme de réception.

La fonction préparation de la trame fonctionne avec la variable ‘entrée’ celle-ci est définie sous forme de tableau en unsigned char car je n'utilise pas de valeur négative et chaque élément du tableau pèse 1 octet.

Cette fonction utilise aussi la variable ‘boucle’ pour connaître la longueur de la trame. Elle servira par la suite lors du Checksum et de l'envoi.

Pour reconstituer une trame adéquate, la trame doit être conforme au protocole d'envoi.

Pour cela, chaque organisation d'octet doit être respectée.

Prenons l'exemple du 4^{ème} octet de la trame Clock, ainsi que la minute 36 et la batterie faible.

	Bit 0	
4th	Bit 1	Date
Byte	Bit 2	1 digit
	Bit 3	minute
	Bit 4	Date
	Bit 5	10 digit
	Bit 6	minute
	Bit 7	Batt. Low

En décimal, la minute 36 correspond en binaire à 0010 0100.

La batterie faible correspond à 1 en binaire et décimal.

Celle-ci doit être placée en 8^{ème} position donc au bit 7, pour cela, il faut effectuer un décalage de 7.

La batterie faible correspondra à 1000 0000 en binaire.

Il suffit ensuite d'additionner les minutes avec l'état de la batterie :

```
entrée[3]=minutes+(bat_min<<7); //correspond au dernier bit de la donnée 4
```

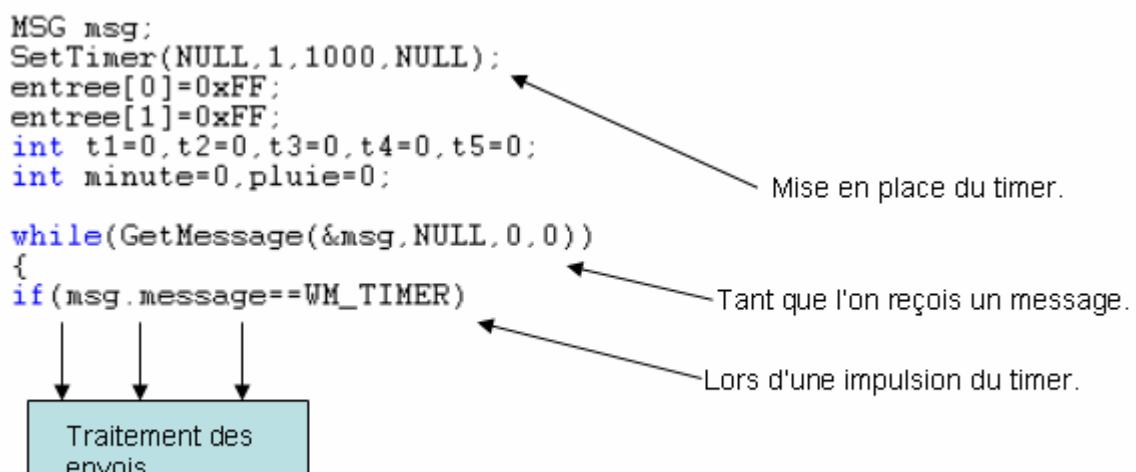
La valeur contenue dans le 4^{ème} octet de la trame Clock sera donc : 0010 0100 + 1000 000 = 1010 0100 binaire = 164 décimal.

Si la trame a été créée de façon manuelle, celle-ci envoi tout de suite la trame désirée.

	Bit 0	0
4th Byte	Bit 1	0
	Bit 2	1
	Bit 3	0
	Bit 4	0
	Bit 5	1
	Bit 6	0
	Bit 7	1

Le choix manuel permet de mettre en évidence par l'utilisateur un paramètre météorologique de la trame, par exemple, le choix manuel est pratique lorsque l'on veut obtenir une température négative ou dépassant les 99,9 degrés.

Le choix automatique permet de mettre en évidence la cadence d'envoi de la station Oregon. Celle-ci est gérée par un timer qui a comme rôle d'orchestrer les différents envois :



La fonction création du checksum a comme rôle de reprendre toutes les données, les additionnées et en sortir le checksum.

```
bool creachecksum()
{
    //checksum
    checksum=0x00;           ← Initialisation
    for(i=0;i<boucles-1;i++)
    {
        checksum=checksum+entree[i];//additionne toutes les valeurs
    }
    entree[boucles-1]=checksum&255;//prend l'octet le plus faible
    return true;
}
```

Ici, on utilise la variable ‘boucles’ qui est en fait une image du nombre d’octet dans une trame.

La boucle ‘FOR’ a comme fonction de permettre l’addition de toutes les valeurs.

Ensuite on effectue un masque de 255 sur la valeur obtenue pour n’obtenir que l’octet de poids faible.

La fonction envoi de la trame permet d’envoyer octet par octet les données météorologiques.

```
bool envoitrame()
{
    printf("\nLa trame à envoyer est la suivante:\n");

    //Ecriture sur les sorties//
    for (i=0;i<boucles;i++)
    {
        err = WriteFile(IdCom,&entree[i],1,&Nbécrits,NULL);
        if(err==false)
        {
            AfxMessageBox("écriture impossible");
        }
        else
        {
            printf("%2.2x ",entree[i]);
        }
    }
    printf("\n");
    return true;
}
```

Annotations for the WriteFile function call:

- Handle
- Pointeur de la donnée à écrire
- Nombre d'octet à écrire
- indicateur du nombre d'octets écrits
- Pointeur pour une structure d'entrée/sortie

L’envoi sur la ligne Rs232 se fait via WriteFile car on écrit sur la ligne série comme si l’on écrivait dans un fichier.

La fonction fermeture de la liaison ferme la ligne Rs232.

```
bool fermserie()
{
    bool varferm;
    if(CloseHandle(IdCom) == 0) // Fermeture de com impossible
    {
        varferm=true;
    }
    else
    {
        varferm=false;
    }
    return varferm;
}
```

Ici, on se reporte à IdCom et testons son état pour vérifier si la fermeture a bien eu lieu.

3.5. Test de fonctionnement: Orphy

La station Orphy (Non présente dans le système proposé) synthétise l'analyse de trames en RS232 et le stockage, diffusion des données météorologiques.

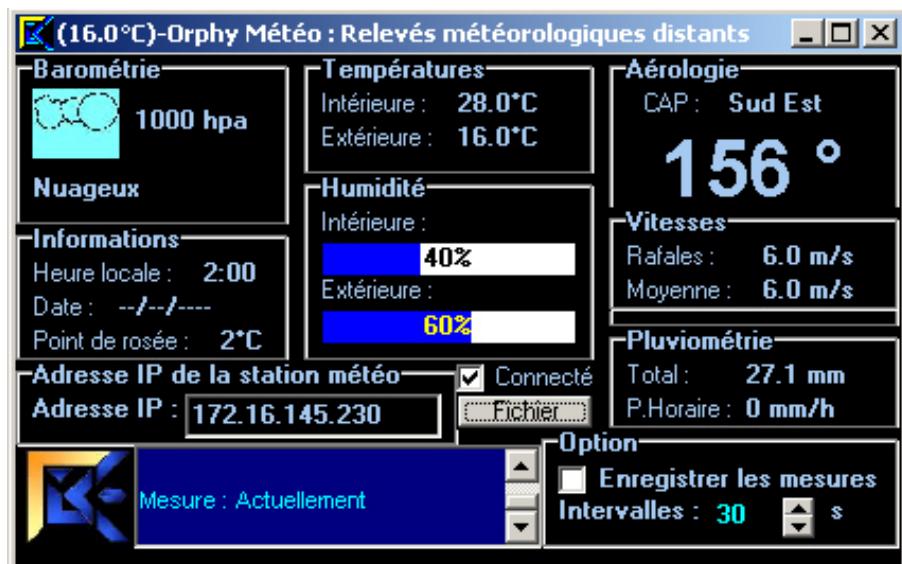
Ayant trouvé cette solution relativement ‘clé en main’, j’ai décidé d’utiliser le système Orphy pour tester mon programme d’émulation.

Pour cela, il suffit de relier un PC muni du programme d’émulation via la liaison RS232, a la place de l’unité principale Oregon.

Attention, le module Orphy ne fait pas partie du système que nous proposons.



Voici les valeurs météorologiques que j’ai pu relever après quelques minutes. (Logiciel d’émulation en mode automatique).

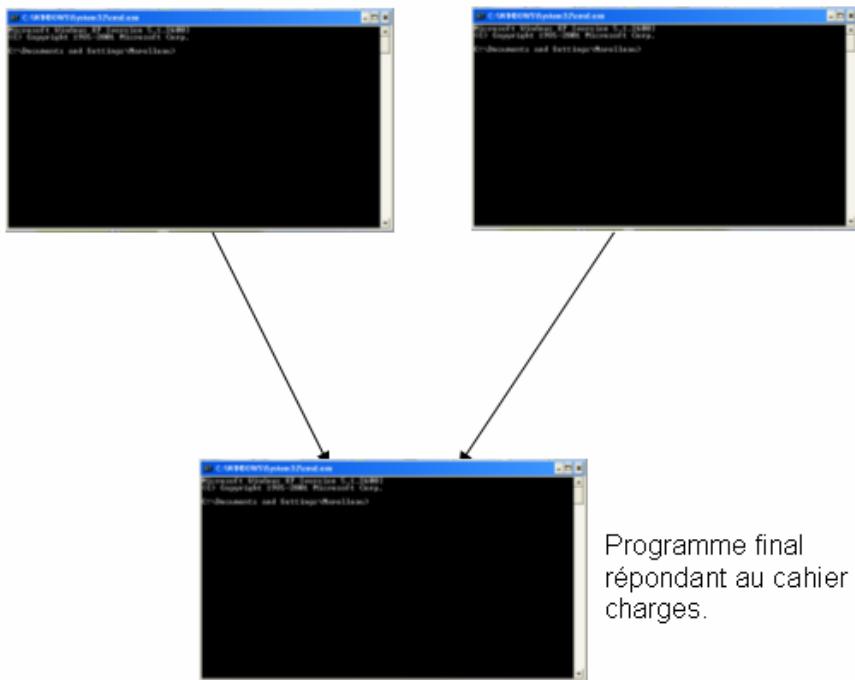


4. Création du programme Oregon

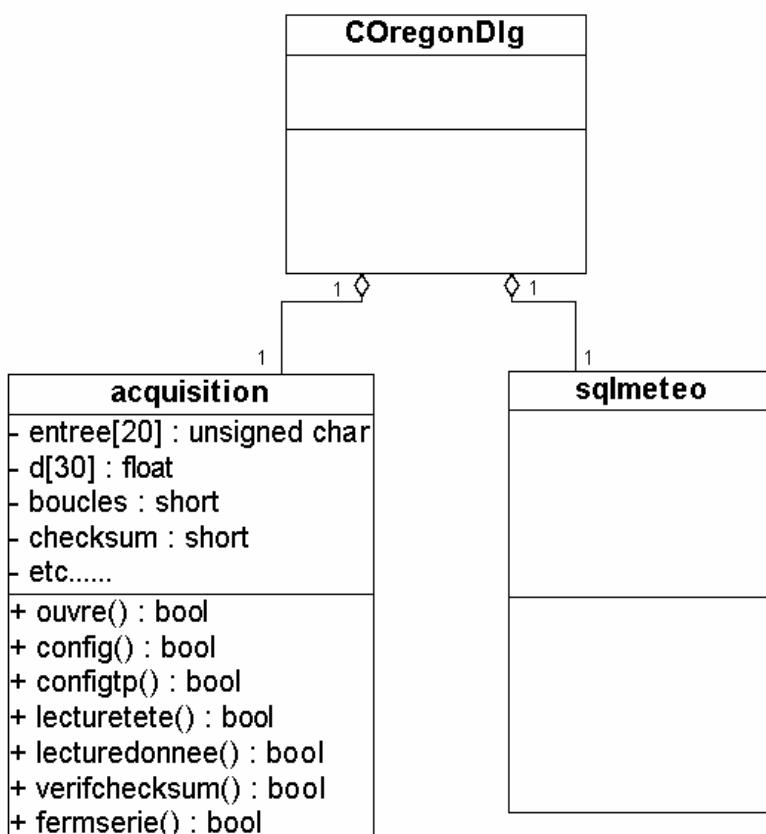
Compte tenu que le programme Oregon est relativement complexe, j'ai trouvé plus juste de décomposer mon programme en deux programmes distincts qui seront fusionnés par la suite.

Programme de réception et d'analyse de trames envoyées par la station météo.

Programme de connexion et d'écriture sur une base SQL.



La création du programme d'émulation m'a conduit à cette réflexion sur les classes : Pour concevoir mon programme, il me faudra 3 classes. Les classes acquisitions et Sqlmeteo seront prises en charge par la classe COregonDlg.

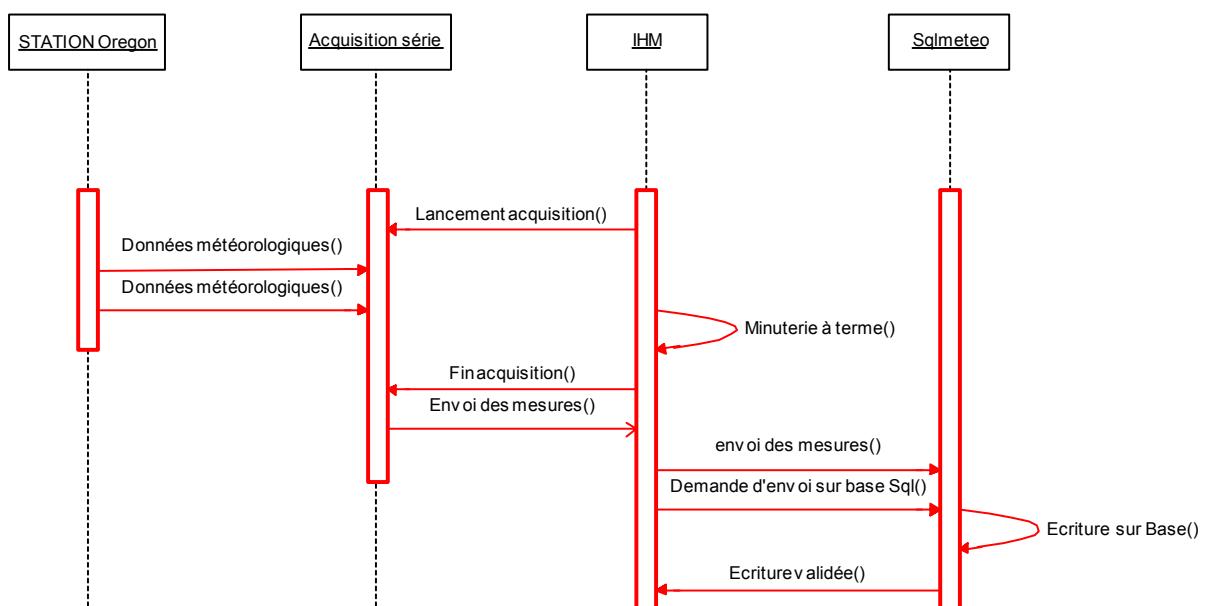


➔ La classe **COregonDlg**
Affiche les informations, gère l'IHM et contrôle les 2 autres classes.

➔ La classe **Acquisition**
Ouvre la communication entre la station météo et le programme et enregistre temporairement les mesures.

➔ La classe **Sqlmeteo**
A comme rôle d'écrire les mesures météorologiques dans la base de donnée.

Voici un diagramme de séquence représentant les différents rôles de chaque classe. Ici, on remarque bien que la classes COregonDlg (IHM) gère les dialogues.

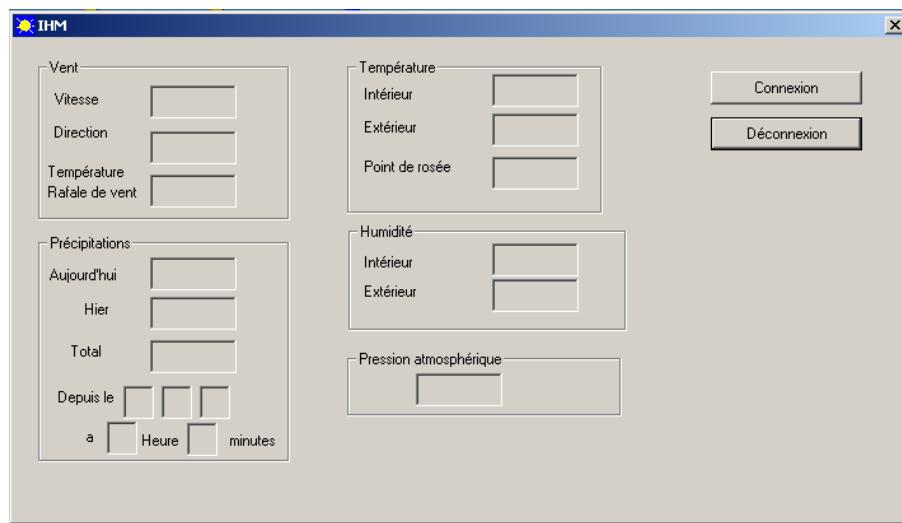


4.1. Prototypage IHM

L'IHM est une représentation graphique du programme qui s'adresse bien entendu à la personne qui mettra en service notre système. Cet IHM servira donc à diagnostiquer et visualiser l'état des liaisons et de quelques variables.

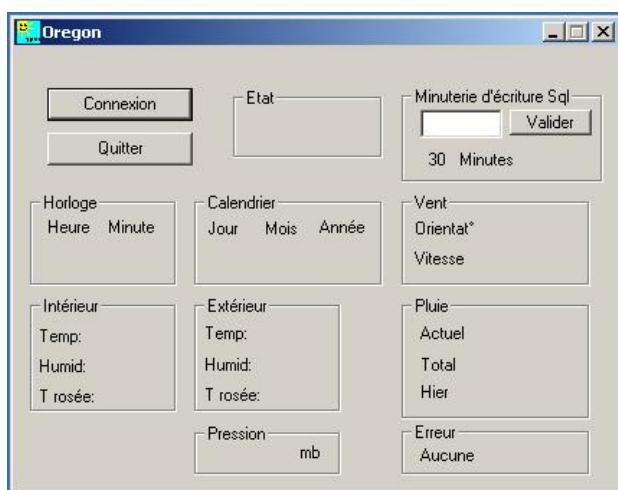
En aucun cas le programme sera utilisé pour visualiser les mesures car c'est le rôle du site internet. Cependant, il permet une confirmation des valeurs et du fonctionnement général.

Voici un premier prototype IHM du programme Oregon :



Celui-ci comporte un bouton connexion et déconnexion ainsi que les champs de valeurs : vent, température, précipitation, humidité et pression atmosphérique.

Le prototype final retenu pour le programme Oregon est le suivant :



Celui-ci est plus petit et comporte un bouton de réduction.

Pour des raisons pratiques, certains champs d'information ont été supprimés.

Le champ erreur a été ajouté ainsi que l'horloge et le calendrier.

Cet IHM permet à l'utilisateur de modifier l'intervalle d'écriture MySql, ce qui permet de voir rapidement si le système fonctionne ainsi que de faire des mesures plus précises.

4.2. Programme de réception et d'analyse de trame

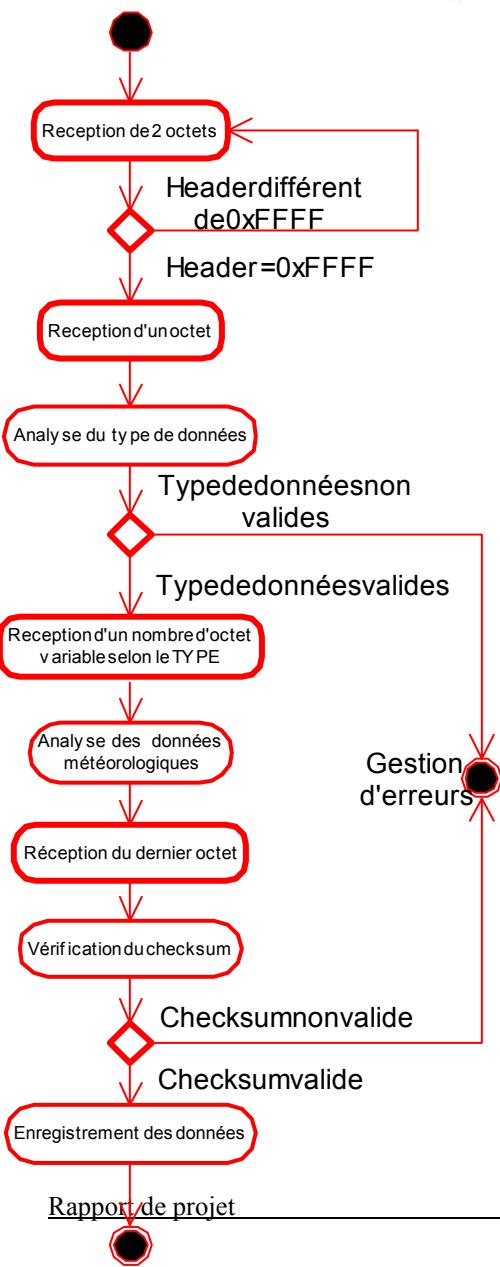
Ce programme possède une structure quasi identique que le programme d'émulation.
Attention, le programme d'émulation et le programme de réception et analyse de trame sont des programmes bien distincts.

```

if(acq.ouvre()==false)      //ouverture de la ligne
{
    if(acq.config()==false)   //paramétrage de la ligne
    {
        if(acq.configtp()==false) //configuration des temps
        {

/////////////////////////////////////////////////////////////////
do{
    if(acq.lecturetete()==false) //lecture de la tête
    {
        if(acq.lecturedonnee()==false) //lecture des données
        {
            if(acq.verifchecksum()==false) //vérification du checksum
            {
                if(acq.analysedonnee()==false) //analyse des données
                {

```



Le programme fonctionne avec la classe acquisition définie dans le fichier serie_meteo.h

4.2.1. Le fonctionnement du programme

Le programme ouvre la ligne RS232 et la paramètre comme le précise la documentation constructeur.

Puis, il écoute sans cesse pour repérer la tête correspondant à 0xFFFF.

Le programme lit l'octet suivant correspondant au type.

Ensuite, il lit les données météorologiques.
Pour finir, il lit le checksum.

Le checksum étant validé, il enregistre les valeurs météorologiques dans ces variables en vue d'une utilisation ou modification future.

Puis, le programme recommence la procédure.

Les fonctions d'ouverture, configuration de ligne, configuration des temps et fermeture de ligne sont les mêmes que celles étudiées pour le programme d'émulation.

La fonction lecturedonnee permet de lire sur la ligne RS232. Au démarrage de celle-ci, on récupère l'heure du système, cette récupération permet d'avoir l'heure sur l'interface du programme.

Au tout démarrage du programme, la variable boucle est initialisée à 100, cela permet de détecter la première lecture qui entraînera la récupération de l'heure.

```
bool acquisition::lecturedonnee()
{
    if (boucles==100) //cela signifie qu'il s'agit de la toute première lecture
    {
        //récupération de l'heure du poste
        time_t now = time (NULL);
        struct tm tm_now = *localtime (&now);

        d[18]=tm_now.tm_min;
        d[19]=tm_now.tm_hour;
        d[20]=tm_now.tm_mday;
        d[21]=(tm_now.tm_mon+1);
        d[22]=(tm_now.tm_year-100);
    }
    boucles=0; //RAZ
```

La lecture sur la ligne série tout comme pour le programme d'émulation, s'effectue via la fonction ReadFile car lire sur la ligne série équivaut à lire sur un fichier.

La variable ‘entree’ à encore été retenue car celle-ci est définie sous forme de tableau en unsigned char, chaque élément du tableau pèse 1 octet.

```
//enregistrement de la trame dans le tableau entree
for(i=3;i<boucles; i++) //sa longueur varie selon boucles
{
    err = ReadFile(IdCom,&entree[i],1,&Nblus,NULL);
    if(err==false)
    {
        AfxMessageBox("lecture impossible");
    }
    else
    {
    }
}
return true;
```

La fonction verifchecksum permet de vérifier la validité du checksum : elle lit le dernier octet de la trame avec un ReadFile puis additionne la totalité des valeurs de la trame, garde le dernier octet et compare avec le checksum.

La variable cks est initialisée lors de la détection du type de données et est utilisée pour simplifier l'addition du checksum, celle-ci est en fait l'addition des 3 premier octet reçu qui sont en fait : 0xFF + 0xFF + octet de type.

```

bool acquisition::verifchecksum()
{
    checksum=0x00;
    //lit la dernière valeur de la trame
    err = ReadFile(IdCom,&entree[boucles],1,&Nblus,NULL);
    if(err==false)
    {
        AfxMessageBox("lecture impossible");
    }
    else
    {
        //printf("Checksum :%x\n",entree[boucles]);
    }

    for(i=3;i<boucles;i++)
    {
        checksum=checksum+entree[i];
    }
    checksum=checksum+cks;
    checksum=checksum&255;//1'octet de poids faible seulement est gardé

    if (checksum!=entree[boucles])
    {
        return false;
    }
    else
    {
        return true;
    }
}

```

La fonction analysedonnee permet l'analyse et le stockage des données météorologique à l'aide de différents masques et de différentes variables.

```

case 0x0E:
{
    d_tampon_u=entree[3]&15;//récupération des unitées
    d_tampon_d=(entree[3]&112)>>4;//récupération des dizaines
    d_bat_clok=(entree[3]&128)>>7;//récupération de l'état de la batterie
    d_minute=d_tampon_u+(d_tampon_d*10);//calcul de la minute

    printf("Minutes: %d\n",d_minute);//affichage
    printf("Batterie: %d\n",d_bat_clok);

    d[18]=d_minute;      //enregistrement
    d[23]=d_bat_clok;

    return d_minute;     //enregistrement effectué
}
break;

```

Ici, la variable d est un tableau défini en unsigned char car chaque élément du tableau pèse 1 octet. Cette variable contiendra toutes les informations météorologiques.

Le tableau d'organisation des données est présent dans l'annexe.

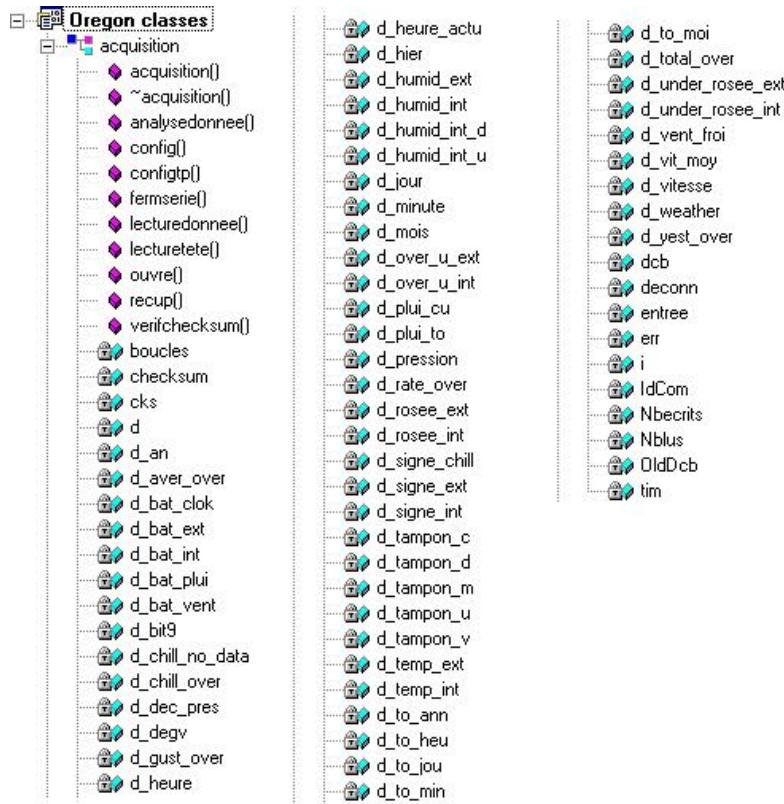
Selon le type reçu, le programme récupère à l'aide d'un masque les données et les enregistres dans le tableau **d**.

Le tableau **d** constitue un espace de stockage du programme pour les mesures météorologiques.

Le codage complet des fonctions d'analyse et réception de trame sont consultables dans le dossier technique (serie_meteo.cpp)

4.2.2. La classe acquisition

La classe acquisition possède comme fonction les fonctions vues ci-dessus et fonctionne avec différentes variables.



[Le codage de la classe acquisition est visible dans le dossier technique \(serie_meteo.h\).](#)

Les variables commençant par ‘**d_**’ correspondent aux différentes données météorologiques.

Les variables ne commençant pas par **d_** sont des variables liées au fonctionnement du programme.

4.2.3. Mise à l'heure du serveur

La mise à l'heure du serveur se fait par rapport à l'unité principale, c'est elle qui, toutes les heures envoi la trame de type 'Clock' il suffit donc de placer le code ci-dessous lors de la détection et analyse de la trame 'Clock'.

Chaque heure, donc à chaque envoi de la donnée 'Clock', le programme met à l'heure le serveur.

Le bon fonctionnement de cette fonction sera constaté par la suite lors d'un test avec le programme de simulation.

```
//mise a l'heure du pc

d_heure_actu=d_heure-2; //pour le décalage

_SYSTEMTIME st;           //création d'une structure
                           //SYSTEMTIME
GetSystemTime(&st);       //récup heure système
st.wHour=d_heure_actu;    //heure machine=heure prog
st.wMinute=d_minute;      //minute machine=minute prog
SetSystemTime(&st);       //mise a l'heure système

return d_minute;          //enregistrement effectué
```

4.2.4. Test

Après branchement de la station Oregon au Pc équipé du programme, on constate que la procédure de lecture fonctionne ainsi que la mise à l'heure.

```
"E:\Project\Nicolas\C++\Serie_meteo\Debug\serie
Valeur présente sur les entrée :99
Valeur présente sur les entrée :69
Valeur présente sur les entrée :26
Valeur présente sur les entrée :0
Valeur présente sur les entrée :0
Valeur présente sur les entrée :0
Valeur présente sur les entrée :17
Valeur présente sur les entrée :15
Valeur présente sur les entrée :19
Valeur présente sur les entrée :1
Valeur présente sur les entrée :6
Moyenne pleine: 1
Total plein: 0
Batterie: 0
Hier plein: 0
Pluie moyenne: 99
Pluie totale: 26
depuis 17M 15H 19J 1M 6A
Pluie hier: 0
Valeur présente pour le checksum :83
CA MARCHE
lecture effectue:
1 pour quitter:
```

Cette impression d'écran montre un envoi de type pluie.

4.3. Programme d'écriture sur la base MySql

La base SQL est implantée sur le serveur météo et est consultable via une page Web. Il est donc capital de pouvoir écrire dans cette base pour que la consultation puisse avoir lieu.



4.3.1. Le fonctionnement du programme

Pour pouvoir écrire ou lire dans une base SQL, il faut procéder en 3 étapes:

1) Se connecter à la base SQL

Paramètres:

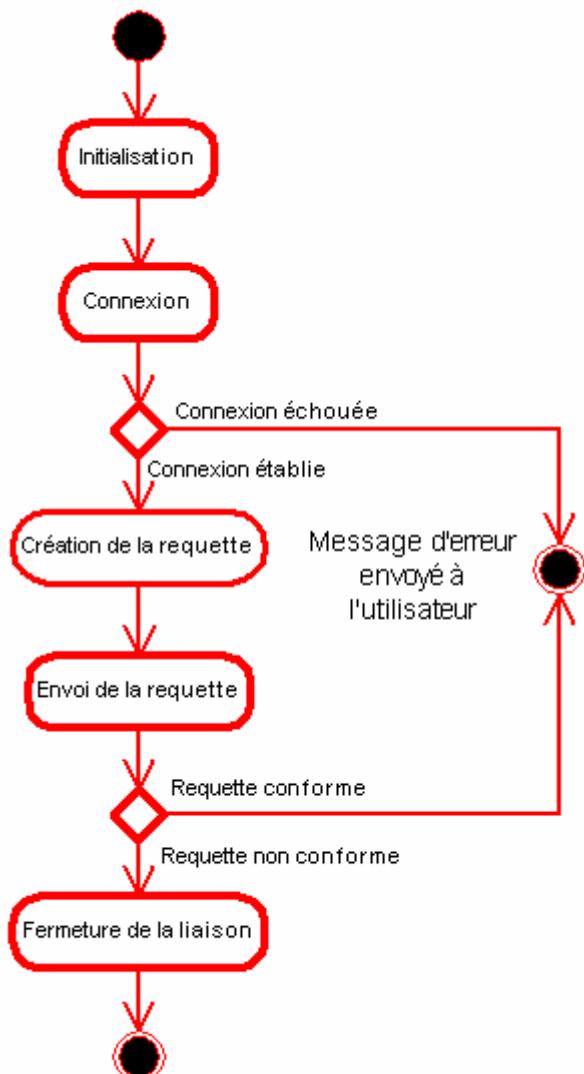
- L'adresse du serveur,
- Le port de transmission,
- Le nom de la base de donnée,
- Le nom de l'utilisateur,
- Le password de l'utilisateur.

2) Envoyer une requête

Paramètres:

- Les différents champs de la base,
- Le format de chaque champ,
- Le nom de la base.

3) Fermer la liaison



Il en émane donc la structure de lancement de fonction suivante :

```
if(sql_m.init()==true)      //initialisation de la connexion SQL
{
    if(sql_m.connexion()==true)  //connexion a la base SQL
    {
        sql_m.mesures(erreur); //récupération des mesures et envoi de la variable
        if(sql_m.requette()==true) //écriture sur la base SQL
        {
            if(sql_m.close()==true) //fermeture de la liaison SQL
            {
                if(acq.fermserie()==true) //fermeture de la ligne série
            }
        }
    }
}
```

La fonction initialisation permet d'initialiser la liaison Sql.

```
bool sqlmeteo::init() //initialisation
{
    if((conn=mysql_init(NULL))==NULL)
    {
        return false; //initialisation mauvaise
    }
    else
    {
        return true; //initialisation bonne
    }
}
```

Cette fonction utilise mysql_init pour pouvoir initialiser la liaison sql.

Ici, il n'apparaît pas les données nécessaires à la connexion car ceux-ci s'initialisent dans un fichier texte lors de la création de la classe (une explication sera données dans les pages à venir).

La fonction connexion permet la connexion à la base.

```
bool sqlmeteo::connexion() //connexion
{
    client_flag=0;
    if(!(mysql_real_connect(conn,host,user,passwd,db,port,unix_socket,client_flag)))
    {
        return false; //Connexion mauvaise
    }
    else
    {
        return true; //Connexion bonne
    }
}
```

Ici, on utilise mysql_real_connect avec comme paramètre l'adresse du serveur, le nom de l'utilisateur, le mot de passe, etc.

La fonction mesures permet la création de la requête qui sera créée avec sprintf pour pouvoir respecter par la suite le format de toutes les données.

Cette requête sera enregistrée sous forme de chaîne de caractère dans la variable query.

```
sqlmeteo::mesures() //récupération des mesures
{
    sprintf(query,"INSERT INTO mesure(-----
    return true;
}
```

A la place du trait rouge se trouvera :

Le nom des champs de la base

Le format des données

Les variables contenant les données

Prenons cet exemple simplifié contenant 4 champs :

```
"INSERT INTO mesure(id,temp_i,temp_e,pluie) VALUES('','%.f','%.f','%d');",d[12],d[8],d[6]);
```

Le premier champ est toujours id car il s'agit du numéro d'enregistrement.

Le chiffre de la température intérieur sera sous la forme de float avec un chiffre après la virgule, idem pour la température extérieur.

La pluie, elle sera enregistrée sous forme d'un chiffre entier.

La température intérieur, extérieur et la pluie correspondent respectivement aux données 12, 8 et 6 du tableau **d**, pour l'instant, le tableau **d** a été initialisé avec des valeurs fixes dans la classe Sql_météo en vue d'une fusion avec le programme d'analyse et de réception.

Pour envoyer les données météorologiques contenues dans le tableau **d**, il sera donc nécessaire de les récupérer dans la classe acquisition.

La fonction requette permet l'envoi de la requête préparée si dessus.

Ici, on utilise la fonction mysql_query pour la requête.

```
bool sqlmeteo::requete() //lancement de la requete
{
    if(mysql_query(conn, query) ) //requete
    {
        //retourne le message d'erreur pour l'appel le plus récent à une fonction de l'API qui peut réussir ou échouer
        printf("Erreur due a query: %s\n",mysql_error(conn));
        return false; //erreur
    }
    else
    {
        return true;//requete bonne
    }
}
```

mysql_error permet de visualiser la potentielle erreur possible.

La fonction close permet de fermer la liaison sql.

```
bool sqlmeteo::close() //fermeture de la liaison SQL
{
    mysql_close(conn); //fermeture de la connexion
    return true; //fermeture ok
}
```

Celle-ci utilise la fonction mysql_close.

Le codage complet des fonctions Mysql est consultable dans le dossier technique (sql_meteo.cpp).

4.3.2. La classe Sqlmeteo

De ces informations précédentes vont émaner une classe Sqlmeteo qui contiendra ces différentes fonctions et variables.

Pour le bon fonctionnement du programme, il est nécessaire d'ajouter les classes propres à mysql:



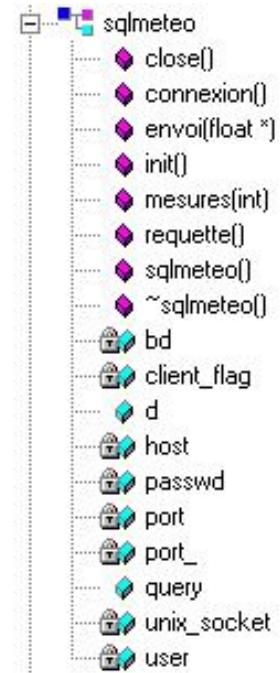
mysql.h



mysql_com.h



mysql_version.h



[Le codage de la classe sql_meteo est consultable dans le dossier technique \(sql_meteo.h\)](#)

4.3.3. L'organisation

Etant donné qu'il y a communication entre le programme et la base de donnée mysql, nous avons du nous mettre d'accord sur les différents champs et formats à utiliser.

Voici le tableau regroupant tout les champs de la base ainsi que leurs types :

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
<input type="checkbox"/>	<u>id</u>	mediumint(9)			Non		auto_increment	
<input type="checkbox"/>	<u>annee</u>	year(4)			Oui	NULL		
<input type="checkbox"/>	<u>mois</u>	char(2)	latin1_swedish_ci		Oui	NULL		
<input type="checkbox"/>	<u>jour</u>	char(2)	latin1_swedish_ci		Oui	NULL		
<input type="checkbox"/>	<u>heure</u>	char(2)	latin1_swedish_ci		Oui	NULL		
<input type="checkbox"/>	<u>minute</u>	char(2)	latin1_swedish_ci		Oui	NULL		
<input type="checkbox"/>	<u>batt_temps</u>	binary(1)			Oui	NULL		
<input type="checkbox"/>	<u>tempext</u>	float			Oui	NULL		
<input type="checkbox"/>	<u>batt_tempe_ext</u>	binary(1)			Oui	NULL		
<input type="checkbox"/>	<u>tempint</u>	float			Oui	NULL		
<input type="checkbox"/>	<u>batt_tempe_int</u>	binary(1)			Oui	NULL		
<input type="checkbox"/>	<u>humext</u>	tinyint(4)			Oui	NULL		
<input type="checkbox"/>	<u>humint</u>	tinyint(4)			Oui	NULL		
<input type="checkbox"/>	<u>temproseext</u>	tinyint(3)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>temproseint</u>	tinyint(3)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>ventvitesse</u>	float		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>ventdirection</u>	smallint(5)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>batt_vent</u>	binary(1)			Oui	NULL		
<input type="checkbox"/>	<u>pression</u>	smallint(5)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>pluiejournee</u>	smallint(5)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>pluiehier</u>	smallint(5)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>pluietotale</u>	smallint(5)		UNSIGNED	Oui	NULL		
<input type="checkbox"/>	<u>pluieannee</u>	year(4)			Oui	NULL		
<input type="checkbox"/>	<u>pluie mois</u>	tinyint(3)			UNSIGNED	Oui	NULL	
<input type="checkbox"/>	<u>pluie jour</u>	tinyint(3)			UNSIGNED	Oui	NULL	
<input type="checkbox"/>	<u>batt_pluie</u>	binary(1)			Oui	NULL		
<input type="checkbox"/>	<u>prevision</u>	char(1)	latin1_swedish_ci		Oui	NULL		
<input type="checkbox"/>	<u>erreur</u>	tinyint(4)			Non	0		

Bien entendu, mon programme devra s'adapter à tous les champs et à tous les formats de la base.

4.3.4. Test

Après l'insertion de l'utilisateur « programme » dans la base Mysql j'ai procédé à un test qui c'est avéré concluant.

On remarque ici qu'il y a eu plusieurs mesures enregistrées dans la base.

Afficher : 30 enregistrement(s) à partir de l'enregistrement n° 30											
en mode horizontal et répéter les en-têtes à chaque groupe de 100 > >> Page n°: 1											
Trier sur l'index: aucune Exécuter											
← →	id	année	mois	jour	heure	minute	batt_temps	tempext	batt_tempe_ext	tempint	batt_temp
<input type="checkbox"/>	1	2006	4	6	15	5	0	24.2	0	24.1	0
<input type="checkbox"/>	2	2006	4	6	15	14	0	24.3	0	21.6	0
<input type="checkbox"/>	3	2006	4	6	15	16	0	24.4	0	21.5	0
<input type="checkbox"/>	4	2006	4	6	15	21	0	24.4	0	23.4	0

On retrouve ces enregistrements sur le site Internet.



4.4. Programme Final

Le programme final est constitué des deux programmes étudiés ci-dessus.

Celui-ci regroupe le programme de réception et d'analyse de trames ainsi que le programme d'écriture Sql.

Le diagramme d'activité général du programme est consultable dans la partie annexe.

Sur ce diagramme, on constate que la minuterie Sql crée une coupure entre la partie serie_météo chargée de l'acquisition et de l'analyse et la partie Sql_météo chargée de l'écriture sur la Base.

4.4.1. Fonctionnement général

Le programme final comporte aussi quelques améliorations techniques :

- Un fonctionnement multi-tâche,
- Une possibilité de changement d'intervalle d'écriture Sql (via IHM),
- Une possibilité de changement de paramètres Sql (via fichier texte),
- Une gestion et affichage des erreurs.

Pour assurer le bon fonctionnement du programme, il est nécessaire d'utiliser **le multi-tâche**.

La première tâche fonctionne comme un chef d'orchestre et lance les différentes fonctions des classes serie_meteo et sql_meteo

La seconde tâche permet un affichage des données météorologiques sur l'IHM, donne 'la main' à l'utilisateur pour pouvoir par exemple réduire et restaurer l'application ou changer la minuterie Sql.

Voici les deux algorithmes des deux tâches :

Tâche 1 :

Faire

Créer 3 classes : acquisition, sqlmeteo et COregonDlg

Initialisation de la minute de départ, de la minute actuelle et de la variable d'erreur

Ouverture et paramétrage de la ligne Rs232

Faire

Réception des données météorologiques et enregistrement

Tant que minute départ différente de minute actu

Envoi du tableau d par un passage en argument

Lecture de config.txt

Ouverture et paramétrage de la liaison Mysql selon config.txt

Création de la requête

Envoi de la requête

Fermeture de la liaison série et Mysql

Tout le temps

Tâche 2

Faire

Actualiser l'affichage des données et des erreurs

Lancer les fonctions propres aux boutons

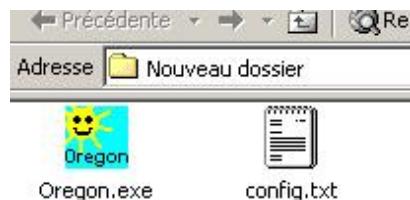
Laisser 'la main' à l'utilisateur

Tout le temps

Le codage de la partie multitâche ainsi que **de nombreux commentaires** sont consultables dans le dossier technique (OregonDlg.cpp).

Le fichier config.txt est un fichier contenant toutes les informations relatives à la connexion sur une base Mysql.

Lorsque l'on lance le logiciel pour la première fois, le fichier config.txt sera créé :



Il apparaît dans l'ordre, séparé d'une tabulation :



- L'adresse du serveur,
- Le nom d'utilisateur de la base,
- Le mot passe,
- Le nom de la base,
- Le port de communication.

Cela permet de créer une certaine capacité d'adaptation du programme face à une base Mysql ne comportant pas forcement les mêmes configurations que nous avons choisies.

Si l'utilisateur modifie ce fichier config.txt puis l'enregistre, les nouveaux paramètres seront pris lors d'un nouveau démarrage.

```
sqlmeteo::sqlmeteo()
{
    //Chargement de la configuration de la ligne
    unix_socket=NULL; //socket

    //depuis le fichier txt
    FILE * config; // declaration du pointeur de fichier source
    config=fopen("config.txt","r");// ouverture du fichier source

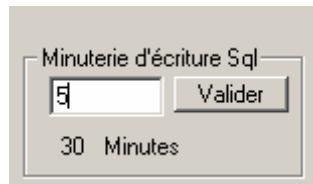
    if(config!=NULL) // si le pointeur different de NULL (LE FICHIER EXISTE)
    {
        //lire fichier source et enregistrer
        fscanf(config,"%s\t%s\t%s\t%s\t%d",host,user,passwd,db,port_);
        port=port_[0];
        fclose(config); // Fermeture du fichier
    }
    else //si pas de fichier txt
    {
        config=fopen("config.txt","w");

        // ecrire les donnees dans le fichier destination
        fprintf(config,"%s\t%s\t%s\t%s\t%s","127.0.0.1","programme","password","meteo","3306");

        config=fopen("config.txt","r");

        //lire fichier source et enregistrer
        fscanf(config,"%s\t%s\t%s\t%s\t%d",host,user,passwd,db,port_);
        port=port_[0];
        fclose(config); // Fermeture du fichier
    }
}
```

La fonction minuterie permet à l'utilisateur, via l'IHM du programme de modifier le temps en minute entre chaque écriture sur la base Mysql.



Le choix de cette petite option a été retenu pour enrichir les fonctionnalités du programme et permettre des écritures de données plus longues ou plus courtes sur la base.

Ce code est exécuté lorsque l'utilisateur clique sur le bouton valider de la minuterie Sql.

```
void COregonDlg::OnVal()
{//bouton valider pour le changement de la minuterie d'écriture Sql
    char min[3];
    char string[3];

    //Capture la valeur entrée par l'utilisateur
    CEdit *Champedit=(CEdit *) GetDlgItem(IDC_TEMP);
    Champedit->GetWindowText(string,3);
    sscanf(string,"%d",&min_calibrage);

    //Affiche la valeur de l'intervalle
    CStatic *inter=(CStatic*)GetDlgItem(IDC_MINUTEINTER);
    sprintf(min,"%d",min_calibrage);
    inter->SetWindowText(min);
}
```

Ce code capture la valeur insérée dans le champ éditable et enregistre la valeur dans la variable min_calibrage, le changement de la minuterie prend effet instantanément et ne demande pas de redémarrage du programme.

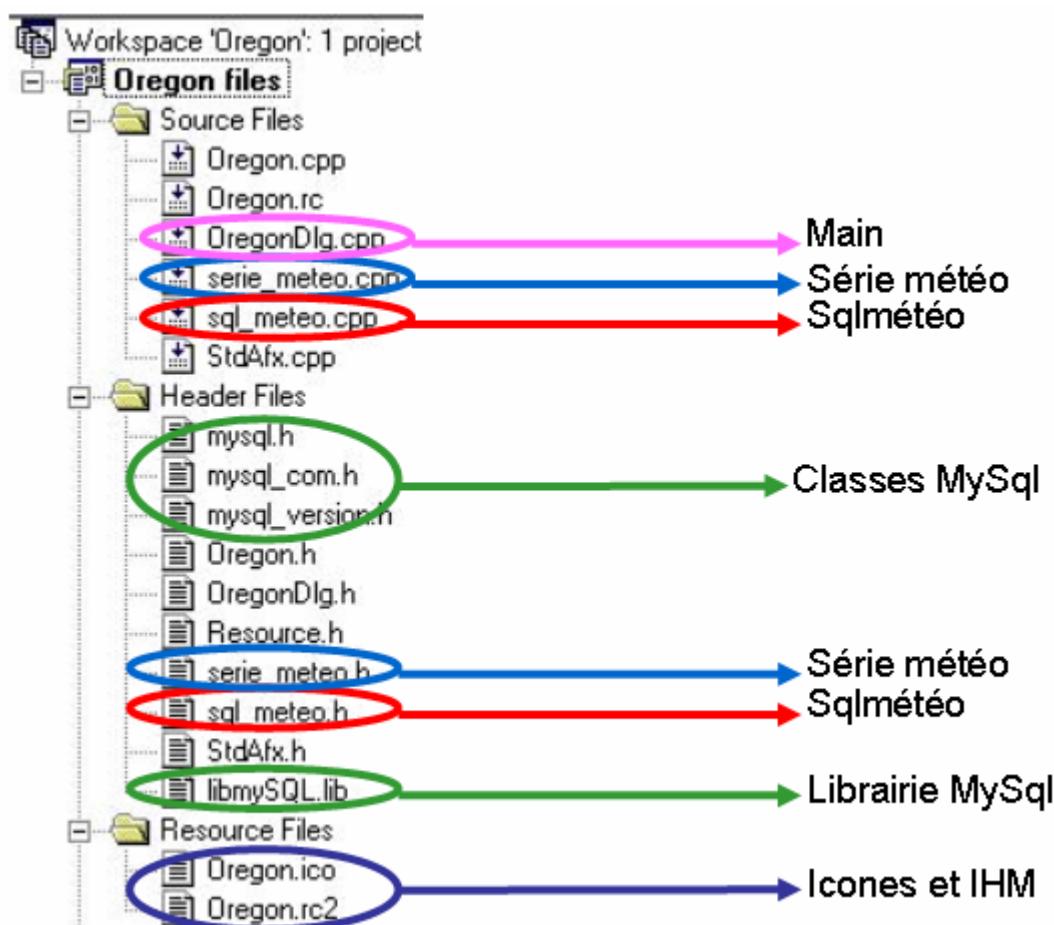
Du point de vue des **erreurs**, chacune d'entre elles sont répertoriée dans 4 catégories :

- Erreur série,
- Erreur Sql_serveur,
- Erreur donnée,
- Déconnexion.

Les erreurs sont aussi expliquées dans le dossier technique et dans le fichier d'aide.

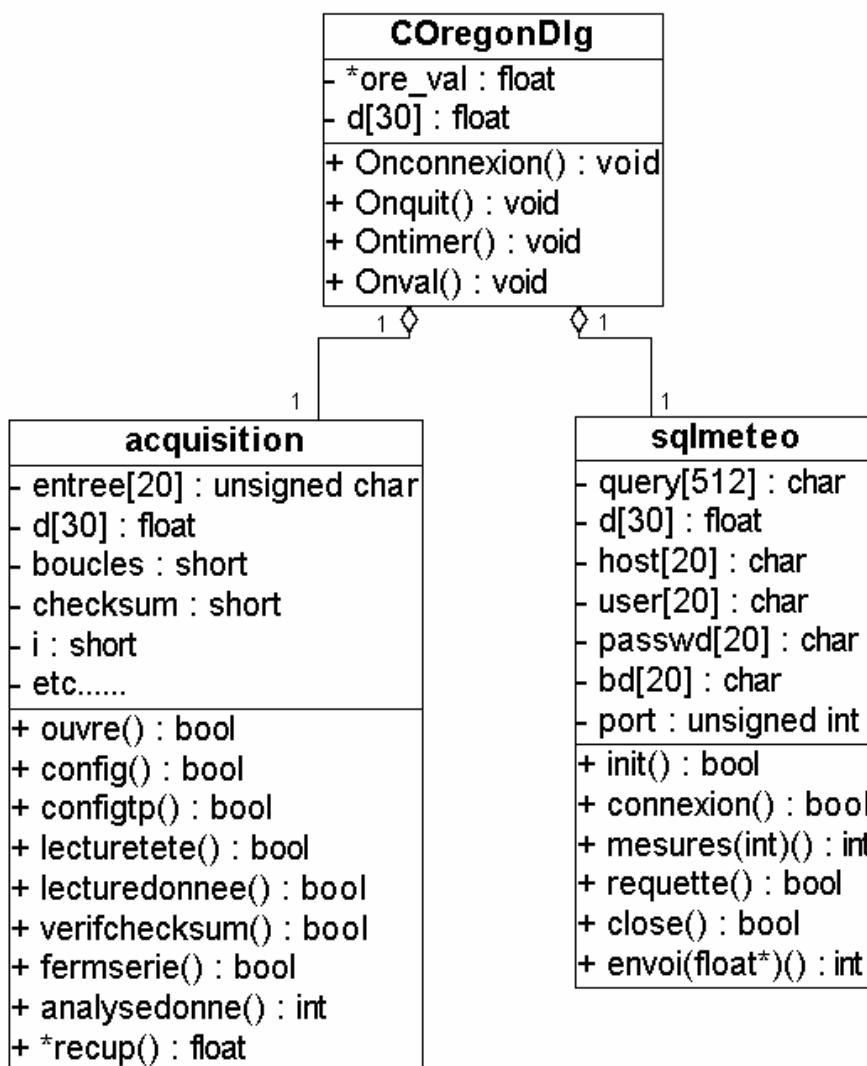
4.4.2. Organisation des fichiers

- OregonDlg.cpp joue le rôle de l'arbitre, c'est lui qui lance les différentes fonctions des classes acquisition et Sqlmeteo.
- OregonDlg.cpp fonctionne en multitâche, il est capable de faire l'acquisition des données météorologiques tout en prenant en compte l'interaction entre le logiciel et l'utilisateur.
- serie_meteo.cpp et serie_meteo.h contiennent les fonctions et les variables liées à l'acquisition.
- sql_meteo.cpp et sql_meteo.h contiennent les fonctions et les variables liées à l'écriture sur une base MySql.
- mysql.h mysql_com.h et mysql_version.h sont des classes nécessaires au bon fonctionnement d'un dialogue Mysql.
- Oregon.ico et Oregon.rc2 contiennent les informations liées à l'IHM et à l'icône.



4.4.3. Organisation des classes

Voici le diagramme de classe du programme final :



La relation entre la classe COregonDlg et sqlmeteo est une relation d'agrégation.

Idem pour la relation entre la classe COregonDlg et acquisition.

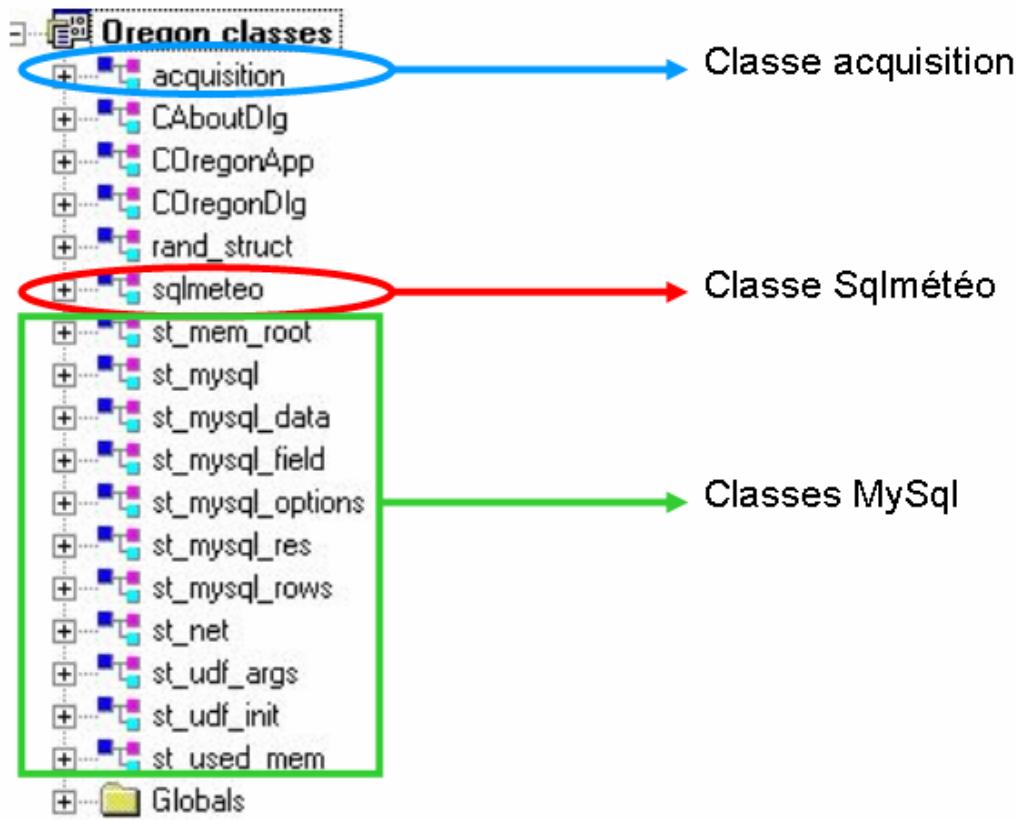
Ces trois classes sont créées en même temps dans COregonDlg.cpp lors du démarrage du programme ce qui laisse la possibilité à la classe COregonDlg d'actionner en quelque sorte les différentes fonctions des classes acquisitions et sqlmeteo.

Aucune liaison n'existe entre la classe acquisition et sqlmeteo.

Le passage du tableau de données et de la variable erreur se fait par un passage en argument depuis COregonDlg. Celui-ci va chercher le tableau dans acquisition puis l'envoi sur sqlmeteo.

Ce choix a été retenu car il centralise le traitement et le déroulement du processus permettant par la suite de fonctionner avec du multitâche.

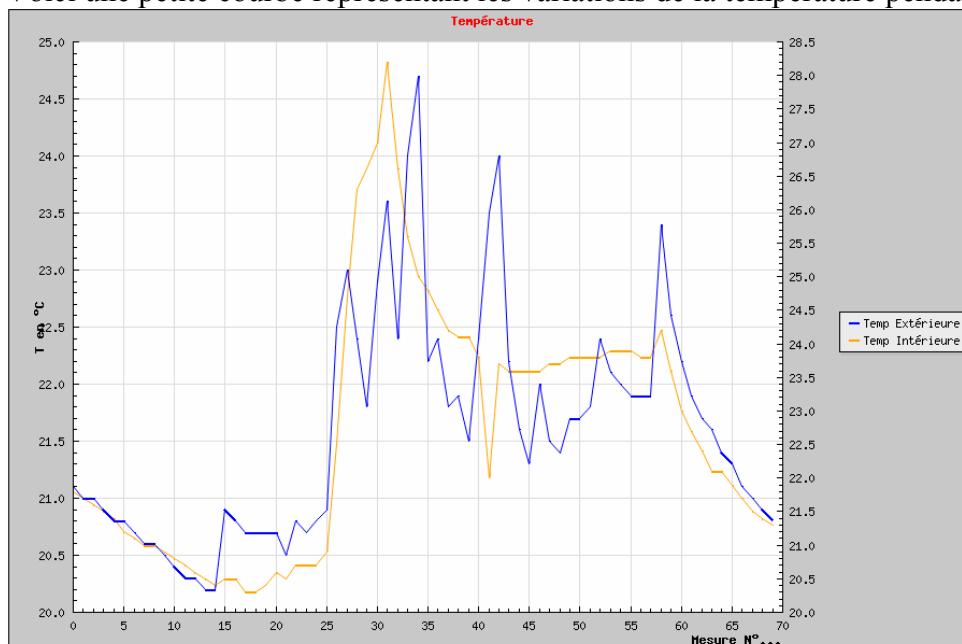
On retrouve sur cette représentation des classes la classe acquisition, la classe Sqlmeteo ainsi que les classes commençant par **st_** qui correspondent aux classes MySql.



4.4.4. Test grandeur nature

A la suite de la création de ce programme, nous avons réalisé un test en écrivant toutes les 30 minutes sur la base Mysql.

Voici une petite courbe représentant les variations de la température pendant 1 jour :



5. Suppléments

5.1. Fichier aide avec HTML Help workshop

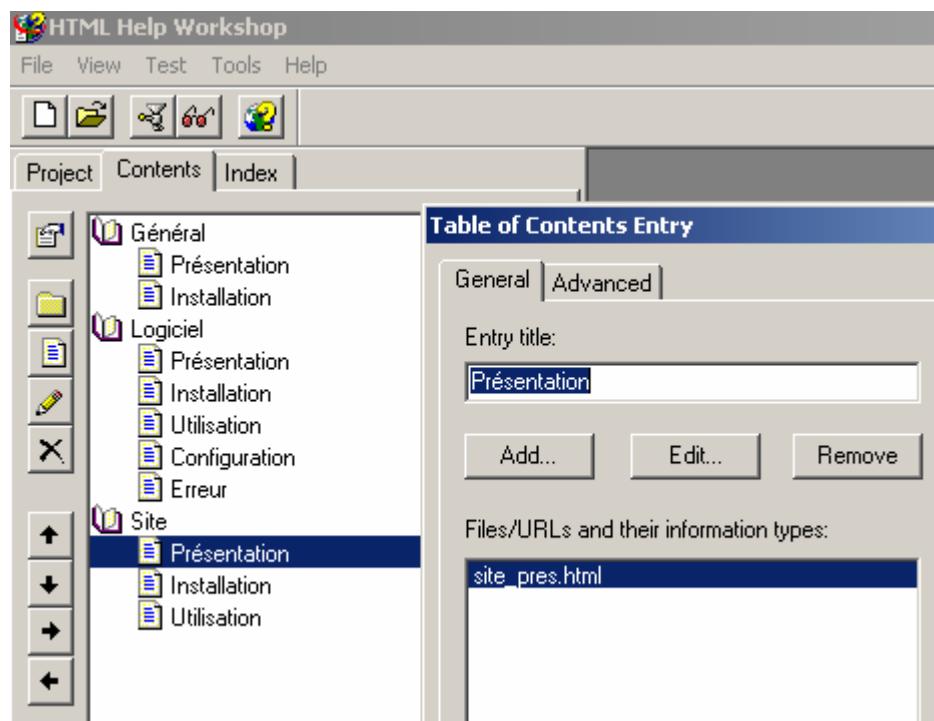
Afin de fournir à l'utilisateur une documentation propre à l'utilisation de notre système j'ai réalisé un fichier aide possédant l'extension .chm

Celui-ci possède 3 grandes rubriques :

- Général
- Logiciel
- Site

Pour créer un fichier d'aide, il faut faire autant de pages html que de pages souhaitées ainsi qu'un dossier contenant les images relatives aux pages.

Ensuite, il suffit de créer un nouveau projet dans le logiciel HTML Help workshop et d'y associer toutes les pages html.



Avec les boutons 'insert a heading' et 'insert a page' je crée facilement une arborescence contenant chacune des pages.

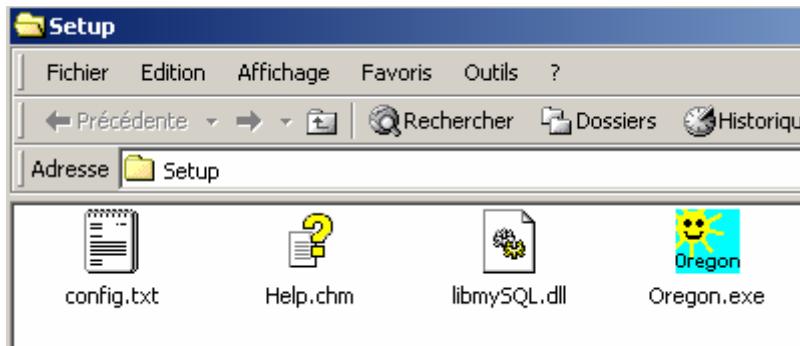
Pour finir, je compile le projet puis se crée le fichier aide :



5.2. Installation automatique avec Inbox 2.0

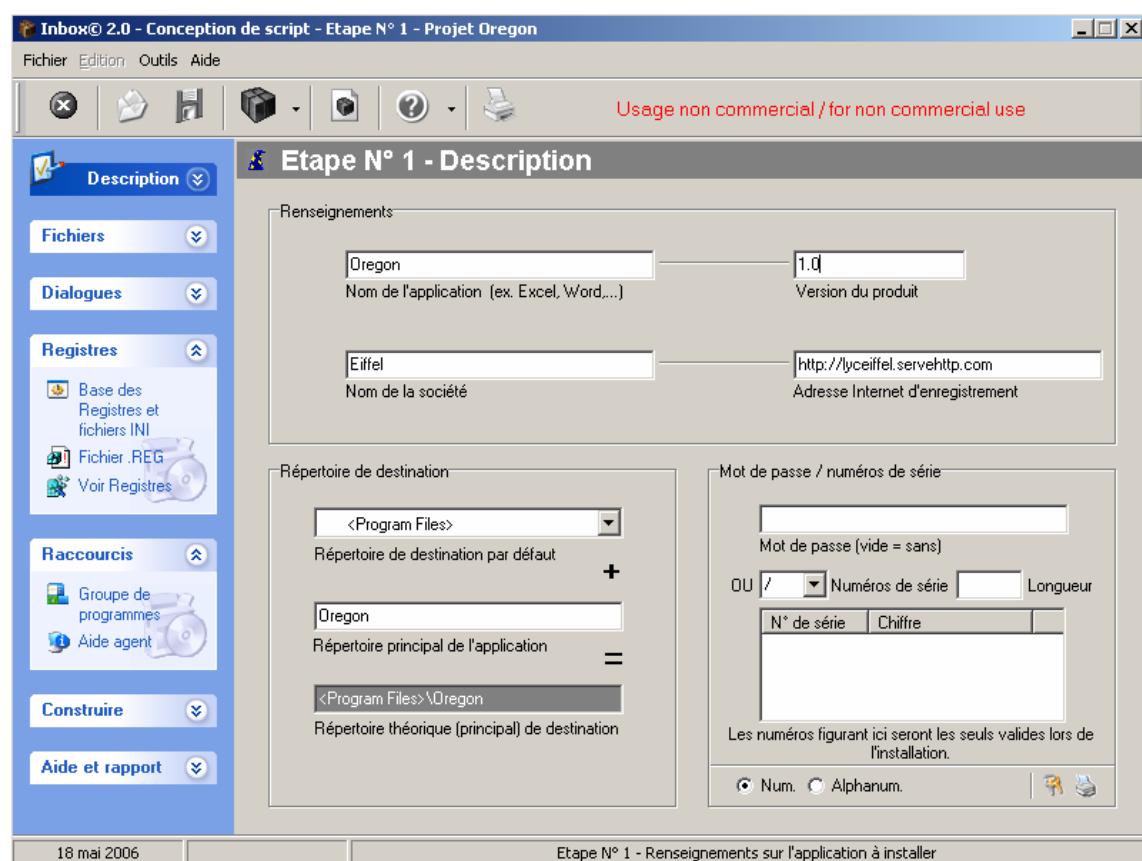
Pour pouvoir implanter facilement le logiciel sur un poste et proposer à l'utilisateur des raccourcis et une aide, il est intéressant de créer un fichier d'installation.

Pour cela, il faut mettre dans un dossier tous les éléments que l'on désire installer.



L'insertion de libmySQL.dll est obligatoire pour permettre le bon fonctionnement du programme sur un ordinateur pas forcément équipé des librairies Mysql

Ensuite, Inbox 2.0 propose la conception d'un fichier d'installation en plusieurs étapes incluant la possibilité d'écrire dans le registre ou d'ajouter des raccourcis supplémentaires.

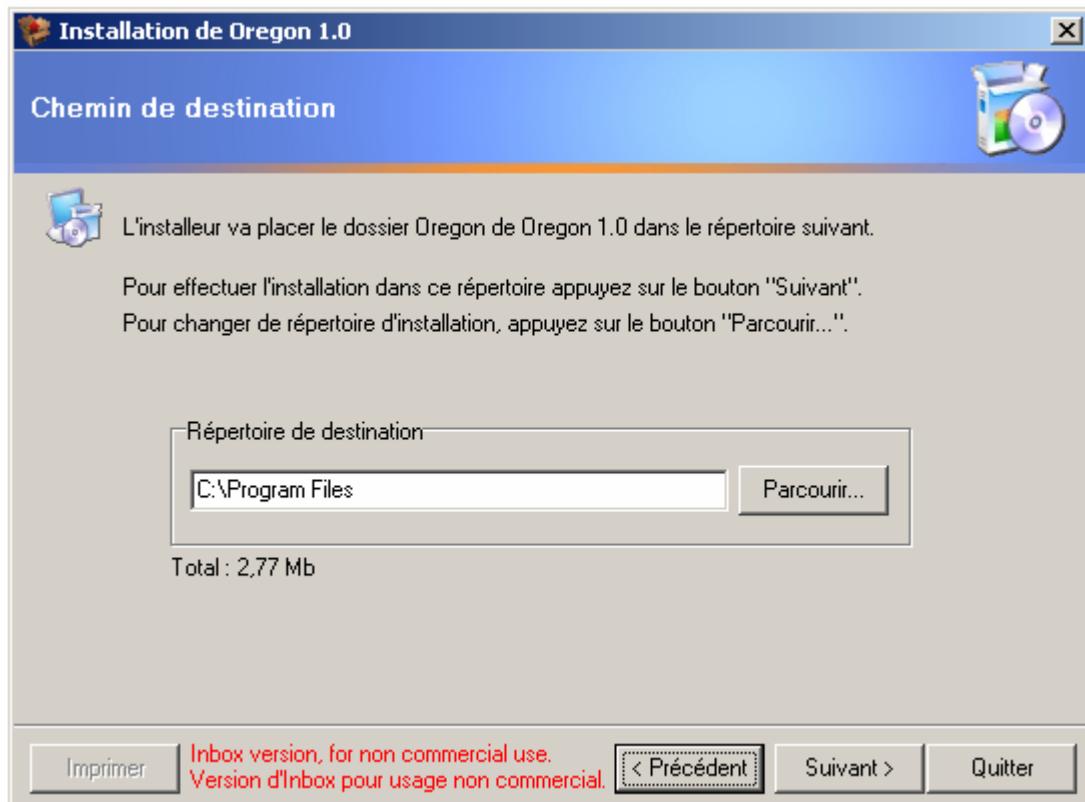


Après avoir complété chaque étape, le programme construit l'exécutable en lui associant un nom et un icône. Il m'a semblé intéressant de mettre sur le bureau des raccourcis de l'exécutable, de l'aide ainsi que de la désinstallation.

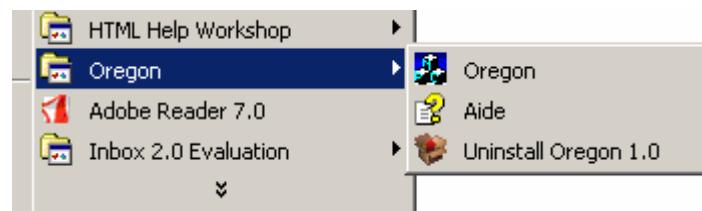
Voici les icônes apparaissant sur le bureau :



Lorsque l'on lance l'installation, une interface de ce style questionne l'utilisateur pour récupérer les informations propres à l'installation :

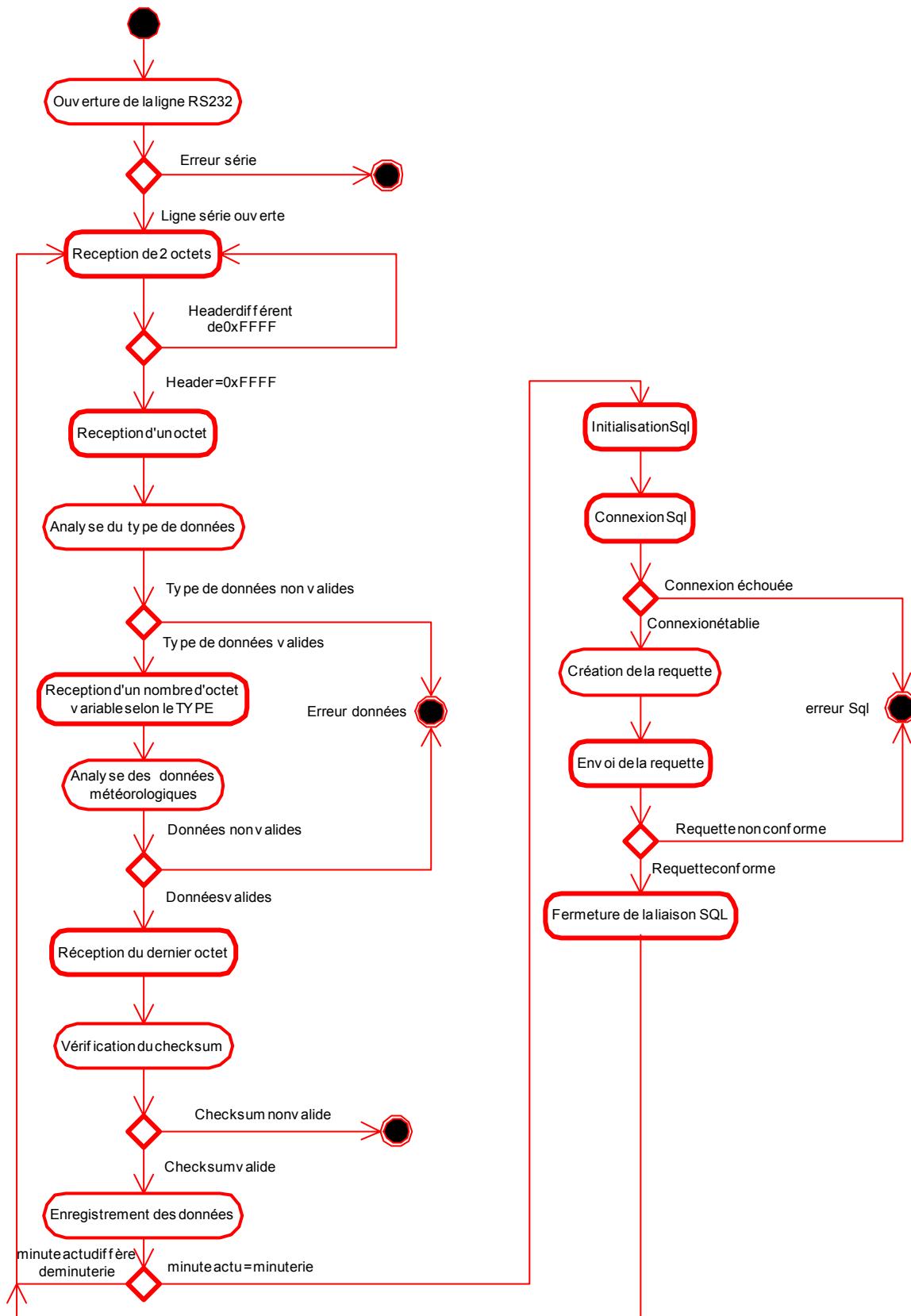


L'installation terminée, il est créé dans le menu démarrer un lien vers le programme, l'aide et la désinstallation :



6. Annexes

6.1. Diagramme d'activité général du programme



6.2. Oregon PCLINK protocole V 0.2

Oregon PCLINK protocol Version 0.2

1. Le port série envoi les données à 9600bps depuis la station de réception par le protocole RS232
2. Pour que le pc puisse recevoir les données de l'Oregon, la broche '**Request to send**' du PC doit être activée pour recevoir les données, si le cas contraire, on ne recevra rien.
3. Quand l'Oregon est en train d'envoyer des données, il enverra une entête 'FFFF' en premier, puis suivit du code concernant le type de données,
4. A la fin de la transmission d'un trame, l'Oregon enverra le "checksum" des données.
5. L'Oregon enverra les données au pc lorsque de nouvelles données lui sont transmises.
6. L'Oregon enverra la donnée 'Minute' au PC chaque minute pour s'assurer que la connection est bonne.
7. L'Oregon enverra la donnée 'clock' au PC chaque heure ou lorsqu'une nouvelle heure est reçue par radio fréquence,
8. L'Oregon n'enverra pas les données constamment au PC, il enverra les données bytes par bytes. (Si il y a d'autres tâches pour l'Oregon à faire - Comme recevoir des info concernant le vent, L'Oregon recevra en premier les données, puis les enverra au PC).

Data		Data Type							Clock	EXTBTH
		Wind	Rain	TH	Mushroom	T	BTH	Minute		
Header 1	Bit 0	1	1	1	1	1	1	1	1	1
	Bit 1	1	1	1	1	1	1	1	1	1
	Bit 2	1	1	1	1	1	1	1	1	1
	Bit 3	1	1	1	1	1	1	1	1	1
	Bit 4	1	1	1	1	1	1	1	1	1
	Bit 5	1	1	1	1	1	1	1	1	1
	Bit 6	1	1	1	1	1	1	1	1	1
	Bit 7	1	1	1	1	1	1	1	1	1
Header 2	Bit 0	1	1	1	1	1	1	1	1	1
	Bit 1	1	1	1	1	1	1	1	1	1
	Bit 2	1	1	1	1	1	1	1	1	1
	Bit 3	1	1	1	1	1	1	1	1	1
	Bit 4	1	1	1	1	1	1	1	1	1
	Bit 5	1	1	1	1	1	1	1	1	1
	Bit 6	1	1	1	1	1	1	1	1	1
	Bit 7	1	1	1	1	1	1	1	1	1
3rd Byte (DEVICE) TYPE	Bit 0	wind	rain	thermo hygro	mushroom	thermo only	thermo hygro baro	Minute	Clock	00000110 thermo hygro baro
	Bit 1									
	Bit 2									
	Bit 3									
	Bit 4									
	Bit 5									
	Bit 6									
	Bit 7									
4th Byte	Bit 0	--	--	Channel number	--	Channel number	--	Date 1 digit minute	Date 1 digit minute	--
	Bit 1									
	Bit 2									
	Bit 3									
	Bit 4	Gust over	Rate over	Dew under	Dew under	--	Dew under	Date 10 digit minute	Date 10 digit minute	Dew under
	Bit 5	vgerage ov	Total over	--	--	--	--	Date 10 digit minute	--	--
	Bit 6	Low batt.	Low batt.	Low batt.	Low batt.	Low batt.	Low batt.	Date 10 digit minute	Low batt.	Low batt.
	Bit 7	--	esterday ov	--	--	--	--	Batt. Low	Batt. Low	--
5th Byte	Bit 0	Wind direction	Current Rain Rate	Temp 0.1°C	Temp 0.1°C	Temp 0.1°C	Temp 0.1°C	Check-sum	Date 1 digit hour	Temp 0.1°C digit
	Bit 1	1° digit	1 digit in mm/hr	digit	digit	digit	digit			
	Bit 2	Wind direction	Current Rain Rate	Temp 1°C	Temp 1°C	Temp 1°C	Temp 1°C			
	Bit 3	10° digit	10 digit in mm/hr	digit	digit	digit	digit			
	Bit 4	Wind direction	Current Rain Rate	Temp 1°C	Temp 1°C	Temp 1°C	Temp 1°C			
	Bit 5	10° digit	10 digit in mm/hr	digit	digit	digit	digit			
	Bit 6	Gust Wind Speed	Total Rainfall 0.1 digit in mm	Over/Under Sign	Over/Under Sign	Over/Under Sign	Over/Under Sign			
	Bit 7	0.1m/sec	0.1m/sec	Sign	Sign	Sign	Sign			
6th Byte	Bit 0	Gust Wind Speed	Total Rainfall 1 digit in mm	Hum 1% digit	Hum 1% digit	Check-sum	Hum 1% digit	Date 1 digit Day	Temp 10°C digit	Temp 100°C OverUnder Sign
	Bit 1	1 m/sec	10 digit in mm	1% digit	1% digit		1% digit			
	Bit 2	Gust Wind Speed	Total Rainfall 10 digit in mm	Hum 10% digit	Hum 10% digit		10% digit			
	Bit 3	10 m/sec	10 digit in mm	Sign	Sign		Sign			
	Bit 4	Gust Wind Speed	Total Rainfall 100 digit in mm	Temp 10°C	Temp 10°C		Temp 10°C			
	Bit 5	100 m/sec	100 digit in mm	Temp 100°C	Temp 100°C		Temp 100°C			
	Bit 6	Gust Wind Speed	Total Rainfall 1000 digit in mm	Over/Under	Over/Under		Over/Under			
	Bit 7	1000 m/sec	1000 digit in mm	Sign	Sign		Sign			
7th Byte	Bit 0	Gust Wind Speed	Total Rainfall 10000 digit in mm	Hum 100% digit	Hum 100% digit	Check-sum	Hum 100% digit	Date 1 digit Month	Hum 1% digit	Hum 10% digit
	Bit 1	1000 m/sec	10000 digit in mm	100% digit	100% digit		100% digit			
	Bit 2	Gust Wind Speed	Total Rainfall 100000 digit in mm	Temp 1000°C	Temp 1000°C		Temp 1000°C			
	Bit 3	10000 m/sec	100000 digit in mm	Temp 10000°C	Temp 10000°C		Temp 10000°C			
	Bit 4	Gust Wind Speed	Total Rainfall 1000000 digit in mm	Over/Under	Over/Under		Over/Under			
	Bit 5	100000 m/sec	1000000 digit in mm	Sign	Sign		Sign			
	Bit 6	Gust Wind Speed	Total Rainfall 10000000 digit in mm	Temp 100000°C	Temp 100000°C		Temp 100000°C			
	Bit 7	1000000 m/sec	10000000 digit in mm	Over/Under	Over/Under		Over/Under			

Serveur Météo – BTS 2005-2006

8th Byte	Bit 0	Average Wind Speed 0.1 m/sec	Total Rainfall 100 digit in mm	Dew Temp 1°C digit	Dew Temp 1°C digit		Dew Temp 1°C digit	Date 1 digit Year	Dew Temp 1°C digit
	Bit 1						Dew Temp 10°C digit	Date 10 digit Year	Dew Temp 10°C digit
	Bit 2								
	Bit 3								
	Bit 4	Average Wind Speed 1 m/sec	Total Rainfall 1000 digit in mm	Dew Temp 10°C digit	Dew Temp 10°C digit				
	Bit 5								
	Bit 6								
	Bit 7								
9th Byte	Bit 0	Average Wind Speed 10 m/sec	Yesterday Rainfall 1 digit in mm	Check-sum	Check-sum		ADC BARO Reading	Date 1 digit Year	Dew Temp 1°C digit
	Bit 1								
	Bit 2								
	Bit 3								
	Bit 4		Yesterday Rainfall 10 digit in mm						
	Bit 5	Chill no data							
	Bit 6	Chill over							
	Bit 7	Sign							
10th Byte	Bit 0	Wind Chill 1°C digit	Yesterday Rainfall 100 digit in mm	Check-sum			Weather Status	Date 1 digit Year	ADCbit9
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4	Wind Chill 10°C digit	Yesterday Rainfall 1000 digit in mm						Weather Status
	Bit 5								---
	Bit 6								
	Bit 7								
11th Byte	Bit 0	Check-sum	Total Start Date 1 digit minute	Check-sum			Sea level offset 0.1 digit mb	Date 1 digit Year	Sea level offset 0.1 digit mb
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4		Total Start Date 10 digit minute						Sea level offset 1 digit mb
	Bit 5								---
	Bit 6								
	Bit 7								
12th Byte	Bit 0	Check-sum	Total Start Date 1 digit hour	Check-sum			Sea level offset 10 digit mb	Date 1 digit Year	Sea level offset 1 digit mb
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4		Total Start Date 100 digit hour						Sea level offset 10 digit mb
	Bit 5								---
	Bit 6								
	Bit 7								
13th Byte	Bit 0	Check-sum	Total Start Date 1 digit Day	Check-sum			Sea level offset 100 digit mb	Date 1 digit Year	Sea level offset 100 digit mb
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4		Total Start Date 10 digit Day						Sea level offset 1000 digit mb
	Bit 5								---
	Bit 6								
	Bit 7								
14th Byte	Bit 0	Check-sum	Total Start Date 1 digit Month	Check-sum			Sea level offset 1000 digit mb	Date 1 digit Year	Check-sum
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4		Total Start Date 10 digit Month						
	Bit 5								
	Bit 6								
	Bit 7								
15th Byte	Bit 0	Check-sum	Total Start Date 1 digit Year	Check-sum			Check-sum	Date 1 digit Year	Check-sum
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4								
	Bit 5								
	Bit 6								
	Bit 7								
16th Byte	Bit 0	Check-sum		Check-sum			Check-sum	Date 1 digit Year	Check-sum
	Bit 1								---
	Bit 2								
	Bit 3								
	Bit 4								
	Bit 5								
	Bit 6								
	Bit 7								

6.3. Tableau d'organisation des données

Données envoyées par Oregon		Data
Vent	Degrés orientation	0
Vent	Vitesse	1
Vent	Vitesse moyenne	2
Vent	Vent froid	3
Vent	Batterie	4
Pluie	Précipitation actuelle	5
Pluie	Précipitation total	6
Pluie	Précipitation hier	7
Pluie	Date minute	24
Pluie	Date heure	25
Pluie	Date jour	26
Pluie	Date mois	27
Pluie	Date année	28
Pluie	Batterie	29
Extérieur	Température extérieur	8
Extérieur	Humidité extérieur	9
Extérieur	Température de rosé	10
Extérieur	Batterie	11
Intérieur	Température intérieur	12
Intérieur	Humidité intérieur	13
Intérieur	Température de rosé	14
Intérieur	Prévision	15
Intérieur	Pression atmosphérique	16
Intérieur	Batterie	17
Date	Minute	18
Date	Heure	19
Date	Jour	20
Date	Mois	21
Date	Année	22
Date	Batterie	23

6.4. Journal de bord

Semaine 1 Prise en connaissance du projet en partenariat avec monsieur ROGER Olivier
 Distinction entre le projet "Oregon Champollion" et "station météo Eiffel"
 Mise en disposition d'un poste de travail répondant au cahier des charges
 Esquisses de diagrammes de séquences
 finalisation des diagrammes
 installation de Visual C++ 6,0, Microsoft Word 97
 Installation réseau de la salle informatique
 Installation réseau passage de câbles rj45 dans les faux plafonds
 Configuration des postes en réseau
 Lecture de la documentation technique du système Oregon
 Familiarisation avec la station météo
 Test des différentes fonctionnalités
 Etude du protocole de communication

Semaine 2 Traduction du protocole de communication
Modification des diagrammes de séquences:
Diagramme Mise à jour des données, de l'heure (clock) et la vérification de connectivité
Création d'un UML du programme d'émulation de la station Oregon
Mise en place d'une collaboration d'échange de données
Esquisse d'un schéma de programmation
Entrevue professeur élève

Semaine 3 Finalisation Diagrammes de séquences
Réflexion sur le mode de stockage des données
Création d'une procédure de test de dialogue RS232
Validation de la procédure de test
Création d'un programme permettant d'émuler la station OREGON

Semaine 4 Création d'un programme permettant d'émuler la station OREGON
Réévaluation des variables

Semaine 5 Modélisation du programme
Finalisation du programme
Test de fonctionnalité en utilisant un câble Null-modem et un PC supplémentaire

Semaine 6 Revue de projet numéro 1
Création du programme d'analyse et de réception des trames

Semaine 7 Création du programme d'analyse et de réception des trames
Travail sur la classe acquisition

Semaine 8 Mise en place de la station/module Orphy
Test du programme d'émulation de la station Oregon avec l'installation Orphy
Validation du programme d'émulation
Structuration du programme de réception
Créations de la classe acquisition

Semaine 9 Finalisation du programme de réception
Procédures de test pour le programme de réception

Semaine 10 Créations de la classe Sql_meteo pour la base de donnée
Création d'un programme d'écriture en SQL

Semaine Procédures de tests du programme d'écriture Sql

11

Fusion des deux programmes

Semaine

12

Ajout de la gestion d'erreurs

Création de l'IHM

Ajout d'un mode de paramétrage de la minuterie d'écriture Sql

Ajout de la possibilité de paramétrage de la configuration Sql via un fichier texte

Semaine

13

Test grandeur nature

Création d'une aide à l'utilisation du système

Rédaction du rapport de projet

Semaine

14

Revue de projet numéro2

Création d'un fichier d'installation

Semaine

15

Fin de la rédaction du rapport de projet

Préparation à l'examen

7. Conclusion

Durant ce projet, j'ai pu aborder un système nouveau ainsi que des grandeurs physiques qui m'étaient relativement inconnues.

Prendre en connaissance un projet m'a demandé une capacité d'analyse accrue du point de vue conception UML et schématisation du système.

J'ai pu également consolider mes bases en conception logicielle car j'ai su créer deux logiciels relativement complexes : le programme d'émulation qui m'a permis de comprendre le fonctionnement du système et le programme 'Oregon final' répondant parfaitement au cahier des charges et possédant certaines options pratiques.

Travailler chacun sur une partie bien distincte entraîne forcément des désagréments lors de la réunion des deux parties. Dans notre cas, il s'agissait de la liaison entre le programme et la base MySql, j'ai su donc m'adapter aux différentes exigences que demandait l'écriture sur une base de données.

V-2 Mr Roger : Communication serveur client**SERVEUR METEO : PARTIE #2 BASE DE DONNEES – COMMUNICATION CLIENT SERVEUR****Par Olivier ROGER**

SOMMAIRE :

I - CAHIER DES CHARGES SIMPLIFIE.....	74
II-1 PRESENTATION DU PROJET	74
II-2 EXPRESSION DU BESOIN	74
II-3 MOYENS DISPONIBLE ET CONTRAINTES DE REALISATIONS	74
II-4 TRAVAIL A EFFECTUER	74
<i>II-4-1 Analyse.....</i>	74
<i>II-4-2 Conception.....</i>	74
II- ANALYSE	76
II-1 MODELISATION DU SERVEUR	76
II-2 DIAGRAMME DE CONTEXTE.....	77
II-3 CAS D'UTILISATION.....	78
II-4 DIAGRAMME DE SEQUENCE	79
<i>II-4-1 Administrer la base de données.....</i>	79
<i>II-4-2 Consulter une page Web</i>	80
III- CHOIX EFFECTUES.....	81
III-1 SOLUTIONS TECHNIQUES.....	81
III-2 MATERIEL.....	81
<i>III-2-1 Système d'exploitation et services</i>	81
III-3 OUTILS DE DEVELOPPEMENT	81
<i>III-3-1 Services utilisés</i>	81
<i>III-3-2 Codage.....</i>	82
<i>III-3-3 Graphisme</i>	82
<i>III-3-4 Divers</i>	82
IV- TRAVAIL REALISE.....	83
IV-1 TRAVAUX PRELIMINAIRES.....	83
IV-2 INSTALLATION DU SERVEUR	83
<i>IV-2-1 Partitionnement</i>	83
<i>IV-2-2 Installation du système</i>	83
<i>IV-2-3 Vérification et sauvegarde</i>	83
IV-3 INSTALLATION DES SERVICES	84
<i>IV-3-1 IIS</i>	84
<i>IV-3-2 Mysql</i>	84
<i>IV-3-3 PHP</i>	84
<i>IV-3-4 PHPMyAdmin</i>	85
IV-4 DETERMINATION DE LA PRESENTATION DES PAGES	85
<i>IV-4-1 Réflexion</i>	85
<i>IV-4-2 Réalisation</i>	85
<i>IV-4-3 Respect des contraintes</i>	89
<i>IV-4-4 Problème rencontré</i>	89
<i>IV-4-5 Présentation de la page Web</i>	91
IV-5 REALISATION DE LA BASE DE DONNEES	92
<i>IV-5-1 Contraintes</i>	92
<i>IV-5-2 Analyse.....</i>	92
<i>IV-5-3 Optimisation</i>	93
<i>IV-5-4 Création de la base de données</i>	94
IV-6 EXPLOITATION DES DONNEES	96
<i>IV-6-1 Fichier de configuration</i>	96
<i>IV-6-2 Affichage simple.....</i>	96
<i>IV-6-3 Calculs</i>	99
<i>IV-6-4 Gestion des archives</i>	101
<i>IV-6-5 Génération de courbes</i>	102
<i>IV-6-6 Récapitulatif des fonctionnalités du site</i>	106
<i>IV-6-7 Simulation de fonctionnement.....</i>	106
IV-7 MISE EN LIGNE DU SITE	108

<i>IV-7-1 Sécurité</i>	108
<i>IV-7-2 Configuration réseau</i>	109
IV-8 MISE EN PLACE D'UNE DMZ ET D'UN FIREWALL	110
<i>IV-8-1 Analyse</i>	110
<i>IV-8-2 Mise en service</i>	111
<i>IV-8-3 Interface de gestion</i>	111
<i>IV-8-4 Problèmes rencontrés</i>	112
V- CONCLUSION	113
VI-ANNEXES :	114
VI-I RECETTE D'INSTALLATION PHP5	114
<i>VI-1-1 Installation</i>	114
<i>VI-1-2 Configuration</i>	114
VI-2- ORGANISATION DES FICHIERS	116
VI-3 SCRIPT DE SIMULATION	116
VI-4 WEBOGRAPHIE / BIBLIOGRAPHIE	118
VI-5 JOURNAL DE BORD	118

TABLE DES ILLUSTRATIONS :

<i>Figure 1: Modélisation</i>	76
<i>Figure 2 : Diagramme de contexte</i>	77
<i>Figure 3 : Cas d'utilisation</i>	78
<i>Figure 4 : Séquence – Administration Bdd</i>	79
<i>Figure 5 : Séquence – Consultation page Web</i>	80
<i>Figure 6 : Installation de Mysql.....</i>	84
<i>Figure 7 : Croquis des pages web.....</i>	85
<i>Figure 8 : Evolution de la page web</i>	86
<i>Figure 9 : Première modélisation bdd</i>	92
<i>Figure 10 : Modélisation finale bdd</i>	93
<i>Figure 11 : Principaux type de la BDD</i>	94
<i>Figure 12 : Création de la base de données</i>	94
<i>Figure 13 : Sélection du langage SQL</i>	94
<i>Figure 14 : Affichage de données</i>	98
<i>Figure 15 : utilisation des calculs.....</i>	100
<i>Figure 16 : Utilisation des archives.....</i>	101
<i>Figure 17 : Résultat d'une recherche</i>	102
<i>Figure 18 : Graphique sur une journée</i>	105
<i>Figure 19 : Graphique sur une période</i>	105
<i>Figure 20 : Redirection de l'IP</i>	109
<i>Figure 21 : Modélisation de la DMZ</i>	110
<i>Figure 22 : Interface de gestion Ipcop</i>	111

I -Cahier des charges simplifié

II-1 Présentation du projet

Cette partie du projet doit permettre d'accéder aux données météorologiques stockées dans la première partie du projet. Les données doivent être accessibles et administrables depuis un réseau local (Intranet) et depuis Internet.

La station utilisée dans le cadre du projet est la station Oregon WMR928. Elle est identique à la station utilisée par le collège Champollion, à qui ce projet est destiné.

II-2 Expression du besoin

On souhaite offrir la possibilité aux utilisateurs de la station météo (ici des collégiens) de consulter les relevés effectués au cours d'une journée mais également des relevés plus anciens. Une IHM devra donc être élaborée pour permettre l'exploitation des données stockées dans une base Mysql. Cette interface doit être visualisable quel que soit le navigateur utilisé : Internet Explorer 6.0, Netscape 7.0, Opera et Firefox.

II-3 Moyens disponible et contraintes de réalisations

Pour réaliser ce projet nous disposons d'un serveur Celeron 2.6Ghz 1Go de ram et 20Go de disque dur. On peut associer à ce serveur les systèmes d'exploitation Windows 2000 pro ou Windows 2000 Serveur.

Le serveur Web doit être réalisé avec Wamp5 qui est une installation automatique de PHP5, Mysql et apache.

Les langages de programmation autorisés sont : PHP, Mysql, HTML. L'environnement de développement PHP est libre.

II-4 Travail à effectuer

II-4-1 Analyse

- S'approprier de la modélisation du système
- Finaliser la modélisation du système
- Caractériser les informations fournies par la station météo

II-4-2 Conception

- Installation des outils de développement.
- Définir la présentation des pages Web
- Réaliser une base de données cohérente de simulation, correspondant à la station météo WRM98.
- Administrer cette base de données.
- Assurer l'exploitation, l'affichage (graphique et textuel) des données à partir d'un poste client en respectant les critères de choix de celui-ci.

- Interconnecter la station météo avec le serveur météo.
- Administrer la base de données réelle WRM98.
- Assurer l'exploitation, l'affichage (graphique et textuel) des données réelles WRM98 à partir d'un poste client en respectant les critères de choix de celui-ci.
- Gérer la planification
- Assurer la traçabilité des travaux
- Rédiger les documents relatifs au projet

II- Analyse

La partie que je traite dans ce projet s'oriente sur le serveur et le client. Les différentes analyses effectuées portent donc sur ces entités.

II-1 Modélisation du Serveur

Voici une modélisation du serveur hors contexte, c'est-à-dire sans les éléments qui l'entourent habituellement. Pour une modélisation complète se référer au dossier commun.

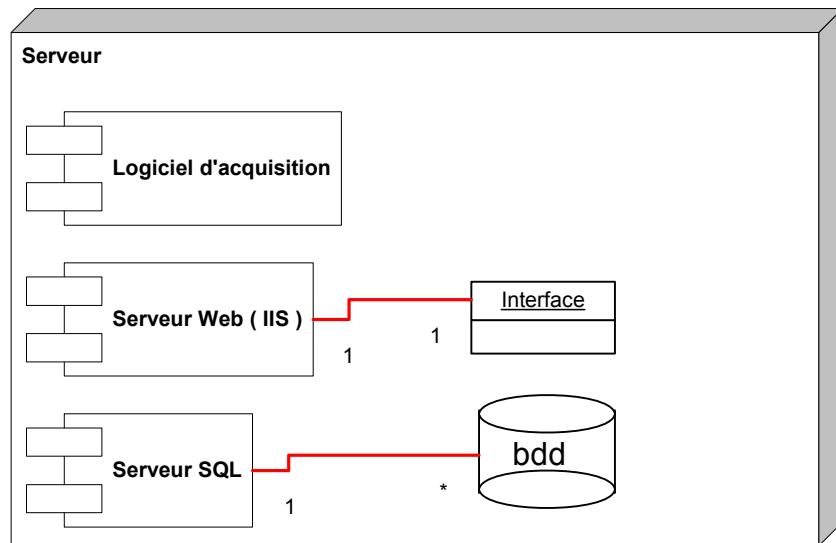


Figure 15: Modélisation

Le serveur météo se compose de trois grands ensembles :

- Le logiciel d'acquisition permettant de récupérer et stocker les données envoyées par la station (cette partie ne me concerne pas)
- Un serveur Web avec un interpréteur de pages PHP afin de pouvoir afficher des pages dynamiques sur l'interface de visualisation.
- Un serveur SQL avec une ou plusieurs bases de données en fonction des besoins. Ces bases de données permettront le stockage des différentes mesures.

II-2 Diagramme de contexte

Le serveur accueillant la base de données ainsi que le programme d'acquisition peut être représenté de la manière suivante :

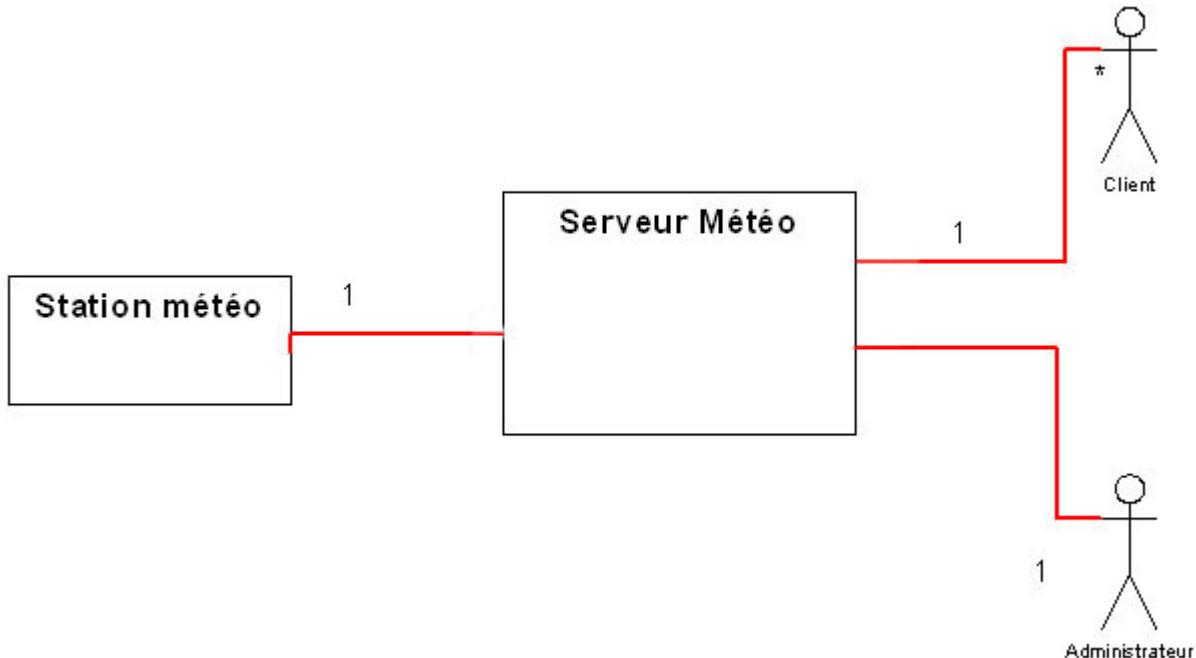


Figure 16 : Diagramme de contexte

Le serveur météo est en liaison avec la console principale de la station météo via liaison RS232.

Le serveur est accessible par deux types de personnes différents : Le client qui aura la possibilité de consulter l'interface et l'administrateur qui pourra consulter l'interface et administrer la base de données. Comme le montre les multiplicités l'interface est accessible à 1 ou plusieurs clients en même temps.

II-3 Cas d'utilisation

Le cas d'utilisation est proche du diagramme de contexte. Il permet de visualiser les actions proposées par l'élément étudié. En l'occurrence on peut voir quelles sont les possibilités du serveur :

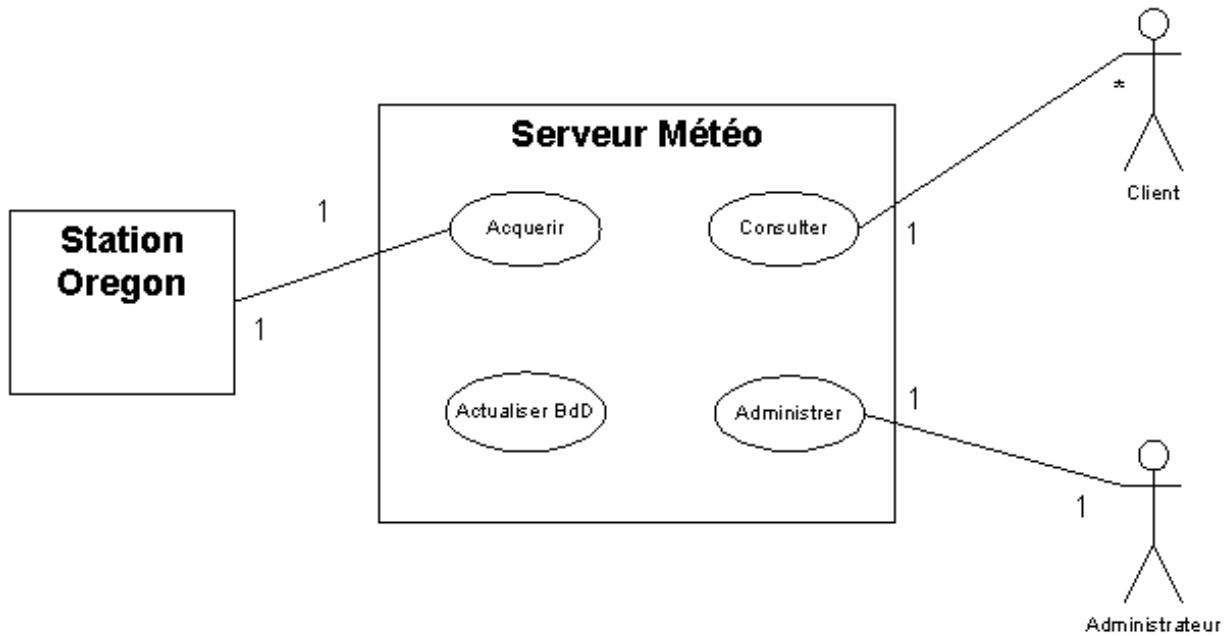


Figure 17 : Cas d'utilisation

Le serveur permet donc :

- D'acquérir les données envoyées par la station.
- D'actualiser la base de données avec les mesures.
- De consulter les pages Web contenant les relevés.
- D'administrer la base de données via un outil adéquat.

Dans ce projet mon travail se porte donc sur les actions « Consulter » et « Administre »

II-4 Diagramme de séquence

Le diagramme de séquence est une modélisation temporelle du système. Il permet de comprendre comment s'effectue les échanges entre les différents éléments.

II-4-1 Administrer la base de données

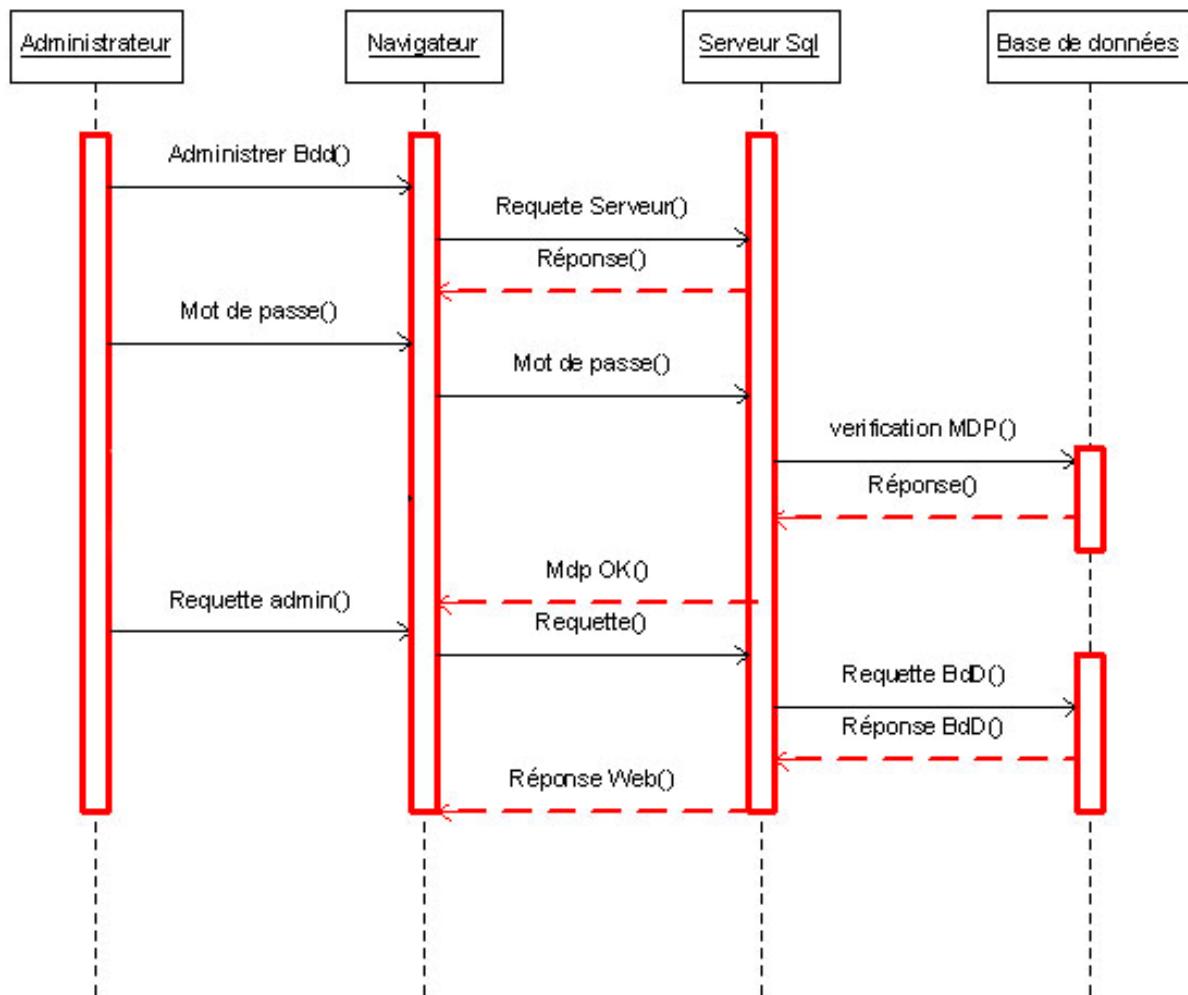


Figure 18 : Séquence – Administration Bdd

On peut voir ici que l'administration de la base par l'administrateur se fait via un navigateur Web.

L'administrateur effectue une requête sur le serveur Web pour accéder au serveur SQL. Un mot de passe lui est alors demandé. Si ce mot de passe correspond l'administrateur peut effectuer des requêtes SQL toujours via une interface Web. La requête est alors interprétée par le serveur SQL qui agit sur la base de données. Pour finir l'opération réalisée par l'administrateur est confirmée par un message, qui peut également être un message d'erreur si la requête n'est pas correcte.

II-4-2 Consulter une page Web

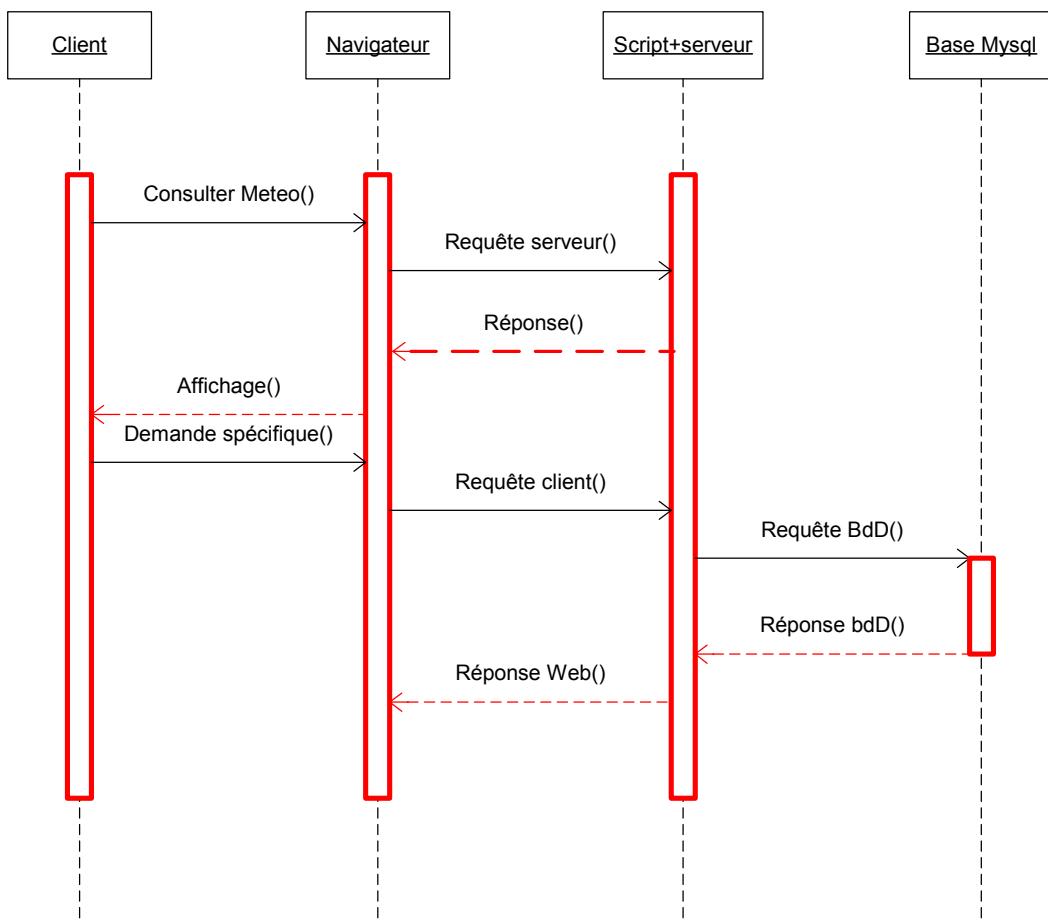


Figure 19 : Séquence – Consultation page Web

Le procédé de consultation d'une page Web est presque identique au procédé d'administration car ils déroulent de la même manière. La seule différence majeure est le fait que l'accès aux pages ne soit pas protégé contrairement à la base de données.

On peut remarquer que les requêtes SQL ne sont générées que si la page consultée par l'utilisateur le nécessite. La communication peut donc très bien s'effectuer entre le navigateur et le serveur Web sans passer par le serveur SQL.

III- Choix effectués

III-1 Solutions techniques

La principale question lors de l'analyse du problème fut la communication entre le programme d'acquisition et les différents scripts PHP permettant d'afficher les relevés météo. Dans un premier temps nous souhaitions utiliser un fichier temporaire, qui permettrait de stocker la dernière mesure relevée. Ce fichier aurait été alors lu par un script PHP pour finalement écrire son contenu dans la base de données.

Il s'est vite avéré que cette solution n'était pas appropriée puisque le fichier ne tolère pas les accès simultanés. C'est-à-dire qu'il n'aurait pas été possible de lire lorsqu'une écriture avait lieu et inversement.

Très rapidement le choix s'est orienté vers l'utilisation de la base de données MySQL tant au niveau du logiciel d'acquisition que des scripts PHP car cette base de données accepte les connexions simultanées et on peut donc sans aucun problème effectuer diverses opérations en même temps.

III-2 Matériel

Comme précisé dans le cahier des charges nous disposons de 2 postes de travail ainsi qu'un serveur. Ces 3 postes sont identiques, cependant le serveur pourra être différent lors de l'utilisation finale du projet

III-2-1 Système d'exploitation et services

Chaque poste de travail est équipé d'un système Windows 2000 Professionnel. Pour le serveur j'ai opté pour la version « Server » de Windows 2000. Ce choix me paraît justifié par l'utilisation que l'on souhaite faire du poste, c'est-à-dire stocker des données et afficher des pages Web.

Windows 2000 Server possède tous les services nécessaires à la mise en place d'un serveur Web. C'est pourquoi je n'ai pas utilisé Wamp5 comme indiqué dans le cahier des charges mais IIS, PHP et MySQL installé manuellement. Le principal intérêt est d'avoir un serveur Web complètement intégré au système d'exploitation sans services inutiles et donc plus léger et stable.

III-3 Outils de développement

Afin de mener à bien ce projet j'ai effectué certains choix logiciels. Rien n'étant imposé j'ai choisi d'utiliser des logiciel que je connaissais déjà et qui dans la mesure du possible étaient gratuits.

III-3-1 Services utilisés

Les services utilisés pour la réalisation du serveur Web complet sont donc en concordance avec le cahier des charges, sauf le serveur http qui est IIS 4 en lieu et place d'apache.

Pour l'interprétation des pages PHP j'ai opté pour la dernière version en date de l'installation c'est-à-dire PHP 5.12.

La base de données est quant à elle supportée par MySql 4.17. Elle est administrable grâce au script PHPmyAdmin 2.70.

PHPmyAdmin est un script en PHP qui permet de facilement créer des bases de données, des tables ou encore des champs.

Afin de transférer les fichiers du poste de développement vers le serveur j'utilise le service FTP inclue dans IIS.

Mis à part PHP5, aucune version des outils n'était spécifiée c'est pourquoi j'ai opté pour les dernières disponibles (et stables) afin d'offrir une certaine pérennité au système.



III-3-2 Codage

Afin de réaliser la partie codage du projet j'ai utilisé un logiciel gratuit à coloration syntaxique : Notepad++. Il présente le très gros avantage de gérer l'indentation et ce pour de nombreux langages.



III-3-3 Graphisme

Le logiciel de graphisme utilisé devait me permettre de réaliser une maquette du site afin d'en extraire les éléments graphiques nécessaire à la construction du site.

Pour cela j'ai choisi le logiciel Adobe Photoshop qui est malheureusement payant mais qui offre une période d'essai de trente jours amplement suffisante.



III-3-4 Divers

Pour finir j'ai utilisé d'autres logiciels tels que Filezilla qui est un client FTP me permettant de transférer les fichiers de mon poste de travail vers le serveur.

J'ai également utilisé différents navigateurs tel Opéra, Netscape, Firefox, Internet Explorer 6, Konqueror et Safari.

IV- Travail réalisé

IV-1 Travaux préliminaires

Ne disposant pas de serveur pouvant accueillir les pages web et la base de données, il m'a fallut installer un poste avec le système d'exploitation Windows 2000 serveur.

Pour se faire j'ai du vérifier dans le BIOS l'ordre de démarrage des périphériques afin que l'ordinateur puisse démarrer sur le cd de Windows 2000. Une fois cette précaution prise on peut commencer l'installation de Windows 2000 serveur.

IV-2 Installation du serveur

IV-2-1 Partitionnement

Le partitionnement permet de diviser le disque dur en différentes parties bien distinctes. Avec l'outil de partitionnement disponible à l'installation de Windows j'ai choisi de diviser le disque dur de 20 Go de la manière suivante :

- 4 Go pour le système (format NTFS)
- 10Go pour les données (format NTFS)
- 6 Go pour les sauvegardes (format FAT)

On obtient donc 3 parties complètement indépendantes qui permettent une restauration rapide du système en cas de problème grave

IV-2-2 Installation du système.

Après avoir partitionné le disque on lance l'installation. Elle ne s'interrompt que ponctuellement afin de demander certaines précisions à l'utilisateur comme le numéro de série du produit ou encore le type de licences. On choisit un type de licences « par serveur » c'est-à-dire que pour chaque client se connectant à Windows 2000 server nous devrons disposer d'une autorisation. Cela détermine le nombre de connexions simultanées accepté par le serveur.

Au cours de l'installation divers autres paramètres sont à spécifier tel que le nom du serveur (STATION_METEO) ou encore le fuseau horaire utilisée. Mais tout ceci reste commun à n'importe quelle installation de Windows.

IV-2-3 Vérification et sauvegarde

Une fois l'installation terminée il est impératif de s'assurer que le serveur est opérationnel. C'est-à-dire qu'il est utilisable et que le réseau est correctement configuré. Pour vérifier l'état du fonctionnement du réseau, on contrôle tout d'abord la bonne configuration des périphériques (les deux cartes réseau) puis on utilise la commande PING afin de tester la présence d'autres machines sur le réseau. Si la commande ne retourne pas d'erreur et donc que les autres machines ont été détecté c'est que le serveur est prêt à l'emploi.

Avant de poursuivre il faut impérativement effectuer une sauvegarde du système à l'aide du logiciel GHOST afin d'être en mesure de repartir d'une installation saine très rapidement en cas de problème.

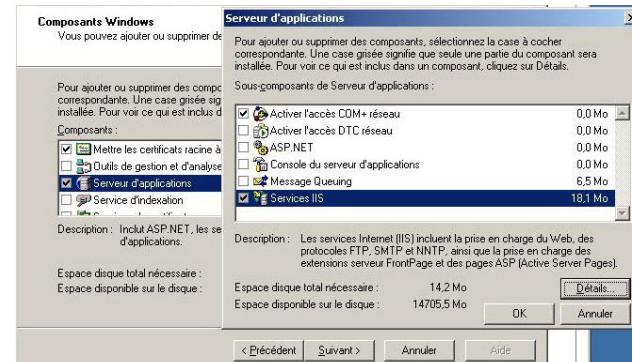
IV-3 Installation des services

Le serveur à pour but de stocker et utiliser des données sur un ou plusieurs réseaux. Il me fallait donc installer les divers services nécessaires à ces fonctions.

IV-3-1 IIS

IIS (Internet Information Services) étant un service de Windows 2000 son installation est très simple. Il suffit de se rendre dans les composants Windows via le panneau de configuration puis de choisir d'installer IIS.

Une fois cette installation terminée le serveur est en mesure d'afficher des pages web simples en revanche le serveur n'est pas encore capable d'interpréter des pages du type .php ou .asp, il faudra pour cela installer l'interpréteur approprié.



IV-3-2 Mysql

La base de données imposée dans le cahier des charges est Mysql. J'utilise ici la version 4.17 qui est la dernière version stable en date de l'installation. L'installation est très simple si l'on utilise le setup disponible sur le site de Mysql. Il suffit de suivre la procédure et de répondre à quelques questions pour que Mysql se configure automatiquement et puisse cohabiter avec IIS. Lors de l'installation on choisit d'exécuter Mysql en tant que service afin que les bases de données soit accessibles même sans qu'une session ne soit ouverte sur le serveur.



Figure 20 : Installation de Mysql

IV-3-3 PHP

L'installation de PHP est plus compliquée que les deux services précédents puisque le seul installateur disponible est loin d'être complet. Il faut donc décompresser l'archive complète de PHP dans un dossier (Ex : d:\php5) puis intégrer PHP à IIS afin qu'il soit exécuté en mode ISAPI (Internet Service API) et que IIS sache interpréter les pages PHP. Pour plus de détail au sujet de l'installation se référer à l'annexe I : *Recette d'installation de PHP5*.

IV-3-4 PHPMyAdmin

PHPMyAdmin est un script PHP permettant d'administrer un serveur Mysql. Grâce à lui on peut par exemple facilement ajouter des utilisateurs, des bases de données et effectuer toutes les opérations possibles concernant.

L'installation est très simple puisqu'il suffit de décompresser l'archive dans le dossier du serveur web. Toutefois il peut être nécessaire d'activer l'extension mbstring (fonctions de manipulation des chaînes) de PHP dans le fichier php.ini.

Une fois l'installation terminée on sécurise l'accès à PHPMyAdmin afin que seul les utilisateurs enregistrés y aient accès.

Par défaut le compte root de Mysql n'a pas de mot de passe, il faut donc lui en attribuer un et par la même occasion créer un utilisateur « programme » qui ne sera utilisé que par le logiciel d'acquisition.

IV-4 Détermination de la présentation des pages

IV-4-1 Réflexion

Ce projet étant avant tout destiné à des collégiens il était indispensable de réaliser des pages Web à la fois claires et intuitives mais qui dispensent suffisamment d'informations pour les utilisateurs plus aguerris.

C'est pourquoi j'ai décidé de réaliser une première page affichant les 3 mesures principales (températures, pluviométrie et vent) et d'autres pages affichant des données plus détaillées.

IV-4-2 Réalisation

La réalisation des pages web s'est effectuée en 3 temps :

Tout d'abord une première réflexion sur un papier permet de se donner une idée de la forme que prendront les pages web



Figure 21 : Croquis des pages web

Une fois le croquis terminé il faut passer à la seconde étape de la réalisation. Elle consiste à représenter les pages web comme elles apparaîtront à l'utilisateur. Pour cela j'ai utilisé le logiciel Photoshop qui m'a permis de réaliser une première image.

Cette première image subit alors très vite une modification afin de mieux répondre aux attentes exposées précédemment.

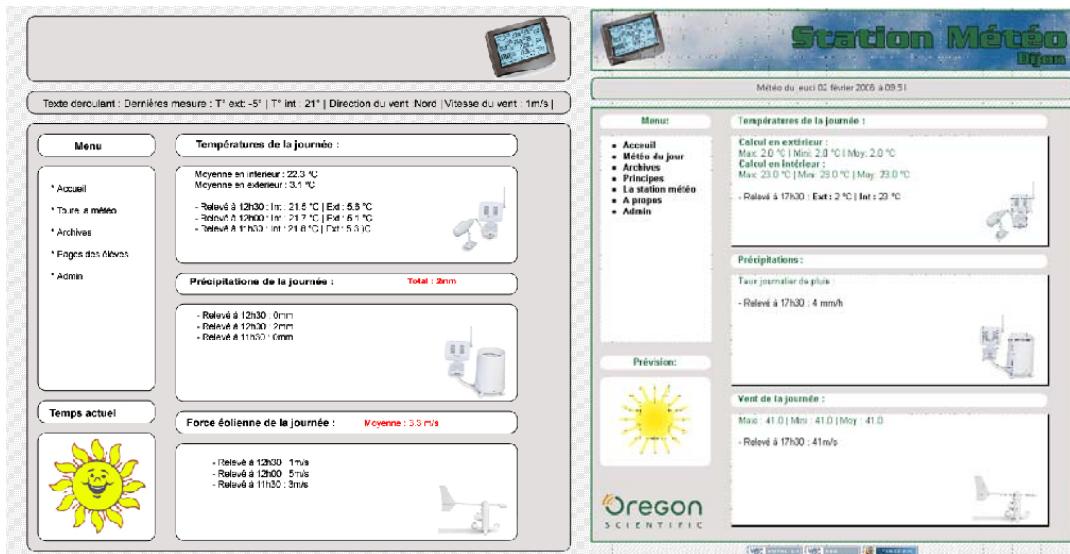


Figure 22 : Evolution de la page web

Pour le moment les pages web n'existe que sous forme d'images, il faut donc faire en sorte de les transformer en page html.

J'ai donc choisi de coder les pages web en xHTML/CSS. Cette norme contrairement au HTML normal permet de différencier le fond de la forme. La page html contiendra uniquement la structure alors qu'un fichier annexe (fichier css) contiendra tous la mise en page.

Une page html ce divise en 3 parties :

- La déclaration de la page symbolisée par les balises <html></html>
- L'entête symbolisée par les balises <head></head>
- Le corps symbolisé par les balises <body></body>

Le lien entre la page html et son fichier de mise en page s'effectue dans l'entête à l'aide de la commande :

```
<link rel="stylesheet" media="screen" type="text/css" title="home" href="style.css" />
```

La structure de la page se définit donc dans le corps s'est à dire entre les balises <body> et </body>. Sur l'image réalisée précédemment on peut voir plusieurs grandes parties :

- La partie supérieure
- La barre en dessous
- Le menu de gauche
- Le cadre de prévision
- L'affichage des données dans le centre

C'est de cette manière que la page html va être codée. Chaque grande partie sera délimité dans un bloc (`<div></div>`) qui portera un nom propre. On obtient donc une page html avec la structure suivante :

```
<body>
<div id="header">
<!--Entête de la page-->
</div>
<div id="deroulant">
    <div class="texte_deroul">
        <!--Contenu sous l entête-->
    </div>
</div>
<div id="global">
    <div id="menu">
        <div class="titre_menu">
            <!--Titre du menu-->
        </div>
        <div class="menu">
            <!--Contenu du menu-->
        </div>
        <div class="titre_temps">
            <!--titre du bloc prévision-->
        </div>
        <div class="temps">
            <!--contenu du bloc prévision-->
        </div>
    </div>
    <div id="corps">
        <div class="titre_temperature">
            <!--titre du bloc température-->
        </div>
        <div class="temperature">
            <!--contenu du bloc température-->
        </div>
        <div class="titre_precipitation">
            <!--titre du bloc pluie-->
        </div>
        <div class="precipitation">
            <!--contenu du bloc pluie-->
        </div>
        <div class="titre_vent">
            <!--titre du bloc vent-->
        </div>
        <div class="vent">
            <!--contenu du bloc vent-->
        </div>
    </div>
</div>
</body>
```

On obtient alors une page au code clair et facilement exploitable cependant cette page seule ne représentera rien à l'écran. Il faut donc réaliser le fichier css (cue sheet style) qui contient toutes les directives de mise en page.

Dans ce fichier chaque élément déclaré dans la page html est défini avec ces propriétés. Par exemple le bloc `<div id=header>` sera défini dans le fichier css par :

```
#header{}
```

Voici la définition complète du bloc header afin de mieux comprendre le fonctionnement du fichier css :

```
#header
{
    height: 85px; /* hauteur du bloc en pixel*/
    width: 700px; /* Largeur du bloc en pixel*/
    background-color: #E2DEDD; /*Couleur de fond*/
    background-image: url("images/header.gif");/* Image de fond*/
    border: 2px solid #2A7A48; /*Bordure de 2 pixel trait continu*/
    margin-left: auto; /* centrage horizontal automatique*/
    margin-right: auto; /* centrage horizontal automatique*/
    margin-top: 10px; /*Marge exterieur supérieur de 10pixels */
}
```

On voit donc que le bloc est entièrement paramétré. On lui spécifie sa taille, sa couleur, son positionnement et même l'image de fond. Le fichier css est donc lu par la page web à chacun de ses affichages afin que la mise en page soit effective.

Une fois le fichier css terminé on obtient alors une page web au format html identique à l'image réalisée. Le prototypage des pages est donc terminé.

IV-4-3 Respect des contraintes.

Dans le cahier des charges il est spécifié que les pages web doivent être compatibles avec les navigateurs internet actuels. J'ai donc ouvert la page réalisée avec différents navigateurs et systèmes d'exploitation.

FireFox 1.5 (Linux /Win)	Internet Explorer 6	Internet Explorer 7
Safari (MacOSX)	Netscape 7	Opera 8
Epiphany (Linux)		

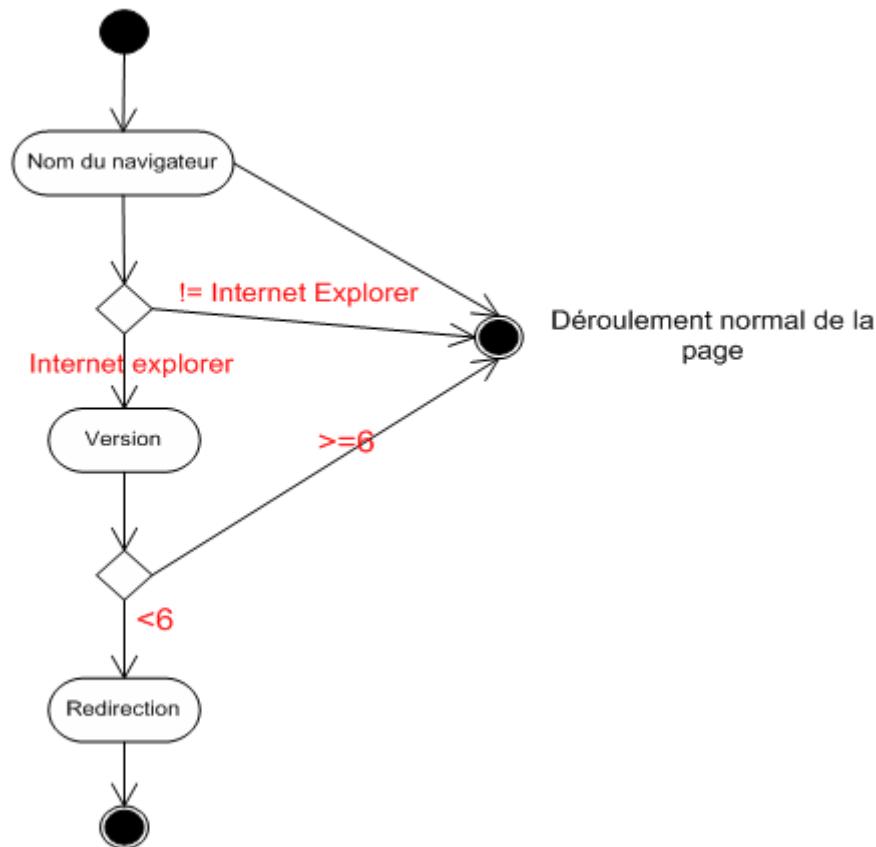
On voit donc grâce à ces illustrations que la page web est entièrement compatible avec tous les navigateurs actuels ce qui répond au cahier des charges.

IV-4-4 Problème rencontré

La compatibilité avec les navigateurs récents est complète cependant, il existe un problème avec internet Explorer 5 qui équipe actuellement moins de 3% des ordinateurs. Afin de ne pas pénaliser les autres utilisateurs qui eux ont un navigateur récent, j'ai réaliser un mini site accessible aux utilisateurs d'IE5.

Pour qu'ils y aient accès il est donc indispensable de détecter la version du navigateur utilisée par les clients qui consultent le site.

Voici comment procéder :



Ce diagramme d'état transition est ensuite traduit en JavaScript.

Dans un premier temps on recherche le nom du navigateur :

```
nav = navigator.appName //Récupération du type de navigateur
```

Puis si le nom de navigateur est internet explorer on recherche la version :

```

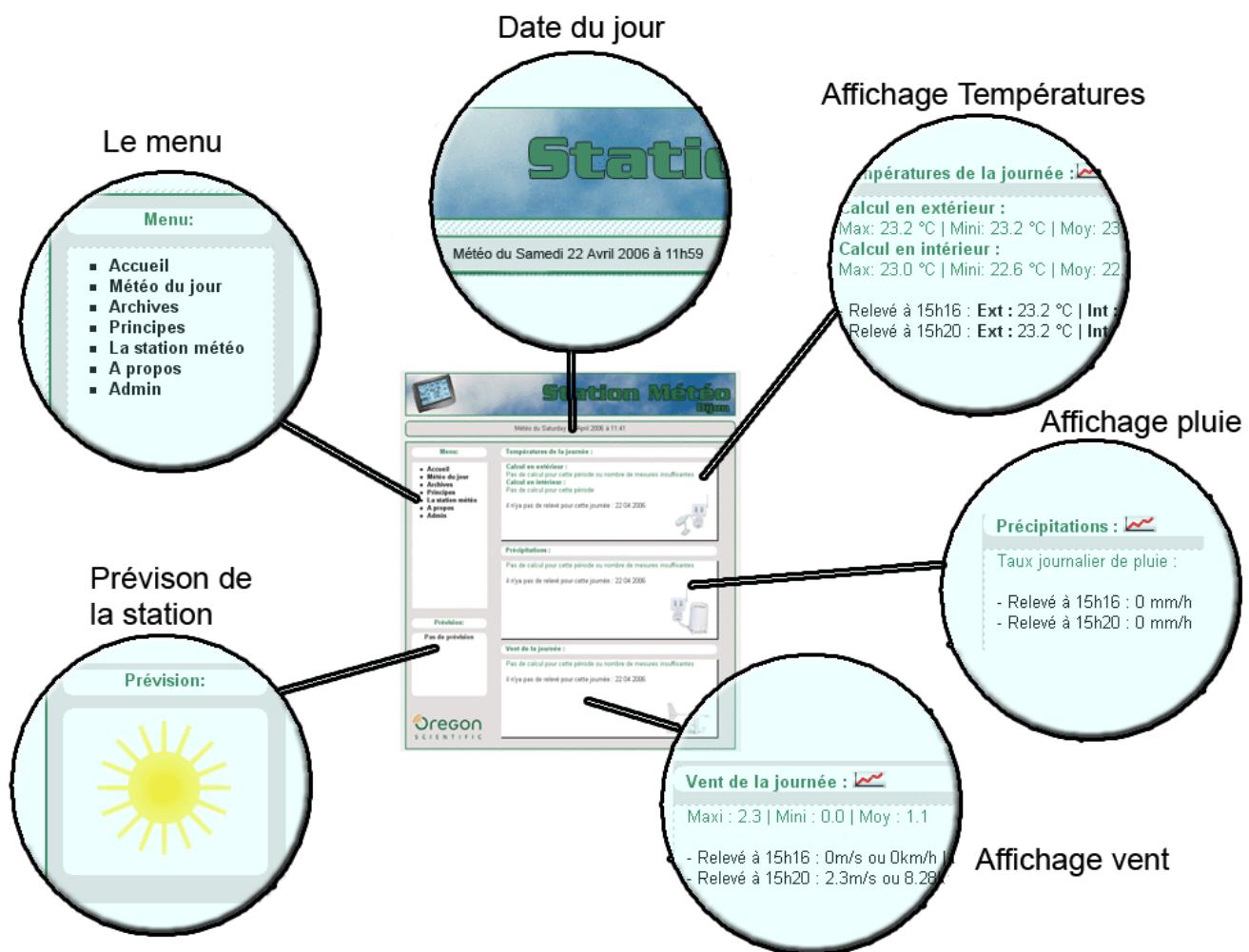
if(nav=="Microsoft Internet Explorer")//Si le navigateur est Internet Explorer
{
    //Détection de la version internet Explorer

    if(navigator.appVersion.indexOf("MSIE")!=-1) //Situé la sous chaîne MSIE dans la chaîne appVersion
    {
        t=navigator.appVersion.split("MSIE")//Supprime MSIE
        ieversion=parseInt(t[1]) // Retourne l'entier correspondant à la version
    }
    // Redirection si ie5

    if (ieversion<6) // Si la version est inférieure à 6 on redirige
    {
        window.location="/ie5/index.php" // Sur un index spécial
    }
}

```

Une fois la version du navigateur extraite de la chaîne, on redirige ou non l'utilisateur.

IV-4-5 Présentation de la page Web

IV-5 Réalisation de la base de données

IV-5-1 Contraintes

Suite à l'étude de la documentation fournie et aux résultats de la simulation de la station j'ai été amené à réaliser une première base de données de test.

Dans un premier temps la base de données avait été créée pour simplement accueillir les mesures de la station, puis après réflexion nous avons décidé d'y inclure les éventuelles erreurs de fonctionnement que pourrait rencontrer le programme ou la station.

IV-5-2 Analyse

Ayant connaissance des différentes données à enregistrer il m'a fallut décider comment serait organiser la base de données.

J'ai dans un premier temps opté pour une base de données avec plusieurs tables que j'ai modélisé de cette manière :

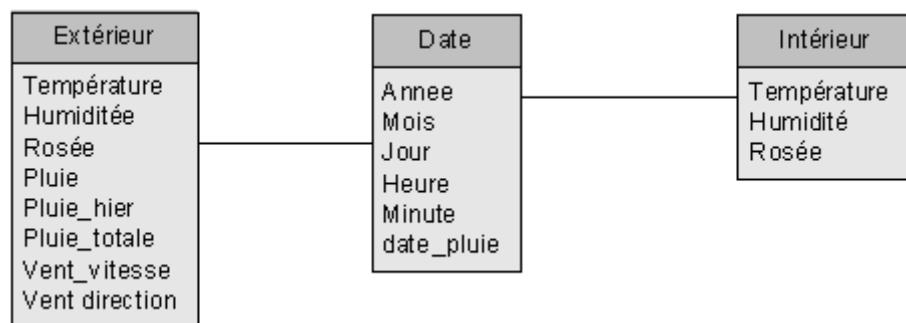


Figure 23 : Première modélisation bdd

Comme on peut le voir cette base de données ce constitue de plusieurs tables reliées entre elles. Les mesures externes et les mesures internes étaient alors différenciées. Chaque mesure étaient alors en relation avec la table Date afin qu'elles soient horodatées.

Cette structure de table c'est très vite révélé inadaptée car peu aisée à utiliser et à administrer.

J'ai donc décidé de revoir complètement la structure de la base et de ne travailler qu'avec une seule table comportant plusieurs champs

mesure
id (medium int)
annee (year)
mois (char(2))
jour (char(2))
heure (char(2))
minute (char(2))
batt_temps (binay)
tempext (float)
batt_temp_ext (binary)
tempint (float)
batt_temp_int (binary)
humext (tinyint)
humint
temproseeext
temproseeint
ventvitesse (float)
ventdirection (smallint)
batt_vent (binary)
pression (smallint)
pluiejournee
pluiehier
pluietetale
pluieannee (year)
pluimois (tinyint)
pluiejour
batt_pluie (binary)
prevision (char(1))
erreur (tinyint)

Figure 24 : Modélisation finale bdd

Chaque enregistrement dans la table mesure sera donc composé de toutes les données envoyées par la station. Le principal avantage est donc que l'on à pas besoin de passer d'une table à l'autre pour connaître la totalité d'un enregistrement. Une table unique facilite également la programmation puisque que l'on ne réalisera qu'une seule requête au lieu de 3 lorsque l'on voudra lire ou écrire des données.

IV-5-3 Optimisation

La base de données étant potentiellement amenée à recevoir de nombreuses information il était indispensable de faire en sorte que chaque enregistrement soit optimisé et donc prenne le moins de place possible sur le disque dur.

La base de données comporte une table et 28 champs. Le type de chaque champ doit donc être choisi en fonction du type de données et de l'utilisation des données.

Par exemple le champ id qui est la clé d'identification aurait pu être déclaré en int mais cela est inutile car :

Un int = 4octets ce qui donne : $2^{(4*8)} = 4294967296$ valeurs possibles. Sachant que la station écrit un enregistrement toutes les 30minutes et en supposant qu'elle fonctionne tous les jours cela représenterais : $4294967296 / (365*48) = 245146$ années d'enregistrement ce qui est totalement inutile.

Il restait donc le choix entre un mediumint (3 octets) ou un smallint (2 octets). En appliquant la même méthode que précédemment on arrive à 957 années pour le medium int et 4 années pour le smallint.

Afin d'être en mesure de supporter un éventuel changement de cadence des enregistrements (toutes les 5 minutes par exemple) j'ai opté pour le mediumint.

Certaines données ne nécessitent pas la même réflexion puisque par exemple les températures qui sont des chiffres à virgule seront obligatoirement des float.

Int	Mediumint	Smallint	Tinyint	float	year
4 octets	3 octets	2 octets	1 octet	4 octets	1 octet

Figure 25 : Principaux type de la BDD

Notre table comporte un total de 28 champs qui une fois optimisé représentent 48 octets. Une journée étant composée de 48 mesures on obtient alors un total de $48 \times 48 = 2.3$ Ko/jours. Soit pour une année 840 Ko de données ce qui est très raisonnable et permet donc d'utiliser un serveur ancien avec un disque dur de petite capacité.

IV-5-4 Création de la base de données

Pour créer la base de données j'utilise l'outil PHPMyAdmin qui facilite l'opération et par conséquent limite le langage SQL à taper. La génération de la base de données est très simple, il suffit de remplir un champ de formulaire avec le nom que l'on souhaite :

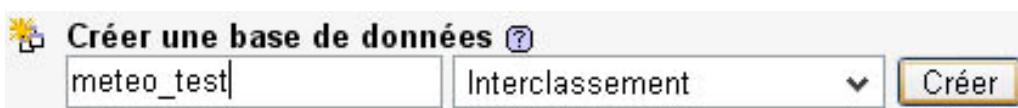


Figure 26 : Création de la base de données

La base de données « meteo_test » est alors créée mais on aurait très bien pu le faire avec cette commande SQL :

```
CREATE DATABASE `meteo_test` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
```

Il faut maintenant créer la table mesure et les champs qui la compose. Une nouvelle fois j'ai eu le choix d'utiliser PHPMyAdmin ou alors une commande SQL. J'ai opté pour le langage SQL qui est à mon goût plus rapide pour cette opération.

Il faut alors via PHPMyAdmin sélectionner la base de données que l'on vient de créer puis choisir l'onglet SQL :



Figure 27 : Sélection du langage SQL

Une fenêtre s'ouvre alors nous proposant de taper du code SQL ou de faire appel à un fichier .sql. Dans ce cas ci j'ai créé la table et les champs en tapant le code suivant :

```
CREATE TABLE `mesure` (
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,
  `annee` year(4) DEFAULT NULL,
  `mois` char(2) DEFAULT NULL,
  `jour` char(2) DEFAULT NULL,
  `heure` char(2) DEFAULT NULL,
  `minute` char(2) DEFAULT NULL,
  `batt_temps` BINARY(1) DEFAULT NULL,
  `tempext` float DEFAULT NULL,
  `batt_tempe_ext` BINARY(1) DEFAULT NULL,
  `tempint` float DEFAULT NULL,
  `batt_tempe_int` BINARY(1) DEFAULT NULL,
  `humext` tinyint(4) DEFAULT NULL,
  `humint` tinyint(4) DEFAULT NULL,
  `temproseeext` tinyint(3) UNSIGNED DEFAULT NULL,
  `temproseeint` tinyint(3) UNSIGNED DEFAULT NULL,
  `ventvitesse` float UNSIGNED DEFAULT NULL,
  `ventdirection` smallint(5) UNSIGNED DEFAULT NULL,
  `batt_vent` BINARY(1) DEFAULT NULL,
  `pression` smallint(5) UNSIGNED DEFAULT NULL,
  `pluiejournee` smallint(5) UNSIGNED DEFAULT NULL,
  `pluiehier` smallint(5) UNSIGNED DEFAULT NULL,
  `pluietotale` smallint(5) UNSIGNED DEFAULT NULL,
  `pluieannee` year(4) DEFAULT NULL,
  `pluiemois` tinyint(3) UNSIGNED DEFAULT NULL,
  `pluiejour` tinyint(3) UNSIGNED DEFAULT NULL,
  `batt_pluie` BINARY(1) DEFAULT NULL,
  `prevision` char(1) DEFAULT NULL,
  `erreur` tinyint(4) NOT NULL DEFAULT '0',
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Une fois le code exécuté on peut vérifier qu'il à été correctement interprété en sélectionnant une nouvelle fois la base de données dans PHPMyAdmin, on aperçoit alors la table et les champs.

phpMyAdmin

Serveur: localhost ▶ Base de données: meteo_test ▶ Table: mesure

[Afficher](#) [Structure](#) [SQL](#) [Rechercher](#) [Insérer](#) [Exporter](#) [Opérations](#) [Vider](#) [Supprimer](#)

Champ Type Interrangement Attribut Null Défaut Extra Action

id	mediumint(9)		Non		auto_increment							
annee	year(4)		Oui	NULL								
mois	char(2)	latin1_swedish_ci	Oui	NULL								
jour	char(2)	latin1_swedish_ci	Oui	NULL								
heure	char(2)	latin1_swedish_ci	Oui	NULL								
minute	char(2)	latin1_swedish_ci	Oui	NULL								
batt_temps	binary(1)		Oui	NULL								
tempext	float		Oui	NULL								
batt_temp_ext	binary(1)		Oui	NULL								
tempint	float		Oui	NULL								
batt_tempe_int	binary(1)		Oui	NULL								
humext	tinyint(4)		Oui	NULL								
humint	tinyint(4)		Oui	NULL								
temproughest	tinyint(5)		UNSHAMED	Oui	NULL							
temprouseste	tinyint(5)		UNSHAMED	Oui	NULL							
ventdresse	float		UNSHAMED	Oui	NULL							
ventdirection	smallint(5)		UNSHAMED	Oui	NULL							
batt_vent	binary(1)		Oui	NULL								
pression	smallint(5)		UNSHAMED	Oui	NULL							
pluiejoueme	smallint(5)		UNSHAMED	Oui	NULL							
pluieheuer	smallint(5)		UNSHAMED	Oui	NULL							
pluototale	smallint(5)		UNSHAMED	Oui	NULL							
pluaceane	year(4)		Oui	NULL								
pluemosos	tinyint(3)		UNSHAMED	Oui	NULL							
pluieur	tinyint(3)		UNSHAMED	Oui	NULL							
batt_pluie	binary(1)		Oui	NULL								
precision	char(1)	latin1_swedish_ci	Oui	NULL								
erreur	tinyint(4)		Non	0								

[Tout modifier](#) [Tout déclencher](#) [Rechercher](#)

IV-6 Exploitation des données

Le site présentant plusieurs pages les données sont utilisées de différentes manières. Soit elle son simplement affichées soit on effectue des recherches de maxi/min ou encore des calculs de moyenne.

N'ayant dans un premier temps pas de mesures réelles à utiliser nous avons ajouté quelques enregistrement à la base de données manuellement puis à l'aide du programme de simulation de la station. La phase de test des scripts PHP n'aura donc pas été réalisée avec des valeurs provenant de la station. Cependant le fonctionnement du site (et donc des différents scripts) a été validée en condition réel.

IV-6-1 Fichier de configuration

Chaque utilisation des données nécessite une connexion à la base de données. Cette connexion requiert différentes informations. C'est pourquoi il est très pratique d'utiliser un fichier de configuration que l'on inclura dans chaque page ayant besoin d'accéder à la base de données.

Ce fichier contient les identifiants de connexion à la base ainsi que les requêtes de connexion :

```
$host = "localhost"; //Serveur
$user = "root"; // utilisateur
$pass = ""; // Mot de passe
$db = "meteo_test"; // base de donnée

///////////////
//connexion
/////////////
$connect=mysql_connect($host,$user,$pass) or die ('<b>Erreur :</b>'.mysql_error());
mysql_select_db($db, $connect) or die ('<b>Erreur :</b>'.mysql_error());
```

Ce fichier permet de simplifier la connexion mais également sont utilisation puisqu'un utilisateur n'utilisant pas les même paramètres de connexion n'aura qu'un fichier à modifier et non pas plusieurs.

IV-6-2 Affichage simple

La première portion de script réalisée permet l'affichage des derniers résultats sur la page d'accueil. Il faut donc rechercher dans la base de données les enregistrements correspondant à la date du jour.

Pour cela il nous faut inclure le fichier de connexion à la base de données :

```
require('includes/infodb.php');
```

A partir de cet instant la connexion à la base de données est établie. Il faut alors sélectionner les données désirées.

On souhaite voir apparaître les mesures de la journée, il faut donc tout d'abord déterminer quel est ce jour. Pour cela on fait appel à la fonction date() de PHP :

```
$jour=date("d");
$mois=date("m");
$annee=date("Y");
```

On récupère donc dans 3 variables différentes : le numéro du jour, du mois et de l'année. On va donc pouvoir utiliser ces variables dans la requête à envoyer à la base de données.

La requête SQL doit donc sélectionner les données dans la base « meteo_test », dans la table « mesure » ou la date correspond avec celle du jour. Cela se traduit :

```
$requete = "SELECT * FROM `mesure` WHERE jour=$jour and mois=$mois and
annee=$annee ORDER BY id DESC LIMIT 10";
$envoi = mysql_query($requete) or die ('Erreur sur la requête
<b>'.$requete.'</b> : <span style="color: red;">'.mysql_error().'</span>');
$nb_result = mysql_num_rows($envoi);
```

La variable « \$requete » contient toute la requête SQL que le serveur recevra. Cette requête est envoyée par la variable « \$envoi » qui utilise la fonction mysql_query(). Pour finir le nombre de résultat de la requête est retourné dans la variable « \$nb_result ».

La sélection faite il faut maintenant l'utiliser (si le nombre de résultats n'est pas nul) afin d'afficher les valeurs souhaitées.

Pour cela on réalise une boucle tant que l'on a des résultats. A chaque passage dans cette boucle on associe à une variable une valeur que l'on affiche.

Algorithme correspondant :

```
DEBUT
    FAIRE « Connexion BDD »
    FAIRE « Sélection données »
    SI « Résultat == NULL »
        ALORS AFFICHER « erreur »
    SINON
        FAIRE TANT QUE « $i<$nb_result »
        FAIRE « variable=valeur de la base »
        AFFICHER « variable »
        FIN TANT QUE
    FIN SI
FIN
```

Voici un exemple d'utilisation de cet algorithme qui permet d'afficher les températures extérieur et intérieur ainsi que l'heure de leur mesure :

```

$nb_result=0;
if($nb_result==0)
{
    echo" il n'ya pas de relevé pour cette journée : $jour $mois $annee";
}
else
{
    while ($i<$nb_result)// tant que l'on à des résultats
    {
        // On récupère dans des variables les données de la base
        $id=mysql_result($envoi,$i,"id");
        $heure=mysql_result($envoi,$i,"heure");
        $minute=mysql_result($envoi,$i,"minute");
        $longeur=strlen("$minute");
        if($longeur==1)
        {
            $minute="0$minute";
        }
        $temp_ext=mysql_result($envoi,$i,"tempext");
        $temp_int=mysql_result($envoi,$i,"tempint");
        // On affiche ces variables
        echo '<a href="releve_detail.php?id='.$id.'"- Relevé à "'.$heure.'. 'h'. '$minute'.
        ': <b>Ext :</b> "'.$temp_ext.'. '°C | <b>Int :</b> "'.$temp_int.'. '°C</a>';
        echo"<br />";
        $i++;
    }
}

```

La fonction **mysql_result()** permet d'affecter le résultat retourné par la base à une variable. Il aurait également été possible d'utiliser un tableau associatif avec la fonction **mysql_fetch_array()**.

Les dernières instructions comprise dans **echo"** ; permet l'affichage d'une phrase contenant les différentes variables avec les valeurs de la table. Cette phrase est en fait un lien qui offre un relevé complet après avoir cliqué dessus.

Voici le résultat qu'obtiendra l'utilisateur :

Figure 28 : Affichage de données

Si les valeurs correspondent aux valeurs que l'on à entré dans la base de données, on peut très facilement valider le fonctionnement du script d'affichage.

IV-6-3 Calculs

L'affichage des données est intéressant puisqu'il permet une vision instantanée des mesures mais il peut être utile d'avoir une tendance sur la journée et donc de calculer les maxi / mini et moyenne des mesures effectuées.

Ces calculs peuvent être effectués directement par une requête MySql qui déterminera seul les bonnes valeurs. Cela présente 2 avantages :

- La facilité d'utilisation.
- Un code bien plus simple du à l'utilisation de fonctions spécifiques.

Tous les calculs sont effectués dans un fichier nommé calcul.php qui est inclus uniquement dans les pages qui en ont besoin.

Algorithme d'un calcul :

```

DEBUT
    SI « résultats présent pour la journée »
        ALORS
            FAIRE « Sélection du maxi »
            FAIRE « Sélection du mini »
            FAIRE « Calcul de la moyenne »
            FAIRE « Stocker résultat dans un tableau »
            FAIRE « Attribuer résultat à variable »
        SINON
            FAIRE « Informer utilisateur »
        FIN SI
FIN

```

Voici l'exemple du calcul des températures maximum, minimum et moyenne d'une journée :

```

$result = mysql_query("SELECT max(tempext) as maxi, min(tempext) as mini, avg(tempext)
    as moy FROM mesure WHERE jour=$jour and mois=$mois and annee=$annee"); // Requête SQL
$val = mysql_fetch_array($result); // Tableau
$temp_ext_maxi=$val['maxi'];
$temp_ext_maxi=number_format($temp_ext_maxi,1); //Formatage à 1 chiffre après la virgule
$temp_ext_mini=$val['mini'];
$temp_ext_mini=number_format($temp_ext_mini,1);
$moy_ext=$val['moy'];
$moy_ext=number_format($moy_ext,1);

```

On peut voir dans la requête que l'on sélectionne le max de tempext et qu'il pourra être utilisé en tant que maxi. De cette manière lorsque l'on réalise un tableau avec les résultats, on pourra retrouver la valeur maximale de la manière suivante : **mon_tableau['maxi']**.

La méthode est la même pour les minimum avec le mot clé **min** et pour la moyenne avec le mot clé **avg** (average).

On remarquera dans la portion de code précédente l'utilisation de la fonction **number_format()** qui permet de limiter le nombre de chiffres après la virgule.

Dans le fichier calcul on retrouvera des calculs pour quasiment toutes les valeurs (températures, vent, pression...) ce qui permet de réaliser une page de statistiques pour la journée :



Figure 29 : utilisation des calculs

IV-6-4 Gestion des archives

La fonction principale de la base de données étant de stocker durablement les mesures effectuées il était indispensable de proposer aux utilisateurs un système de recherches dans les archives de mesures.

J'ai choisi de proposer deux types de recherche :

- Une recherche par jour Ex : Le 12 janvier 2006
- Une recherche par période Ex du mois de janvier au mois de mars 2006
- Une recherche par année Ex : de 2006 à 2007

Cette recherche s'effectue grâce à des listes déroulantes qui permettent à l'utilisateur de choisir ce qu'il souhaite rechercher :

The screenshot shows a user interface for meteorological archive search. At the top, it says 'Archives Météorologiques' and 'Veuillez choisir votre type de recherche :'. Below this are two sections for date selection:

- Date de recherche :** Contains dropdown menus for 'Jour' (Day), 'Mois' (Month), and 'Année' (Year), followed by a 'Recherche' button.
- Période de recherche :** Contains dropdown menus for 'De : Mois' (From Month), 'A : Mois' (To Month), and 'Année' (Year), followed by a 'Recherche' button.

To the left of these sections is a vertical scrollable list of dates from 12 to 31, with '22' currently selected. The entire interface is contained within a light gray box.

Figure 30 : Utilisation des archives

Une fois la date choisie l'utilisateur valide sa recherche. C'est à ce moment qu'intervient une requête SQL qui va aller rechercher les résultats correspondant à la date souhaitée.

Dans le cas d'une recherche sur un jour précis la requête est exactement la même que celle précédemment utilisée. La seul différence est que les variables ne sont plus déterminées par la fonction date mais par l'utilisateur.

En revanche la recherche sur une période est un peu différente puisque l'on va sélectionner les données contenant un mois supérieur ou égal au premier mois sélectionné et inférieur ou égal au deuxième mois sélectionné et où l'année correspond à l'année demandé. On a donc comme précédemment 3 conditions de sélection :

```
$requete = "SELECT * FROM `mesure` WHERE mois>=$mois1_periode AND mois<=$mois2_periode AND annee=$annee_periode ORDER BY id DESC";
```

Une fois la recherche validée par l'utilisateur, la requête SQL lui retourne une réponse. Soit elle est négative car aucune mesure n'a été trouvée pour la date souhaitée soit elle positive

auquel cas on lui indique le nombre de mesures effectuées pendant la période ou le jour de recherche. On affiche également trois liens pour la température, le vent et la pluie qui lui permettront de visualiser les courbes.

Période de recherche :			
De :	Janvier	A :	Avril
	<input type="button" value="▼"/>		<input type="button" value="▼"/>
			2006
	<input type="button" value="▼"/>		<input type="button" value="▼"/>
			Recherche

Il y'a 172 résultat(s) pour la période du 01 / 2006 au 04 / 2006
 Consulter les courbes de : **Température** | **Pluie** | **Vent** |

Figure 31 : Résultat d'une recherche

IV-6-5 Génération de courbes

Comme dis précédemment les courbes peuvent être générées depuis les archives mais aussi depuis la page d'index lorsque le nombre de mesures est suffisant. Donc si plus de deux relevés ont été effectué un icône  apparaît et permet de visionner les courbes relatives. Les courbes sont réalisées à l'aide de la librairie Jpgraph (<http://www.aditus.nu/jpgraph/>) qui permet la génération de nombreux graphiques.

Pour l'utiliser il suffit de décompresser l'archive Jpgraph dans un dossier de notre serveur web et d'inclure le fichier jpgraph.php et le fichier correspondant au type de courbe que l'on souhaite réalisée comme par exemple : jpgraph_line.php pour des courbes linéaires.

Le fichier graphique.php qui générera les courbes doit être en mesure de générer 3 types de courbes différentes et déterminer si c'est la courbe d'une journée ou d'une période.

Pour se faire on appelle le fichier graphique.php en lui donnant divers arguments comme par exemple :

```
graphique.php?jour=$jour&mois=$mois&annee=$annee&mesure=temperature&selec=jour
```

De cette manière le fichier connaîtra la date à utiliser le type de mesure (ici température) et le type de sélection (ici jour).

Algorithme de fonctionnement du fichier de génération des courbes :

DEBUT

LIRE « paramètres »

SI « recherche=jour »

ALORS

 FAIRE « requête de sélection du jour souhaité »

SINON

 FAIRE « requête de sélection de la période souhaité »

FIN SI

SI « type= température » ALORS

 FAIRE « générer courbe température »

SINONSI « type=pluie »

 FAIRE « générer courbe pluie »

SINON

 FAIRE « générer courbe vent »

FIN SI

FIN

Il faut donc dans un premier temps récupérer les paramètres passé au fichier pour cela on utilise la méthode GET qui permet de lire les paramètres situés dans l'adresse du fichier :

```
$jour = $_GET['jour'];
$mois = $_GET['mois'];
$annee = $_GET['annee'];
$type = $_GET['mesure'];
$select = $_GET['selec'];
```

A partir de ces données on effectue alors une requête spécifique selon si le type de recherche est journalière ou périodique.

Puis après cette requête on détermine le type de données. C'est à partir de cette détermination que commence la configuration du graphique. Un exemple avec la température :

```
if($type=="temperature") // Si courbe = température
{
    if ($donnee = mysql_fetch_array($envoi)) // Si valeurs
    {
        do
        {
            $mesure[] = $donnee["tempext"]; //On place les valeurs dans le tableau température
            $mesure2[] = $donnee["tempint"]; //On place les valeurs dans le tableau température
        }while($donnee = mysql_fetch_array($envoi));// Tant que valeurs
    }
    // Définition des variables d'affichage
    $titre="Graphiques des températures";
    $titreX="Mesure N°...";
    $titreY="T en °C";
    $legend="Temp Extérieure";
    $legend2="Temp Intérieure";
}
```

On peut voir que comme précédemment on place les valeurs de la base de données dans un tableau. Mais on voit surtout les premières variables qui concernent les courbes. On définit alors en fonction du type de mesures les titres et les légendes des graphiques. De cette manière chaque graphique est adapté au type de mesures.

Le reste du fichier graphique.php est la création à proprement parlé du graphique et est de la programmation objet puisque Jgraph se base sur près de 105 classes et 1056 méthodes.

La première des opérations à réalisée est la récupération des valeurs à utiliser :

```
$ydata = $mesure;
```

Le contenu du tableau \$mesure sera donc utilisée pour les données en Y sur le graphique. Il faut ensuite créer le graphique et donc lui allouer un espace mémoire :

```
$graph = new Graph(800,400);
```

On à créer ici un graphique d'une taille de 800*400 pixel de manière dynamique (mot clé new).

Il faut alors associer une échelle à ce graphique. Plusieurs choix sont possibles s'est pourquoi il est intéressant de l'adapter en fonction du type de recherche. Une recherche périodique donnant souvent beaucoup de résultat il vaut mieux une échelle en X (nombre de mesure) linéaire qui s'affichera par exemple de 10 en 10 si il y'a 100 mesures. Au contraire pour une journée de mesure il vaut mieux une échelle textuelle donc qui affichera les mesures une à une :

```
if($nb_point>48)
{
    $graph->SetScale ("lin");      // Echelle
}
else
{
    $graph->SetScale ("textlin");
}
```

Après cela vient tout une partie de configuration du graphique que l'on pourrait qualifier d'esthétiques puisqu'elle permet de régler tout les aspect visuel du graphique tel que la couleur des textes et courbes , les légendes , les marges

Exemple :

```
$graph->img->SetMargin(60,180,30,40); // Définition des marges G,D,H,B
$graph->title->Set("$titre"); // Titre général du graphique
$graph->title->SetColor("red"); // Couleur du titre
$graph->xaxis->title->Set("$titreX"); // Titre axe des X
$graph->yaxis->title->Set("$titreY"); // Titre axe des y
$graph-> title->SetFont(FF_FONT1 ,FS_BOLD); // Propriété de la police du titre
$graph->yaxis-> title->SetFont(FF_FONT1 ,FS_BOLD); // Propriété de la police Y
$graph->xaxis-> title->SetFont(FF_FONT1 ,FS_BOLD); // Propriété de la police X
$graph->SetShadow(); // Affiche une ombre autour du graphique
$graph->xgrid->Show(); // Affiche la grille verticale
```

Il ne reste alors plus qu'a générer le système de point, l'ajouter au graphique et afficher le graphique :

```
$lineplot=new LinePlot($ydata); // Génération des point
$graph->Add($lineplot); // Ajout des points
$graph->Stroke(); // Affichage du graphique
```

On obtient alors un graphique adapté en fonction du type de mesure et du type de recherche le tout avec un seul fichier de génération :

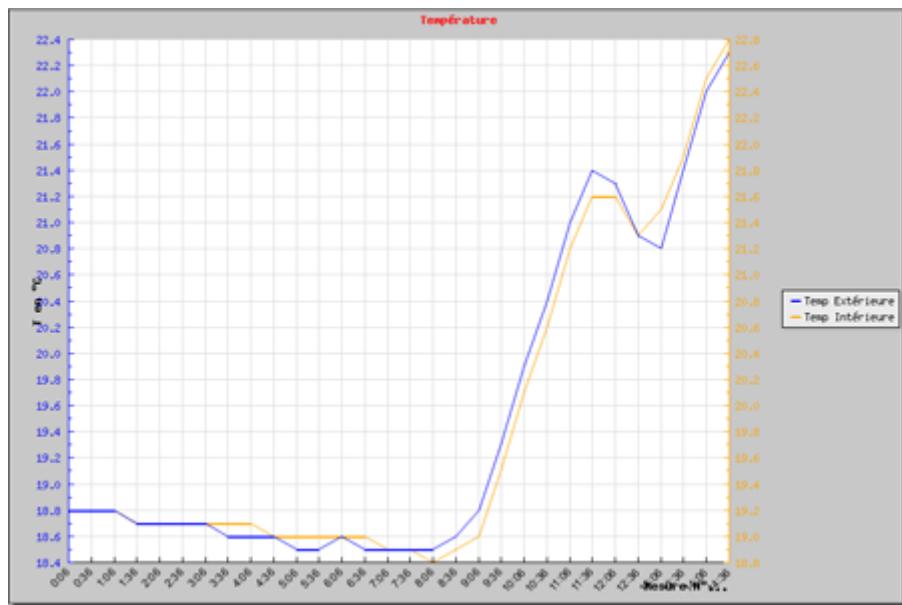


Figure 32 : Graphique sur une journée

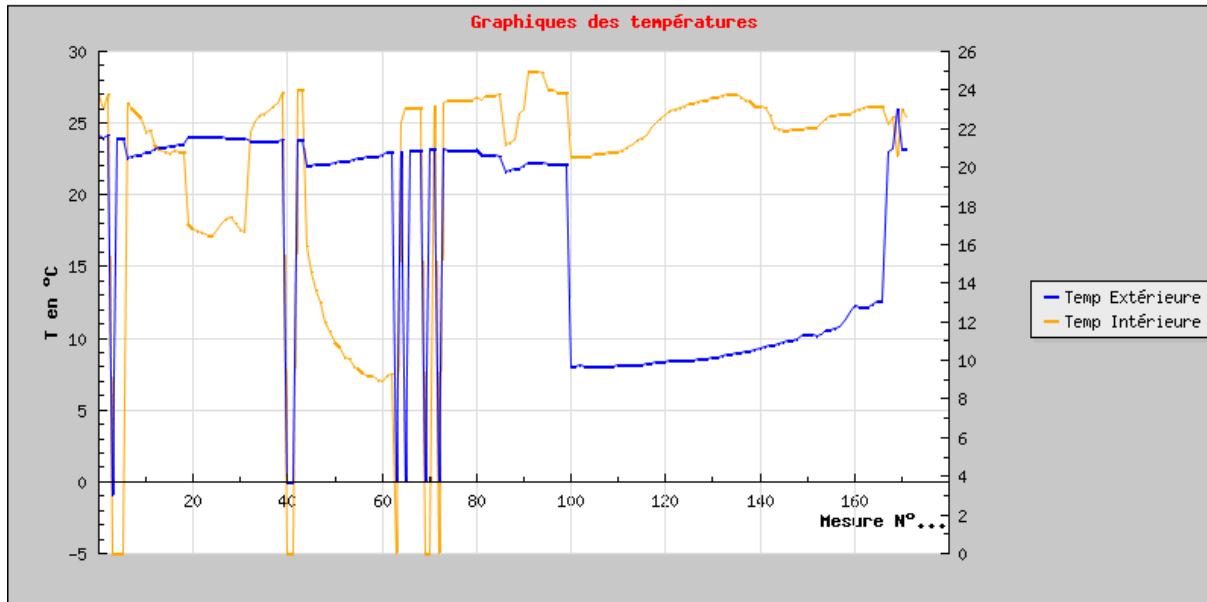


Figure 33 : Graphique sur une période

IV-6-6 Récapitulatif des fonctionnalités du site

Le site offre donc différentes fonctionnalités aux utilisateurs, en voici un récapitulatif ainsi que les méthodes de validation de ces fonctions et les difficultés rencontrées.

- Affichage des données

Le site permet d'afficher des données, de manière concise comme sur la page d'index ou plus détaillé lorsque l'on clique sur un relevé.

Afin de valider le bon affichage des données j'ai tout simplement remplis 2 champs de la base de données puis comparé les données affichées aux données contenues dans la base.

- Calcul des données

Il était intéressant d'avoir une vue globale des relevés de la journée en proposant des calculs de maxi, mini et moyenne. On peut une nouvelle fois voir les calculs sur la page d'index de manière simplifiée et de manière plus complète sur les statistiques météo de la journée.

Encore une fois la validation s'est effectué en entrant quelques valeurs dans la base de données puis j'ai comparé les calculs affichés aux calculs que j'avais réalisés manuellement.

- Recherche d'archives

Pour valider la recherche d'archive j'ai utilisé l'enregistrement précédent puis j'ai effectué une recherche sur le jour concerné. J'ai eu dans un premier temps quelques difficultés avec la recherche par période car la requête présentait quelques erreurs (utilisation de OR au lieu de AND) mais la solution fut rapidement trouvée.

- Génération de courbes

La génération des courbes fut la partie la plus longue à mettre en œuvre car, JPGraph est une librairie entièrement en anglais, il donc fallut que je m'approprie correctement la documentation fournie avant de commencer quoi que ce soit.

J'ai rencontré de nombreuses difficultés notamment avec les échelles ou encore pour arriver à placer deux courbes avec des axes distincts dans une même fenêtre. Mais une lecture détaillée de l'aide m'a permis de solutionner les problèmes.

- Administration

La partie administration du site est à accès protégé et permet de visualiser l'état des batteries, accéder à PHPMyAdmin, visualiser les erreurs ou encore modifier les derniers relevés.

IV-6-7 Simulation de fonctionnement

Afin de valider le fonctionnement du site et des archives j'ai réalisé un script qui permet d'écrire 2 ans de données dans la base. Cela représente 26899 valeurs. Je peux ainsi vérifier la recherche de valeurs via le formulaire et la génération de courbes.

Ce test a mis en évidence un problème au niveau de la génération de courbes. En effet un fonctionnement constant de la station est à la source de 1500 valeurs enregistrées dans la base

de données. Ces 1500 valeurs ne peuvent être affichées correctement, il faudrait un graphique d'une taille très importante ce qui n'est absolument pas adapté à l'affichage sur un écran d'ordinateur. Je n'ai malheureusement pas trouvé de solutions appropriés pour ce problème. Cependant on pourrait doubler voir tripler le temps d'écriture dans la base de données via le programme ce qui permettrait de réduire considérablement le nombre de point sur les courbes sans pour autant nuire à la qualité de l'information.

IV-7 Mise en ligne du site

Le site était jusqu'à maintenant accessible uniquement via réseau local, il est donc nécessaire de connecter le serveur à internet pour rendre le site accessible à tous.

IV-7-1 Sécurité

Avant toute connexion vers le monde extérieur il est indispensable de sécurisé le serveur web ainsi que le serveur lui-même afin qu'il ne soit pas sujet aux attaques virales.

Le premier point pour la sécurité du serveur est d'utiliser au minimum deux partitions afin de séparer les données du système. De cette manière même en cas d'intrusion sur la partition du serveur web le système peut encore être sain.

Afin de sécuriser IIS il y'a plusieurs opération à réaliser :

Il faut supprimer tous les sites inutilisés mais également tous les fichiers ou dossiers suivant qui présentent de potentielles failles de sécurité :

```
C:\WINNT\Help\IISHelp  
MSADC C:\Program Files\CommonFiles\System\msadc  
IISAdmin C:\WINNT\System32\InetSrv\IISAdmin  
Printers C:\WINNT\Web\Printers  
RPC C:\WINNT\System32\RpcProxy
```

La faille la plus connue d'IIS est l'accès à cmd.exe donc à l'invite de commande du système en tapant l'adresse :

```
http://www.mondomaine.com/scripts/..%25c..%25cwinnt/system32/cmd.exe?/c+dir+c:\
```

Cette faille peut très simplement être corrigée par la suppression du dossier : c:\InetPub\scripts.

Grâce à ces premières modifications le serveur IIS est protégé contre les intrusions de pirates. Le serveur Mysql doit également être protégé, il suffit pour cela de s'assurer que le compte root ai bien un mot de passe défini afin que seules les personnes le connaissant est accès aux bases de données. Le serveur web est donc sécurisé.

Il est cependant indispensable d'utiliser un firewall afin de bloquer tous les ports de la machines sauf ceux qui sont réellement nécessaires.

Pour cela j'ai choisi d'utiliser Isafer qui est un firewall gratuit et qui fonctionne sur Windows 2000 serveur ce qui est le cas de très peu de firewall gratuit.

Le firewall me permet de bloquer tous les ports du serveur sauf :

- Le port 80 pour les requêtes http
- Le port 21 pour l'utilisation du serveur ftp
- Le port 3306 pour Mysql

J'ai complété la sécurité du système avec l'antivirus Avast Serveur en version d'essai de 60 jours qui permet de se prémunir des virus.

Une fois tout ceci vérifié j'ai effectué les mises à jour système critiques via Windows Update afin que le système soit à jour et que les failles de sécurité soient corrigées.

IV-7-2 Configuration réseau

Le serveur utilisé est composé de deux cartes réseau, l'une intégrée sur la carte mère est configurée pour être utilisée sur le réseau local de la salle et donc par conséquent le site est déjà accessible aux utilisateurs de ce réseau.

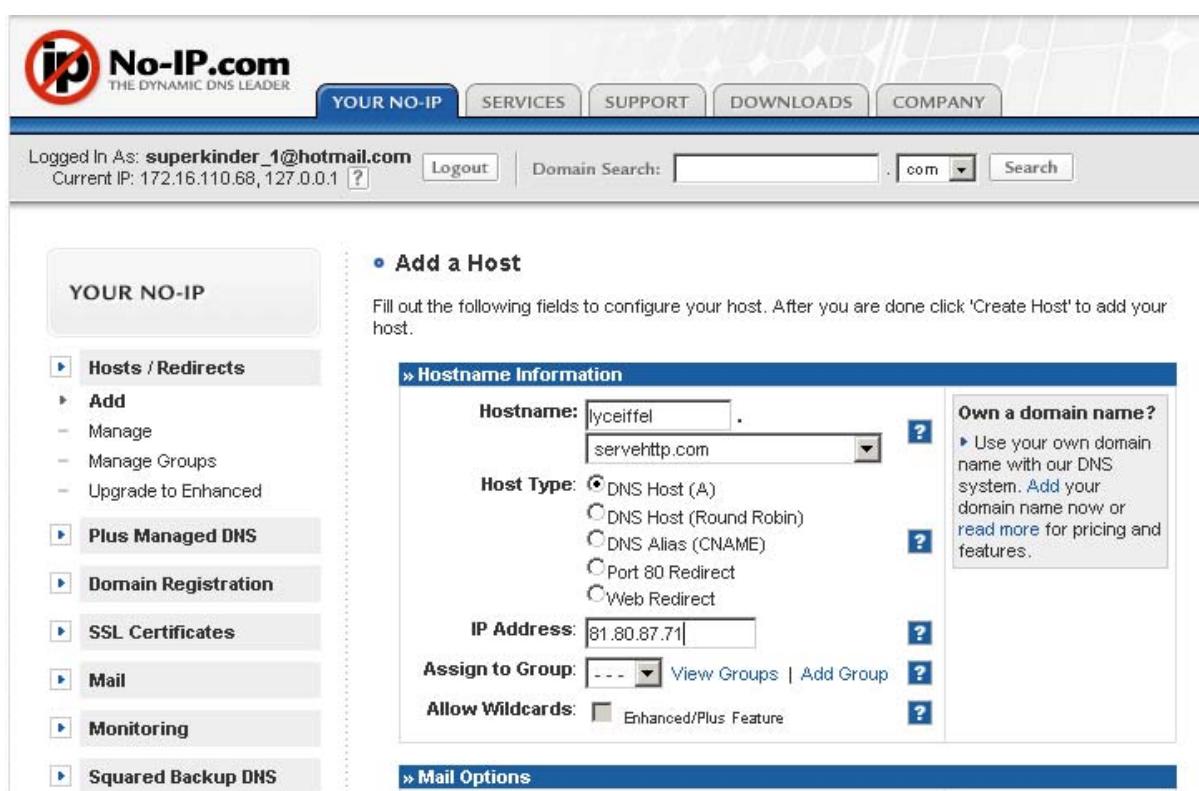
L'autre est une carte PCI qui doit être configurée sur l'IP fixe qui nous est attribué :

IP : 81.80.87.81
MASQUE : 255.255.255.248
DNS 1 : 194.2. 0.20
DNS 2 : 194.2.00.50
Passerelle : 81.80.87.86

Une fois la carte configurée le serveur est accessible via internet par l'adresse : <http://81.80.87.81>

Cependant l'utilisation d'une adresse IP comme adresse n'est que peut pratique, c'est pourquoi j'ai décidé d'utilisé une redirection gratuite qui permet d'associer à cette IP un nom de domaine.

Il suffit pour cela de se connecter au site <http://www.no-ip.com> et d'ouvrir un compte afin de choisir une redirection :



The screenshot shows the No-IP.com website interface. At the top, there's a navigation bar with links for 'YOUR NO-IP', 'SERVICES', 'SUPPORT', 'DOWNLOADS', and 'COMPANY'. Below the navigation, it says 'Logged In As: superkinder_1@hotmail.com' and 'Current IP: 172.16.110.68, 127.0.0.1'. There's also a 'Logout' button and a search bar.

The main content area has a sidebar on the left with links for 'Hosts / Redirects' (selected), 'Plus Managed DNS', 'Domain Registration', 'SSL Certificates', 'Mail', 'Monitoring', and 'Squared Backup DNS'. The main panel title is 'Add a Host'. It contains a sub-section titled 'Hostname Information' with fields for 'Hostname' (set to 'lyceiffel' and 'servehttp.com'), 'Host Type' (set to 'DNS Host (A)'), 'IP Address' (set to '81.80.87.71'), 'Assign to Group' (a dropdown menu), and 'Allow Wildcards' (checkbox). To the right of this form is a box titled 'Own a domain name?' with descriptive text about using your own domain with their DNS system.

Figure 34 : Redirection de l'IP

De cette manière le site est accessible à l'adresse <http://lyceiffel.servehttp.com> ce qui est plus pratique que l'adresse IP qui reste néanmoins utilisable si on le souhaite.

IV-8 Mise en place d'une DMZ et d'un firewall

Ayant pu terminer la partie du projet me concernant légèrement en avance j'ai décidé de mettre en place une DMZ (Zone démilitarisée) et un firewall afin que le réseau local soit totalement sécurisé.

Une zone démilitarisée est une zone « tampon » accessible depuis l'extérieur et située dans un réseau local. Elle est protégée par un firewall. Son principal intérêt est qu'elle permet de mettre à disposition des serveurs vers le monde extérieur (Internet) sans mettre en danger le réseau local puisqu'il n'est pas accessible depuis l'extérieur.

IV-8-1 Analyse

Afin de mettre en place cette sécurité j'ai choisi d'utiliser une distribution linux nommée IPCOP qui est une distribution très légère (moins de 50Mo) dont la seule fonction est de réaliser des firewalls, DHCP et ou DMZ.

Avant toute mise en place j'ai modélisé ce que je souhaitais réaliser :

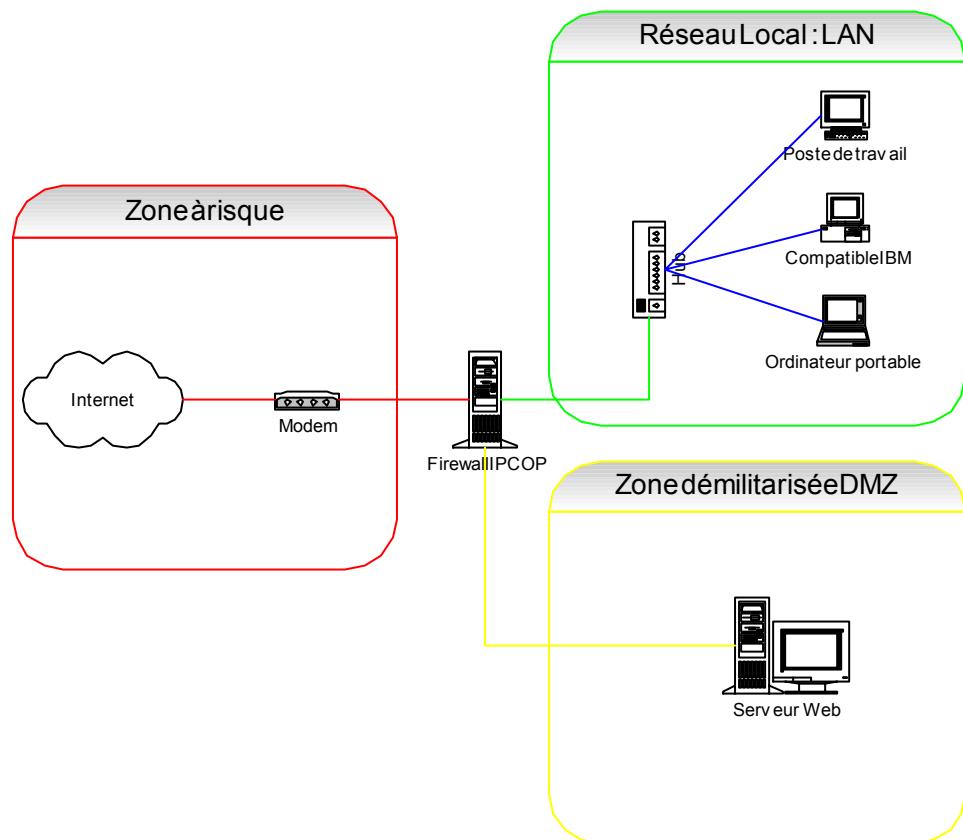


Figure 35 : Modélisation de la DMZ

Grâce à cette modélisation, j'ai pu me rendre compte que le pc que j'utiliserais pour réaliser la DMZ devra être doté de 3 cartes réseaux. Une pour Internet, une pour le réseau local et une pour la DMZ.

IV-8-2 Mise en service

Il faut tout d'abord installer la distribution linux, pour cela on télécharge une image CD que l'on grave.

Nous avons vu que nous avions besoin de 3 cartes réseaux compatibles avec la distribution Linux. C'est pourquoi en plus de la carte réseau intégrer Intel Pro 100 j'ai installé 2 cartes réseau de type SN2000.

Une fois les cartes installées on installe IPcop. Au cours de l'installation une détection des carte réseau à lieu et on les attributs aux interfaces désirées (ROUGE, VERT, ORANGE).

IV-8-3 Interface de gestion

Le serveur est administrable à distance via une interface Web :

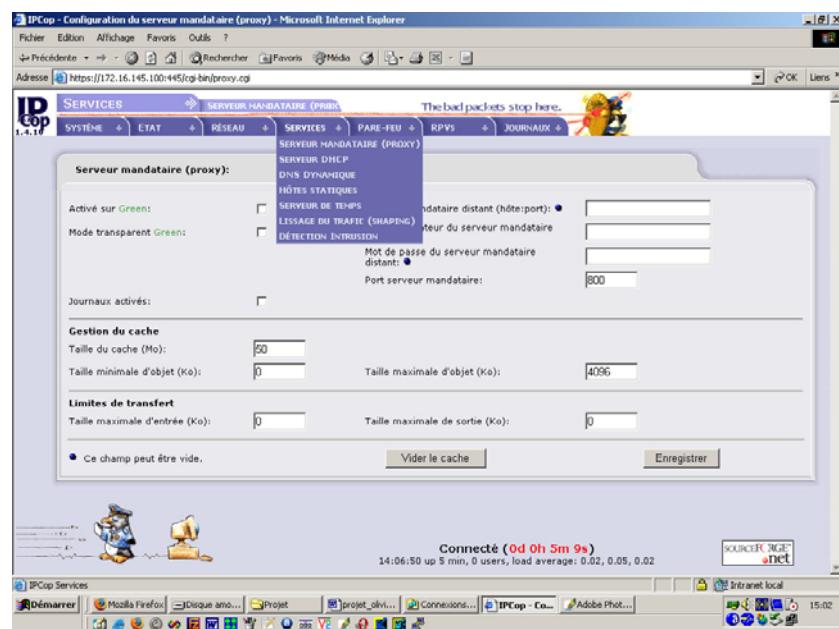


Figure 36 : Interface de gestion Ipcop

Comme on peut le voir le serveur peut également servir de Proxy ou de DNS. Toutes ces fonctionnalités peuvent bien entendu être combinées.

L'interface de gestion du firewall permet de gérer tous les accès au serveur et par conséquent au reste du réseau.

On peut par exemple faire de la redirection de port (port fowarding) qui permet de renvoyer toutes les requêtes reçues par le firewall sur un ordinateur spécifique. On peut également gérer les accès externes (autorisée telle ou telle adresse) ainsi que la DMZ.

IV-8-4 Problèmes rencontrés

Le premier problème rencontré lors de l'installation du serveur est le choix des cartes réseaux. En effet il est très difficile d'utiliser 2 cartes réseau identiques puisqu'on ne peut différencier les cartes que par leur adresse MAC.

Dans un premier temps j'ai voulu opter pour une configuration simple pour me familiariser avec IPcop. J'ai donc utilisé une interface rouge (internet) et une interface verte (local) avec gestion du firewall.

Ayant réussi cette opération j'ai voulu passer à la DMZ donc à une utilisation (VERT ROUGE ORANGE) mais malheureusement je n'ai pas obtenu le résultat escompté.

En effet depuis le réseau local j'avais bien accès à l'interface verte du serveur mais il m'était impossible de passer dans l'interface orange, donc de me connecter au serveur Web.

Par manque de temps je n'ai pas pu me documenter suffisamment à ce sujet pour parvenir à réaliser un firewall ainsi qu'une DMZ.

Une solution nettement plus rapide que j'aurais pu utiliser est l'utilisation d'un routeur ADSL qui généralement propose les mêmes services de manière plus accessible.

V- Conclusion

Au cours de ce projet j'avais pour objectif d'afficher des données météorologiques stockées dans une base de données créée et administrée par mes soins.

La première difficulté rencontrée fut la création de la base de données afin qu'elle soit facilement exploitable tant par moi que par mon binôme.

Afin de générer les courbes j'ai été confronté à une librairie que je n'avais jamais utilisé, il m'a donc fallut me documenter afin d'être en mesure de réaliser des graphiques.

Pour finir la dernière difficulté rencontrée fut la mise en place de la DMZ que je n'ai malheureusement pas réussi à mener à bien.

Malgré les différents obstacles rencontrés j'ai été en mesure de fournir au collège Champollion un site Internet leur permettant de consulter les mesures enregistrées dans la base. Comme demandé dans le cahier des charges la consultation peut être effectuée de manière textuelle ou alors à partir de graphique.

Une des améliorations qu'il serait intéressant de développer est de réaliser des courbes statistiques. Par exemple sur une année on réalise des statistiques précises pour chaque mois, puis on trace des courbes à partir de ces statistiques.

On pourrait également imaginer un site Internet « communautaire » regroupant les relevés de tous les utilisateurs de station Oregon et de notre logiciel afin d'établir des archives météorologiques de toutes la France.

Ce projet m'a permis d'appliquer des connaissances personnelles, mais j'ai également pu découvrir de nouvelles choses tel que la gestion d'un firewall et d'une DMZ via une distribution Linux et non pas par un routeur. Je regrette simplement de ne pas avoir eu assez de temps pour mener à bien la sécurisation du réseau grâce à IPCOP.

VI-ANNEXES :

VI-I Recette d'installation PHP5

VI-1-1 Installation

Le mode Isapi permet à IIS de charger PHP comme s'il faisait partie intégrante d'IIS. Il est donc chargé en mémoire une seule fois. Le mode CGI, au contraire, force IIS à appeler une nouvelle instance de PHP à chaque demande de page PHP. Vous pouvez utiliser CGI si vos appels aux pages PHP sont rares. Je recommande cependant d'utiliser le mode Isapi dans tous les cas.

Décompresser l'archive et copier son contenu dans d:\web\php5 ce dossier sera le dossier de travail avec PHP.

Créer le répertoire **php5\sessions**, il servira au stockage des fichiers de sessions.

Copier les fichiers suivant dans les répertoires indiqués :

- php5isapi.dll et php5ts.dll dans c:\windows\system32\inetsrv
- libmysql.dll dans c:\windows\system32
- php.ini-recommandé dans c:\windows\ en le renommant en php.ini
- php5isapi.dll est le fichier qui fait le « pont » entre IIS et le parser à proprement parler, php5ts.dll.
- libmysql.dll permet d'utiliser MySQL avec PHP, ce qui, avec la version 5 n'est plus implémentée en standard.
- php.ini est le fichier de configuration de PHP.

VI-1-2 Configuration

I-2-1 PHP

Ouvrir le fichier php.ini et l'éditer comme suit :

```
error_reporting = E_ALL  
display_errors = on
```

Permet l'affichage des erreurs

```
register_globals = Off
```

On désactive les variables globales par sécurité

```
extension_dir = "d:\web\php5\ext\"
```

On spécifie le dossier contenant les extensions de PHP

```
extension=php_mysql.dll
```

On précise l'utilisation de Mysql en tant que base de donnée

```
session.save_path = "d:\php5\sessions\"  
session.auto_start = 0
```

Répertoire de stockage des sessions

I-2-1 IIS

Ouvrez une console d'administration des Services Web (**Démarrer > Exécuter > inetmgr** ou **Démarrer > Programmes > Outils d'administration > Gestionnaire des services Internet (IIS)**).

Première étape, ajouter Php comme filtre Isapi.

Pour cela, utilisez l'onglet approprié dans les propriétés du site web.

Choisissez **Ajouter**. Donnez lui le nom **Php5 en Isapi** et sélectionner php5isapi.dll qui se trouve dans **c:\Windows\System32\InetSrv** comme Exécutable.

Il faut associer l'extension .php à notre filtre Isapi. Dans l'onglet répertoire de base, choisir **Configuration** puis **Ajouter**.

L'exécutable est le même que pour le filtre (**php5isapi.dll**), l'extension est donc .php. Les autres options sont par défaut correctes.

IIS sait maintenant que quand il rencontrera une page avec l'extension php, il devra utiliser php5isapi.dll pour la traiter

Il faut maintenant préciser que index.php est la page par défaut du site Web. Pour cela, aller dans l'onglet **Documents**, choisir **Ajouter** et saisir **index.php**. index.php apparaît alors, mais en bas de la liste. Pour être sur qu'IIS regarde d'abord les pages index.php, utiliser le bouton **Monter** jusqu'à le faire arriver en début de liste.

Dernière étape. Autoriser l'extension php. Toujours dans la console de management de IIS, faire un clic droit sur **Extensions du Site Web** et choisir **Ajouter une nouvelle extension de Service Web ...**. Donner le nom **php5** et sélectionner **php5isapi.dll** comme **Fichiers requis** et sélectionner l'option **Définir le statut de l'extension à Autorisée**. **Php5** apparaît dans la liste avec une petite coche verte normalement.

Lancer une commande MS-Dos (**Exécuter > Cmd**) et tapez-y **iisreset** pour relancer IIS et charger Php.

Afin de valider l'installation de php créer une page test.php avec comme contenu :

```
< ? phpinfo() ?>
```

Si lors de sa consultation vous obtenez des informations relatives à PHP votre installation est un succès.

Extrait de : <http://odelmotte.developpez.com/tutoriels/iis/iis6phpmysql/>

VI-2- Organisation des fichiers

Voici l'organisation des fichiers sur le serveur :

Nom	Site Distant : /meteo/	Taille	Type	Date	Heure	Permiss
..						
admin			Dossier dr...	12/05/...	22:11	drwxr-x
ie5			Dossier dr...	19/04/...	19:55	drwxr-x
images			Dossier dr...	12/05/...	21:39	drwxr-x
includes			Dossier dr...	12/05/...	22:10	drwxr-x
apropos.php	2751	PHP Script	06/04/...	08:15	-rwxr--i	
archives.php	11524	PHP Script	06/04/...	08:15	-rwxr--i	
graphique.php	5219	PHP Script	12/05/...	21:31	-rwxr--i	
index.php	3759	PHP Script	06/04/...	08:15	-rwxr--i	
install.php	5355	PHP Script	13/05/...	11:43	-rwxr--i	
meteo_jour.php	3827	PHP Script	31/03/...	13:14	-rwxr--i	
prevision.php	1015	PHP Script	03/02/...	17:09	-rwxr--i	
principes.php	3170	PHP Script	02/02/...	08:24	-rwxr--i	
releve_detail.php	3404	PHP Script	12/05/...	14:19	-rwxr--i	
station.php	3772	PHP Script	03/03/...	16:07	-rwxr--i	
style.css	6629	Cascadin...	06/04/...	08:16	-rwxr--i	

Le dossier contenant PHPMyAdmin est situé au niveau supérieur.

VI-3 Script de simulation

```
<?
$host = "localhost"; //Serveur
$user = "root"; // utilisateur
$pass = "password"; // Mot de passe
$db = "meteo_test"; // base de donnée
///////////////
//connexion
///////////////
$connect=mysql_connect($host,$user,$pass) or die ('<b>Erreur :</b>'.mysql_error());
mysql_select_db($db, $connect) or die ('<b>Erreur :</b>'.mysql_error());

$batt_temps=0;
$batt_tempe_ext=0;
$batt_tempe_int=0;
$batt_vent=0;
$batt_pluie=0;
$annee=2006;
$mois=5;
$jour=18;
$minute=0;
$heure=14;
while($annee<2008) // Tant que pas en 2008
{
    for($i=0;$i<=48;$i++) // boucle de 48 mesures
    {
        $minute += 30;
    // génération aléatoire de mesures
        $tempext=rand(-10,35);
        $tempint=rand(11,35);
        $humext=rand(1,99);
        $humint=rand(1,99);
        $temproseeext=rand(0,20);
        $temproseeint=rand(8,15);
```

```

$ventvitesse=rand(0,10);
$ventdirection=rand(0,359);
$pression=rand(980,1100);
$pluiejournee=rand(0,100);
$pluiehier=rand(0,250);
$pluietotale=2735;
$pluieannee=2001;
$pluiemois=05;
$pluiejour=12;
$prevision=rand(1,6);
$erreur=0;

if($minute==60) // Si 60 minutes sont passé
{
    $minute=0;
    $heure++; // On ajoute une heure
}
if($heure==24) // Si 24 heures
{
    $heure=0;
    $jour++; // On ajoute un jour
}

// Incrémentation du mois
if($mois==2 && $jour==28)
{
    $mois++;
    $jour=0;
}
elseif($mois==4 && $jour==30)
{
    $mois++;
    $jour=0;
}
elseif($mois==6 && $jour==30)
{
    $mois++;
    $jour=0;
}
elseif($mois==9 && $jour==30)
{
    $mois++;
    $jour=0;
}
elseif($mois==11 && $jour==30)
{
    $mois++;
    $jour=0;
}
elseif($jour==31)
{
    $mois++;
    $jour=0;
}
if($mois==12)
{
    $mois=0;
    $annee++;
}

// requete SQL d'écriture
$requete="INSERT INTO mesure VALUES(",
'$annee','$mois','$jour','$heure','$minute','$batt_temps','$tempext','$batt_tempe_ext','$tempint','$batt_tempe_int','$humext','$h
umint','$temproseeext','$temproseeint','$ventvitesse','$ventdirection','$batt_vent','$pression','$pluiejournee','$pluiehier','$pluie
totale','$pluieannee','$pluiemois','$pluiejour','$batt_pluie','$prevision','$erreur")";

$envoi = mysql_query($requete) or die ('Erreur sur la requête <b>'.$requete.'</b> : <span style="color:
red;">'.mysql_error().'</span>');
}

```

VI-4 Webographie / Bibliographie

<http://fr.php.net/> Pour la documentation php

<http://dev.mysql.com/doc/> Pour la documentation Mysql

<http://www.aditus.nu/jpgraph/> Pour tracer les courbes

<http://www.infoclimat.fr/> Pour les informations relatives à la météo

Ainsi que :

Sciences et Avenir n° 710

Publication du 01/04/2006

VI-5 Journal de bord

SEMAINE # 1

Jeudi 5 janvier

Analyse des diagrammes fournis

Réalisation de diagramme de cas d'utilisation de contexte et de séquence

Installation serveur météo 2000 serveur

➔ Drivers réseau / graphique

➔ Wamp5 (config apache : ServerName http://172.16.145.1/ UseCanonicalName Off pour accès depuis le web)

➔ Filezilla server

➔ Test de connexion

➔ Configuration phpmyadmin (config.inc.php : \$cfg['PmaAbsoluteUri'] = '172.16.145.1/phpmyadmin/'; Mise en place d'un mot de passe : privilège bdd + fichier config : auth http)

Préparation post de développement

Installation réseau

➔ Passage de câble

➔ Configuration de poste

Vendredi 6 janvier

UML : Diagramme séquence, représentation météo.

UML : Diagramme d'activité : récupération donnée, archivage.

Réflexion Base de données

SEMAINE # 2

Jeudi 12 janvier

UML : Diagramme de déploiement

BDD : Choix de modèle de base

IHM : réalisation de différentes IHM

Réalisation bdd de test

Vendredi 13 Janvier

Serveur : installation manuel des différents services : IIS 4 , php5.12 ,mysql 4.17, phpmyadmin 2.70

Test et validation

<http://odelmotte.developpez.com/tutoriels/iis/iis6phpmysql/>

Début codage xhtml css

SEMAINE # 3**Jeudi 19 janvier**

Fin codage xhtml /css

Installation ftp sur 2000 serveur (installation du service, création des droits et des utilisateurs)

Test de connexion base de données.

Simulation affichage base de données ➔ début de script php

Vendredi 20 janvier

Script php ➔ détection mini , maxi , moyenne , somme dans un champs de la base

Script php ➔ affichage détail relevé.

SEMAINE # 4**Jeudi 26 janvier**

Java script pour détection de navigateur ➔ redirection si navigateur non compatible

Changement des requêtes SQL pour sélection du jour.

Script PHP ➔ Météo du jour

Vendredi 27 janvier

Optimisation base de donnée

Test de connexion bdd avec programme en c++

Test d écriture bdd avec programme c++

Script d'affichage des prévisions météo.

SEMAINE # 5**Jeudi 2 Février**

➔ Optimisation du code (x)html/php afin de simplifier les modifications

➔ Script php : recherche d'ancienne valeur dans les archives

➔ Préparation revue

Vendredi 3 Février

➔ Développement interface Ie5

➔ Revue

SEMAINE # 6**Jeudi 9 Février**

Administration site : gestion des batteries, réflexion rapport d'erreur
Jpgraph : téléchargement et prise de contact

Vendredi 10 Février

Réalisation d'une courbe jpgraph → essai des différentes mise en forme
Application au site et à la base de données → génération des courbes de T° , Vent et pluie

SEMAINE #9**Jeudi 2 Mars**

Courbes statistiques d'une journée et d'une période.
Protection serveur (firewall+ antivirus)

Vendredi 3 Mars

Test du programme de simulation
Début de rédaction du dossier.

SEMAINE #10**Jeudi 9 mars**

Test du programme de simulation via Orphy → Utilisation de TPloader et Orphymétéo.
Début d'amélioration de l'administration du site (modif des enregistrements)

Vendredi 10 Mars

Fin de l'administration du site internet (possibilité de modifier les enregistrement sans passer par la base de données et phpmyadmin) ;

SEMAINE #11**Jeudi 16 Mars**

Revue de projet n°1
Recodage complet du script de maxi et mini due à des problèmes avec la base de données.
Ajout de bouton sur l'index pour obtenir les courbes de la journée en cours.

Vendredi 17 Mars

Fin du script de calcul
Script pour déterminer la direction du vent
Réalisation de pictogramme pour le vent

SEMAINE #12**Jeudi 23 Mars**

Test réel station météo + base de données
Mise au point de script (calcul, météo jour) pour être en accord avec les relevés de la station
Rédaction rapport commun

Vendredi 24 Mars

Rédaction rapport perso.

SEMAINE #13

Jeudi 30 Mars

Réalisation de schémas explicatifs et recherche d'information sur le DCF77.

Rédaction rapport commun.

Test d'interconnexion entre programme finale et site Internet ➔ test en condition réel de la station

Vendredi 31 Mars

Correction de quelques bugs relatif à la connexion entre les deux parties du projet

Rédaction du rapport.

SEMAINE #14

Jeudi 6 Avril

Récupération de la configuration de l'adresse Ip fixe pour sortie internet

Sécurisation du serveur (Firewall , antivirus, MAJ windows)

Installation carte Pci RS232

Mise en service du 2eme site

Problème pour accès à la zone protégée

Test de longue durée d'enregistrement de mesure

Vendredi 7 avril

Constatation d'une erreur du serveur sql (4h53)

Cours Temps réel Multitâche

SEMAINE #15

Jeudi 13 Avril

Réalisation script php pour accès administration.

Oraux économie et anglais

Vendredi 14 avril

Decision de réaliser un firewall + DMZ à l'aide d'une distribution linux

Installation de carte réseau

Installation de IPcop

SEMAINE #16

SEMAINE #17

Vacances

SEMAINE #18

Jeudi 4 mai

Réinstallation d'IPcop avec cartes réseau compatible

Configuration Ipcop : zone rouge et verte
Recherche de Doc pour DMZ

Vendredi 5 mai

Test d IPcop en configuration ROUGE ORANGE VERT

SEMAINE #19

Jeudi 11mai

Revue de projet #2
Recherche infructueuses sur IPCOP

Vendredi 12 mai

Présentation en Anglais du projet à un groupe de professeurs d'étrangers (Sicilien et Grecs)
Rédaction rapport

SEMAINE #20

Jeudi 18mai

Réussite de la mise en production
Test sur la base de données : script générant 2années de mesures
Rédaction rapport.
Préparation à l'examen

VII- Tests communs et mise en production

VI-1 Tests effectués

Outre les différents tests personnels qui ont pu être évoqués dans précédemment nous avons été obligé de procéder à des tests d'intégration de nos deux parties.

Il fallait donc voir si le comportement du logiciel était adapté à la base de données ainsi qu'au site, le contraire étant également vrai.

Le premier test fut donc de lancer le programme d'acquisition sur le serveur où se trouvait le site. Les premières erreurs ont très rapidement fait leur apparition.

Au niveau du programme, certaines variables n'étaient pas correctement écrites dans la base de données et ce à cause d'une erreur de type. Les modifications apportées (type float) ont résolu le problème.

Au niveau du site le principal problème fut la non présence de certaines données. Les tests personnels ayant toujours été effectués avec des données dans la base, le site provoquait des erreurs lorsqu'aucunes données n'étaient présentes ou si leur nombre étaient insuffisant.

Le problème fut réglé par l'ajout d'une simple condition.

N'ayant pas rencontré de problèmes majeurs lors du regroupement des deux parties du programme nous avons souhaité mettre en production le projet.

VI-2 Mise en production

La mise en production nous permet de valider le bon fonctionnement du projet dans la durée. Une station météo étant sensée enregistrer des données météo non-stop il était indispensable de lancer un test sur plusieurs jours.

Pour cela nous avons configuré le programme sur une écriture de 30 minutes dans la base de données ce qui correspond aux attentes du cahier des charges. A cette cadence, 48 mesures seront enregistrées chaque jour.

Nous avons lancé le serveur ainsi que son programme le jeudi 11 mai et ce jusqu'au jeudi 18 mai soit plus de 330 mesures enregistrées dans la base de données et ce de manière continue.

Durant cette période d'essai aucune erreur n'est survenue nous pouvons donc affirmer que le serveur, le programme et le site sont stables et peuvent être mis en production définitivement.

VII- CONCLUSION

Lors de ce projet nous avons pu utiliser 2 langages de programmation bien distincts : Le C++ qui est un langage compilé et le PHP qui est un langage interprété. Le lien entre les deux langages est effectué via la base de données Mysql.

A la fin du projet nous disposons donc d'une station météo accompagné d'un logiciel et d'un site Internet qui possèdent chacun des assistants d'installation afin qu'ils soient accessible à tous et ce de manière aisée.

Nous avons pu nous rendre compte au cours de ce projet de la difficulté du travail en groupe. En effet lorsque chacun effectue une partie du projet , il est très rare de ne pas rencontrer de problème lors de la réunion des deux parties. Cependant cela nous a obligé à mieux communiquer entre nous afin que les décisions de l'un n'entrave pas celles de l'autre.

L'intérêt de ce projet est l'économie effectuée comparée à un système déjà existant : ORPHY. En effet pour réaliser un programme d'acquisition et un site Internet nous n'avions aucun budget alloué. La seule dépense est l'achat de la station et de ses capteurs (dans le cas où l'on dispose déjà des PC). Au contraire le même système acheté dans le commerce coûte près de 1000€ ce qui est une dépense non négligeable pour l'établissement bénéficiaire de ce projet. Le principal inconvénient est le temps de développement qui peut se révéler long.

ANNEXES :**I- DCF77 : détail de la trame****0 (M) :**

Début de trame (bit à 1).

1 - 14 :

Réservé pour une utilisation future.

15 (R) :

L'émetteur de réserve est actif lorsque ce bit est à 1.

16 (A1) :

Annonce de l'heure d'hiver.

17, 18 (Z1, Z2) :

Ces deux bits codent le fuseau horaire actuel :

Z1	Z2	Fuseau horaire
0	1	CET (Central European Time) = UTC + 1h
1	0	CEST (Central European Sommer Time) = UTC + 2h

CET correspond à l'heure d'hiver, et CEST correspond à l'heure d'été. Il s'agit de l'heure légale de la plupart des pays d'Europe.

19 (A2) :

Indique qu'une seconde va être supprimée pour corriger les irrégularités de la rotation de la terre.

20 (S) :

Bit de début de codage des informations horaires (toujours à 1).

21 - 27 :

Minutes codées en BCD, bit de poids faible en premier :

N° bit	21	22	23	24	25	26	27
Valeur	1	2	4	8	10	20	40

28 (P1) :

Bit de parité (parité paire) des minutes.

29 - 34 :

Heures codées en BCD, bit de poids faible en premier :

N° bit	29	30	31	32	33	34
Valeur	1	2	4	8	10	20

35 (P2) :

Bit de parité (parité paire) des heures.

36 - 41 :

Jour codé en BCD, bit de poids faible en premier.

N° bit	36	37	38	39	40	41
Valeur	1	2	4	8	10	20

42 - 44 :

Jour de la semaine codé en BCD, bit de poids faible en premier :

N° bit	42	43	44
Valeur	1	2	4

45 - 49 :

Mois codé en BCD, bit de poids faible en premier :

N° bit	45	46	47	48	49
Valeur	1	2	4	8	10

50 - 57 :

Année (sur deux chiffres) codées en BCD, bit de poids faible en premier :

N° bit	50	51	52	53	54	55	56	57
Valeur	1	2	4	8	10	20	42	80

58 (P3) :

Bit de parité

59 :

Pas d'impulsion

PLANNING PRÉVISIONNEL

Étudiant	A	B	C	D	Repère tâche	Description de la tâche	0	0	0	0	0	0	R	R					R	R				
							1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	2	2	
X X X X	X	X	X	X	T1.1	S'approprier de la modélisation du système																		
X X X X		X	X	X	T2.1	Finaliser la modélisation du système																		
X X X X		X	X	X	T2.2	Caractériser les informations fournies par la station météo.																		
X X X			X		T2.3	Prototypage IHM présentation des pages Web (collège lycée)																		
X X X X				X	T2.6	Installation des outils de développement																		
X X					T2.10	Réaliser une émulation de la station météo																		
	X		X		T2.10	Réaliser une base de données de simulation																		
						Revue projet 1																		
X X					T3.3	Récupérer les données																		
	X		X		T3.3	Administrer la base de données																		
X X					T3.3	Effectuer la mise à l'heure système																		
X X					T3.3	Stocker les données																		
X X					T3.3	Enregistrer les données dans le serveur																		
X X					T4.1	Tester et valider le stockage des données																		
						Revue projet 2																		
	X		X		T4.1	Assurer l'exploitation de la base de données																		
X X X X			X	X	T5.2	Interconnecter la station météo avec le serveur météo.																		
X X X X			X	X	T5.5	Intégration des sous systèmes																		
X X X X			X	X	T8.1	Gérer la planification																		
X X X X			X	X	T8.2	Assurer la traçabilité des travaux																		
X X X X			X	X	T9.6	Rédiger les documents relatifs au projet																		