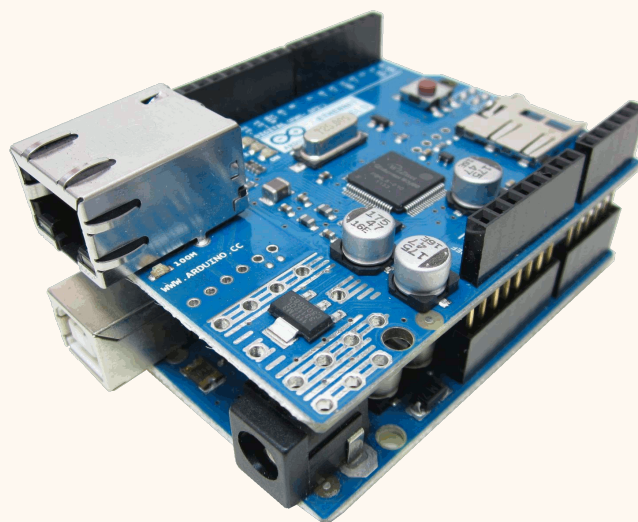


TRAVAUX PERSONNELS ENCADRÉS

du

LYCÉE FRANÇOIS RABELAIS DE CHINON
Baccalauréat scientifique, option sciences de l'ingénieur



Thème du TPE : « Avancée scientifique et réalisations techniques »

Comment relever des données météorologiques et les transmettre sur un espace accessible par tous ?



Soutenu le 25 mars 2015

Encadré par
M. GUIBERT : professeur de sciences de l'ingénieur
M. LABOURDETTE : professeur de sciences de l'ingénieur

Présenté par
Téofil ADAMSKI
Remi BRUYÈRE
Sammy PLAT
Clément BIDAULT

Résumé & remerciements

Depuis quelques années maintenant, nous assistons à un réel développement de la domotique. De nombreux produits sont disponibles pour l'utilisateur afin de rendre sa maison plus « intelligente ».

La *domotique* est l'ensemble des techniques utilisées pour nous rendre le quotidien meilleur. Ce domaine utilise quasiment toutes les techniques existantes : nous pouvons trouver aussi bien de l'électronique que de la physique ou encore de l'informatique. Ces dispositifs permettent, entre autre, de contrôler notre maison depuis l'extérieur : fermeture et ouverture des volets, déclenchement du chauffage, sécuriser notre habitat, ... Mais ils peuvent également augmenter notre confort et embellir notre quotidien comme les home-cinéma, la musique multi-pièce, ...

Aujourd'hui, un des rôles très important de la domotique est de réduire les coûts financiers et les impacts sur l'environnement. Ainsi, des objets technologiques comme des thermostats intelligents ou des capteurs de température se développent.

Par la biais de nos travaux personnels encadrés, nous avons décidé de nous diriger vers ce domaine. Tout au long de ce dossier, nous nous questionnerons sur

Comment relever des données météorologiques et les transmettre sur un espace accessible par tous ?

Ainsi, pour combler cette problématique, nous nous sommes penchés sur le développement et la fabrication d'un releveur de données météorologiques qui est capable de transmettre ses données sur Internet. Notre dispositif permettra alors de contrôler la température et la pression d'une pièce dans le but de réguler le chauffage. Celui-ci peut aussi faire office de mini-station météorologique qui délivrera, en temps réel sur le Web, la température et la pression.

Notre dossier s'organisera en plusieurs parties. Dans un premier temps, nous allons vous présenter notre projet avec son analyse fonctionnelle. Ensuite, nous expliquerons en détails notre démarche pour créer cet objet avec les parties sur l'électronique, la base de données, le site Web et la mise en place de notre serveur. Enfin, nous présenterons les résultats que nous aurons obtenus. À la fin de ce dossier, vous pourrez retrouver en annexe les codes complets utilisés.

Nous pouvons remercier nos deux professeurs de sciences de l'ingénieur M. GUIBERT et M. LABOURDETTE qui nous ont conseillés tout au long de nos recherches.

Table des matières

1	Présentation du projet	1
1.1	Idée de départ	1
1.2	Analyse fonctionnelle	1
1.2.1	Méthode APTE	1
1.2.2	Diagramme des interacteurs	2
1.2.3	Cahier des charges fonctionnel	2
1.2.4	Synoptique et chaîne d'information	4
2	Partie électronique	6
2.1	Une plateforme pour recevoir les données	6
2.1.1	Qu'est-qu'un Arduino ?	6
2.1.2	Programmation d'un Arduino	6
2.2	Le choix du capteur	7
2.2.1	Caractéristiques d'un capteur	7
2.2.2	Capteur utilisé	8
2.3	Connexion de l'Arduino au réseau Ethernet	9
2.4	Mise en place du programme final	11
3	Base de données	12
3.1	Qu'est-ce qu'une base de données ?	12
3.2	Notre utilisation	12
4	Partie site Web	16
4.1	Deux technologies : HTML/CSS et PHP	16
4.1.1	Partie statique : HTML et CSS	16
4.1.2	Partie dynamique : PHP	17
4.2	Ajout des données par l'Arduino	18
4.3	Récupération des données	18
4.4	Le graphique de données	19
5	Installation du serveur	21
5.1	Logiciels utilisés	21
5.2	Mise en place du réseau	22
6	Résultats finaux	24
A	Code complet pour l'Arduino	27

Table des figures

1.1	Le testo 175 T1	1
1.2	Diagramme bête à corne de notre projet	2
1.3	Diagramme des interacteurs	2
1.4	Synoptique récapitulant tous les composants	4
1.5	Chaîne d'information de la partie électronique	5
2.1	Un Arduino Uno R3	6
2.2	Le Base Shield V2 et le BMP180	8
2.3	L'Ethernet Shield V2.0	10
3.1	Structure de notre base	13
3.2	Données de la base sans et avec LIMIT	15
4.1	Exemple de CSS	17
4.2	Notre site sans et avec le CSS	17
4.3	Graphique avec Highchart	20
5.1	Schéma de notre réseau local	22
5.2	Configuration réseau du serveur	23

Liste des tableaux

1.1	Le cahier des charges fonctionnel	3
-----	---	---

Chapitre 1

Présentation du projet

1.1 Idée de départ

L'idée directrice de notre TPE est la suivante : la capture de donnée météorologique et le transfert de ces données sur un site Web accessible à tous. Nous voulions créer un système qui permet à l'utilisateur de connaître la température ainsi que la pression d'un environnement à distance. Ce type de produit existe déjà sur le marché de la domotique mais ils stockent les données sur une carte SD (comme ceux de la marque Testo, qui coûte plus d'une centaine d'euros).



FIGURE 1.1 – Le testo 175 T1 (prix : 150 € TTC)

Nous avons voulu innover en stockant les relevés sur une base de données en ligne. De plus, notre projet relève non seulement la température mais aussi la pression. Nous avons aussi essayé de réduire les coûts.

Nous avons donc pensé à utiliser un microcontrôleur relié à un capteur qui enverra les données vers une base de données. Ensuite, un site Web ira chercher les données sur cette base. Nous présenterons rapidement les éléments cités dans la synoptique et approfondirons les sujets dans les différentes parties du dossier.

1.2 Analyse fonctionnelle

Nous nous sommes donc penchés sur les caractéristiques de ce que nous devons mettre en œuvre pour arriver à mener à bien notre projet.

1.2.1 Méthode APTE

Dans un premier lieu, nous nous sommes demandés pourquoi, comment et pour qui cela pourrait servir. Nous avons donc utilisé la méthode dite « méthode APTE » qui a pour but de déterminer l'utilité de notre projet. Au début, nous avons créé le diagramme bête à corne suivant.

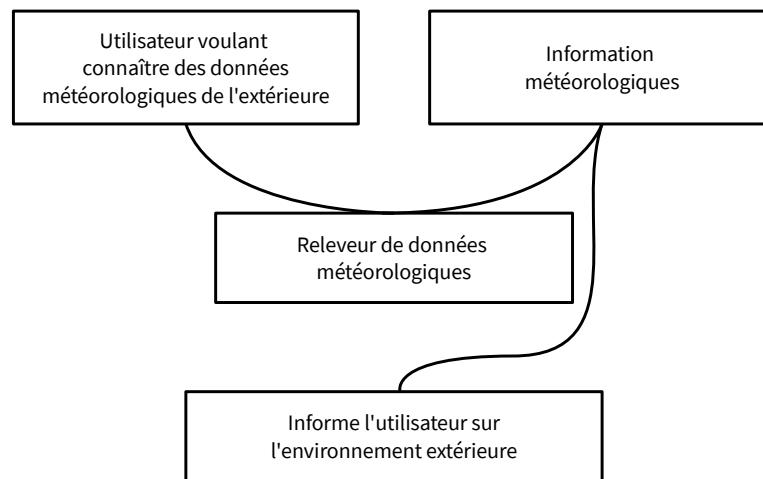


FIGURE 1.2 – Diagramme bête à corne de notre projet

Nous pouvons donc penser que les données météorologiques que nous recueillerons serviront à l'utilisateur. Celui-ci pourra alors être informé de la température et de la pression atmosphérique d'une pièce ou d'un extérieur en fonction de l'utilisation faite par celui qui s'en servira.

Ensuite, il a fallu nous poser quelques questions sur ce qui pourrait conduire la disparition de ce projet. Il pourrait, par exemple, naître un nouveau projet qui puissent capter des données plus précises ou plus fidèles, qui puisse stocker plus de données, ... Mais pour l'instant, nous n'avons pas vu d'autre produit comme le notre !

1.2.2 Diagramme des interacteurs

Suite à cela, nous devons déterminer toutes les fonctions que réalisera notre projet et ses interactions avec l'extérieur. Ainsi, nous avons réalisé le diagramme des interacteurs ou diagramme « pieuvre » suivant.

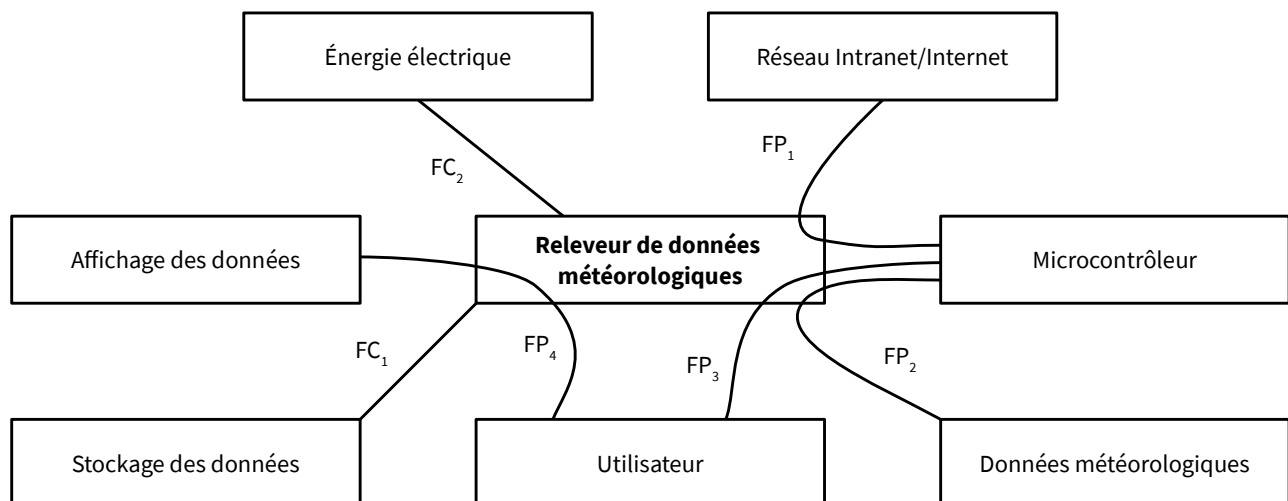


FIGURE 1.3 – Diagramme des interacteurs

Ainsi, sept interacteurs ont été trouvés et il en découle quatre fonctions principales et deux fonctions contraintes.

1.2.3 Cahier des charges fonctionnel

Enfin, pour terminer notre analyse fonctionnelle, nous avons regroupé toutes nos caractéristiques et nos fonctions pour former un cahier des charges fonctionnel (voir tableau 1.1).

Fonctions		Critères	Niveaux	Flexibilité
FP ₁	Doit pouvoir communiquer des informations sur le réseau Intranet/Internet	Prise RJ45 et accès au réseau de lycée (admission d'adresse IP) ou réseau local	aucun	aucun
FP ₂	Doit pouvoir capter des données météorologiques	Capteur de température Capteurs de pression	Précision à 1 °C Précision à 10 hPa	F ₂
FP ₃	Doit être facile d'installation et d'utilisation, être sécurisé pour l'utilisateur	Boîtier de protection Indicateur de mise en marche Raccord au réseau facile	Étanche Voyant (LED) Câble et prise RJ45 (connexion filaire)	F ₀ F ₁ F ₀
FP ₄	Données doivent être visible par l'utilisateur	Site Web avec langage de programmation	Espace inférieur à 10 Mo	F ₂
FC ₁	Doit pouvoir transférer les données sur un moyen de stockage	Base de données	Espace inférieur à 10 Mo	F ₂
FC ₂	Doit utiliser l'électricité pour s'alimenter en énergie	Électricité courante suivie transformateur courant continue	Tension de 5 V à 12 V	F ₀

TABLE 1.1 – Le cahier des charges fonctionnel

Nous avons donc déterminé, en fonction de nos attentes, les critères pour pouvoir réaliser le projet. Le niveau correspond à la valeur du critère, par exemple, le capteur de température doit être au moins précis à 1 °C. Ensuite, plus la flexibilité d'un niveau est faible, moins on peut le modifier. Nous pouvons donc voir, par exemple, que pour la température, on autorise une flexibilité de 2 ce qui veut dire que l'on autorise une précision un peu plus petite. Autre exemple, pour l'esthétique, la flexibilité est au maximum car cela n'empêchera pas les performances ou les fonctionnalités de l'objet.

1.2.4 Synoptique et chaîne d'information

Nous avons commencé à rentrer dans le détail de notre TPE en choisissant notre capteur, qui sera un capteur de température et de pression BMP180 et notre microcontrôleur sera un Arduino Uno pour pouvoir contrôler le capteur. En plus de ce microcontrôleur, nous mettrons un Arduino Ethernet V2.0 qui nous permettra d'envoyer les données sur la base de donnée qui se situera sur un serveur qui hébergera aussi le site Web. Un *switch* sera installé entre l'Arduino et l'ordinateur, nous pourrions également brancher un deuxième ordinateur en tant que client (l'utilisateur final). Pour notre base de données, nous allons utiliser MySQL et, pour notre serveur Web, nous allons utiliser Apache. Tous ces logiciels sont présentés dans la section 5.1.

Nous avons donc représenté les interactions entre les différents composants sur un synoptique (voir figure 1.4).

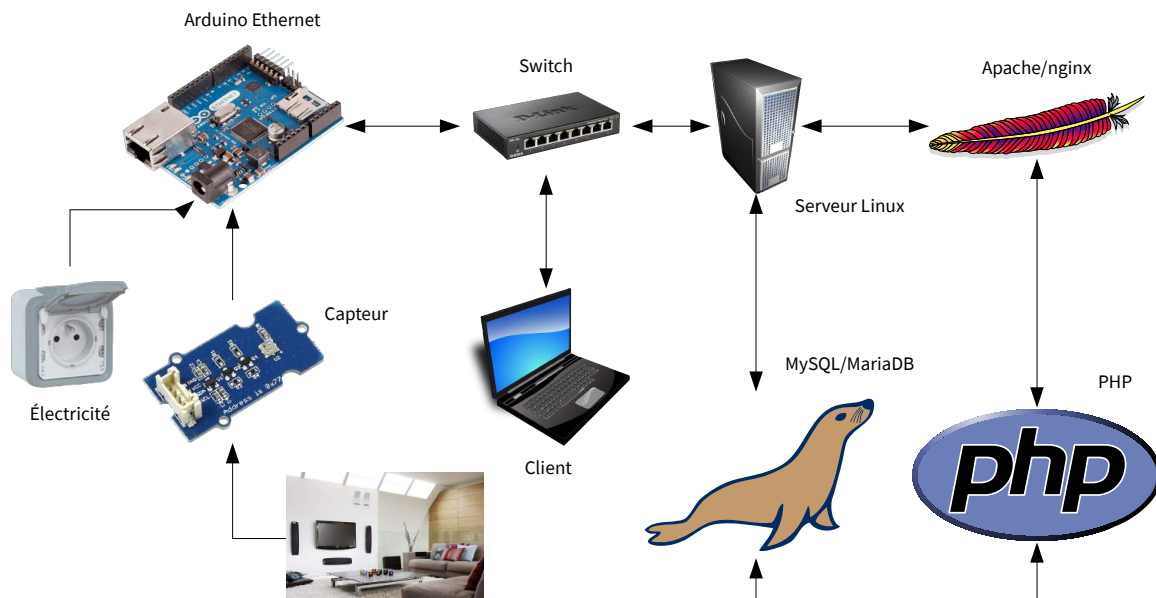


FIGURE 1.4 – Synoptique récapitulant tous les composants

Enfin, nous avons déterminé notre chaîne d'information (voir figure 1.5), de l'acquisition de la température à l'insertion de la base de données. La partie site Web ne figure pas car elle est indépendante de cette partie bien qu'il communique avec la base.

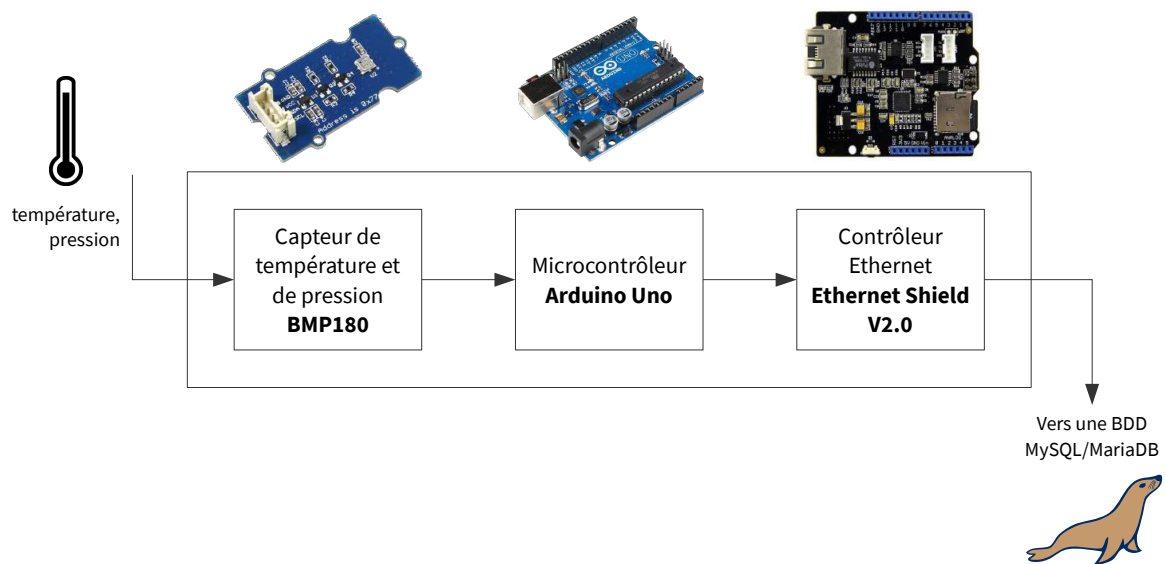


FIGURE 1.5 – Chaîne d’information de la partie électronique

Chapitre 2

Partie électronique

Dans cette partie, nous nous intéresserons à la partie électronique de notre projet. Elle s'accompagnera d'explications théoriques sur les différents composants que nous utilisons sur notre réalisation.

2.1 Une plateforme pour recevoir les données

Comme vu dans la partie précédente, notre projet consiste à capter, au départ, des données météorologiques. Pour faire cela, il nous faut une plateforme qui puisse communiquer avec des capteurs.

Naturellement, nous nous sommes tournés vers un *Arduino*.

2.1.1 Qu'est-ce qu'un Arduino ?

Un Arduino est une plateforme possédant des entrées et des sorties. Il est organisé autour d'un microcontrôleur Atmel. Celui-ci est capable de contrôler différents récepteurs comme une LED, un moteur ou encore un autre microcontrôleur. Il est capable également de recevoir et traiter des données d'un capteur. C'est dans ce cas que nous allons l'utiliser.

Les entrées et les sorties sont commandées à l'aide d'un code compilé sur un ordinateur puis téléversé sur le microcontrôleur.

Il existe une multitude de cartes Arduino. Pour nos besoins, nous allons utiliser un *Arduino Uno*. C'est une carte qui comporte 13 entrées/sorties numériques et 6 analogiques.

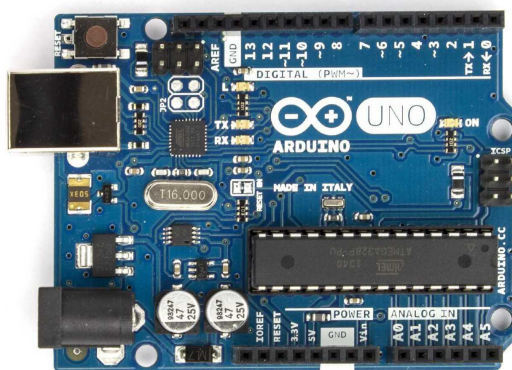


FIGURE 2.1 – Un Arduino Uno R3

2.1.2 Programmation d'un Arduino

Pour programmer un Arduino, il faut utiliser un ordinateur avec l'IDE¹ du constructeur installé. Après l'avoir installé et configuré pour qu'il programme la bonne carte, nous pouvons y insérer du code.

1. Disponible sur <http://arduino.cc/en/Main/Software>.

La langage pour programmer un Arduino est très proche du C++, il s'agit en fait d'une surcouche. De ce fait, ce langage intègre la programmation orientée objet qui est très utile quand on veut développer : cela permet d'écrire du code plus compréhensible. Nous avons appris à nous en servir grâce au tutoriel [1] sur Zeste de Savoir.

La structure d'un code Arduino se divise essentiellement en 2 parties : la première pour initialiser les différents composants, la seconde contient le code qui doit s'exécuter en boucle. Ainsi, pour faire clignoter une LED raccordée à la sortie 13, nous pouvons téléverser le code suivant à l'Arduino.

```
1  #define LED 13 // On stocke l'entrée dans une constante.
2
3  /* Phase d'initialisation */
4  void setup()
5  {
6      pinMode(LED, OUTPUT); // On dit que la LED est une sortie.
7  }
8
9  /* Boucle infinie */
10 void loop()
11 {
12     digitalWrite(LED, HIGH); // On éteint la LED.
13     delay(1000);             // On attend 1000 ms = 1 s.
14     digitalWrite(LED, LOW);  // On allume la LED.
15     delay(1000);             // On attend 1 s.
16 }
```

Comme vous pouvez le voir, le code est relativement simple. C'est pourquoi les plateformes Arduino sont conseillées pour les débutants (comme nous) en électronique. Dans ce code, il est possible d'ajouter des conditions avec `if .. elseif ... else ...` ou des boucles `for` ou `while` pour faire des exemples plus complets.

2.2 Le choix du capteur

Afin de relever nos données météorologiques, il nous faut un capteur.

2.2.1 Caractéristiques d'un capteur

Un capteur est un composant permettant de traduire une grandeur physique en un courant électrique, hydraulique, ... Les capteurs sont utilisés quasiment partout et il en existe beaucoup : certains peuvent capter le courant, le vent, la température ou encore la lumière. Il s'agit en fait d'une interface entre le monde extérieur et un circuit électronique.

Type de sortie Tout d'abord, il existe principalement deux types des capteurs qui se classent en fonction du leur sortie. Les capteurs analogiques sortent une tension ou une intensité qui se traduit par la valeur que capte ce composant.

Nous pouvons trouver également des capteurs numériques qui ne sortent pas un courant mais des états logiques : 0 ou 1. Nous allons nous tourner vers ce type de capteur

Caractéristiques Pour distinguer différents capteurs, il existe des caractéristiques :

- l'étendue : c'est l'écart entre la plus grande et la plus petite valeur mesurable,
- la résolution : c'est la plus petite variation que le capteur est capable de mesurer. Par exemple, un capteur de température capable de distinguer au maximum 20,0 °C de 20,1 °C à une résolution de 0,1 °C ;
- la sensibilité : c'est le rapport entre la variation du signal d'entrée et la variation du signal de sortie ;
- l'exactitude : elle indique le pourcentage d'erreurs commises ;
- la justesse : c'est l'aptitude à donner une valeur juste, c'est l'écart entre le résultat moyen et la valeur vraie ;

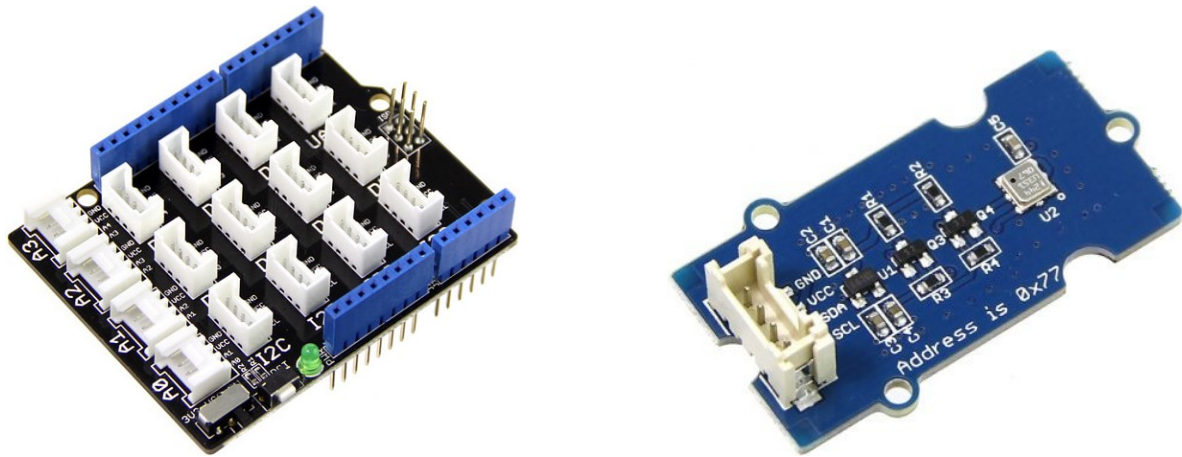


FIGURE 2.2 – Le Base Shield V2 et le BMP180

- la fidélité : elle définit la dispersion des valeurs relevées.
- En fonction de ces différentes caractéristiques, nous pouvons choisir un capteur.

2.2.2 Capteur utilisé

Comme vu dans le premier chapitre, notre capteur se doit de capter la température ainsi que la pression. Au lycée, nos deux professeurs nous ont orienté vers le capteur BMP180 en particulier celui de la marque Seeed Studio. Celui-ci a une particularité, il a besoin d'un *shield* pour pouvoir être branché à un Arduino.

Ce capteur se relie alors à une borne I2C et le *shield* se place sur l'Arduino. L'avantage de celui-ci est de permettre un câblage facile sans rajout de composants électroniques.

D'après le wiki de Seeed Studio [3], l'étendue de mesure de ce capteur va de -40°C à 85°C et il est précis à 2°C près : cela ne satisfait pas totalement notre cahier des charges initial mais suffira pour notre utilisation.

Protocole I²C

Le capteur BMP180 se pilote à l'aide du protocole I²C. Ce protocole est un bus série créé par Philips. Les échanges d'informations se font toujours entre un seul maître et un seul esclave (ici, l'Arduino et le capteur respectivement). La connexion des composants I²C se fait par l'intermédiaire de trois fils :

- la SDA (*Serial Data Line*) : c'est la ligne de données ;
- la SCL (*Serial Clock Line*) : c'est la ligne d'horloge ;
- la masse.

Pour un Arduino Uno, le fils pour la SDA doit se relier au *pin* A4 et la SCL au *pin* A5 mais cela nous est inutile grâce au *shield*.

Sa programmation

Pour communiquer avec le capteur, il va falloir programmer l'Arduino. Pour cela, le constructeur du capteur nous met à disposition une bibliothèque² qui va nous faciliter le travail. Il suffit alors de décompresser l'archive dans le dossier des bibliothèques Arduino.

Dans notre code, il faut inclure le fichier `Barometer.h` qui correspond à l'en-tête de la bibliothèque. Ensuite, il faut déclarer le capteur puis relever ses données. Voici un exemple de code qui permet de capter la température et la pression.

2. Disponible sur leur dépôt GitHub : https://github.com/Seeed-Studio/Grove_Barometer_Sensor.

```

1  #include <Barometer.h> // On inclut la bibliothèque du constructeur.
2  #include <Wire.h>
3
4  float temperature(0), pression(0); // On déclare les variables.
5
6  Barometer capteur; // On déclare le capteur.
7
8  void setup()
9  {
10     Serial.begin(9600); // On initialise la communication série
11     capteur.init();      // et le capteur.
12 }
13
14 void loop()
15 {
16     /* On stocke la température et la pression. */
17     temperature = capteur.bmp085GetTemperature(capteur.bmp085ReadUT());
18     pression = capteur.bmp085GetPressure(capteur.bmp085ReadUP());
19
20     /* On affiche les données. */
21     Serial.print("Température : ");
22     Serial.print(temperature);
23     Serial.println(" °C");
24     Serial.print("Pression : ");
25     Serial.print(pression);
26     Serial.println(" Pa");
27     Serial.println();
28
29     delay(1000);
30 }

```

Comme nous allons envoyer les données sur le serveur tous les quarts d'heures, nous préférons prendre une température et une pression toutes les 90 s et faire la moyenne de toutes les valeurs reçues au bout de 15 min. Cela évitera les valeurs extrêmes et augmentera la fidélité des relevés. Pour se faire, nous utilisons un boucle `for` allant de 0 à 9 (ce qui fait 10 relevés car $10 \times 90 \text{ s} = 15 \text{ min}$). À l'intérieur, nous faisons la somme des relevés. Puis nous divisons par 10 cette somme. Voici un exemple uniquement pour la température.

```

1  // ...
2  float moyenneTemperature(0);
3
4  for (int i(0) ; i < 10 ; i++)
5  {
6     moyenneTemperature += capteur.bmp085GetTemperature(capteur.bmp085ReadUT());
7     delay(90000);
8  }
9
10 moyenneTemperature /= 10;
11 // ...

```

2.3 Connexion de l'Arduino au réseau Ethernet

Un des côtés essentiels de notre projet est de pouvoir communiquer les données sur Internet. Pour faire communiquer l'Arduino à Internet, il est possible de lui ajouter un *shield* comportant une carte réseau. Pour notre projet, nous utilisons le shield Ethernet V2.0 de Seeed Studio.

Ce *shield* est composé d'une puce W5200 qui permet de lancer des requêtes TCP/UDP sur un réseau Ethernet. Il faut bien évidemment le relier à un *switch* ou à une *box* par un câble Ethernet.

Pour programmer ce *shield*, il faut aussi utiliser la bibliothèque³ proposée par Seeed Studio. Le code suivant permet de charger une page (ici, google.fr) et de nous retourner cette page.

3. Disponible sur leur dépôt GitHub : https://github.com/Seeed-Studio/Ethernet_Shield_W5200.

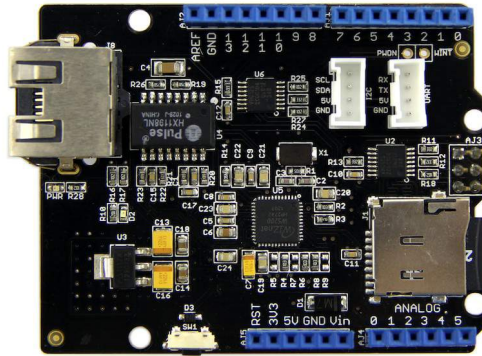


FIGURE 2.3 – L’Ethernet Shield V2.0

```

1  #include <SPI.h>
2  #include <EthernetV2_0.h> // On inclut la bibliothèque.
3
4  #define W5200_CS 10
5  #define SDCARD_CS 4
6
7  byte MAC[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // On donne une adresse MAC de l'Arduino.
8
9  char serveur[] = "google.fr"; // Serveur vers lequel on va faire la requête.
10
11 EthernetClient client; // On crée un client Ethernet.
12
13 void setup()
14 {
15     Serial.begin(9600);
16
17     /* Désactivation de la carte SD */
18     pinMode(SDCARD_CS, OUTPUT);
19     digitalWrite(SDCARD_CS, HIGH);
20
21     /* On commence la connexion Ethernet. */
22     if (Ethernet.begin(MAC) == 0)
23     {
24         Serial.println("Impossible de configurer la connexion Ethernet.");
25
26         while (true); // On arrête alors le programme.
27     }
28
29     delay(1000);
30
31     Serial.println("Connexion.");
32
33     /* On se connecte au port 80 du serveur. */
34     if (client.connect(serveur, 80))
35     {
36         Serial.println("Connecté.");
37
38         client.println("GET / HTTP/1.0"); // On lance une requête HTTP.
39         client.println();
40     }
41     else
42     {
43         Serial.println("Erreur de connexion.");
44     }
45 }
46
47 void loop()
48 {

```

```
49  /* On affiche le résultat de la requête s'il existe. */
50  if (client.available())
51  {
52      char c = client.read();
53      Serial.print(c);
54  }
55
56  /* Quand le client n'est pas connecté, on arrête le client. */
57  if (!client.connected())
58  {
59      Serial.println("Déconnexion.");
60      client.stop();
61
62      while (true);
63  }
64 }
```

Pour initialiser l'Ethernet V2.0, il faut lui assigner une adresse MAC unique comme on le fait à la ligne 7. Une adresse MAC est un identifiant unique stocké dans la carte réseau.

Revenons sur le code allant de la ligne 34 à 44. Tout d'abord, il faut savoir que chaque serveur ont des ports d'ouvert, le port pour le serveur Web, le logiciel qui délivre le site, est le 80. Ainsi, à la ligne 34, nous essayons de nous connecter à `serveur` en écoutant le port 80. Ensuite, nous lançons une requête HTTP pour avoir le contenu de la racine du site.

2.4 Mise en place du programme final

Ainsi, pour notre projet, nous devons associer les deux codes : celui pour la capteur et celui pour l'Ethernet. Pour pouvoir transférer les données à la base de données, nous allons appeler un script PHP situé sur le serveur que nous détaillerons dans la section 4.2. Ce script se présente sous la forme suivante.

```
1 http://192.168.1.2/arduino.php?mdp=123&temperature=21&pression=10079
```

Ici, par cet appel, une entrée avec le temps courant, la température de 21 °C et la pression de 10 079 hPa seront ajoutés dans la base. Le premier argument sert pour sécuriser son accès.

Le code complet se trouve en annexe à la page 27. En téléversant ce code sur l'Arduino, il va capter une température et une pression toute les 90 s. Au bout de 15 min, il fait la moyenne de 10 valeurs. Enfin, il fait une requête et charge la page qui ajoute les données dans la base.

Cependant, il subsiste un problème : l'Arduino charge la page demandée uniquement lorsqu'il est connecté à un ordinateur par USB. Nous n'avons pas réussi à régler ce problème...

Chapitre 3

Base de données

3.1 Qu'est-ce qu'une base de données ?

Une *base de données* est un moyen de stocker, d'organiser et de hiérarchiser des données. Une base de données peut être rapprochée d'un classeur Calc ou Microsoft Excel :

- chaque fichier est une base de données ;
- chaque onglet (feuille) correspond à une table ;
- chaque colonne (ou ligne) est un champ spécifiant une caractéristique des données ;
- chaque ligne (ou colonne) est un enregistrement, une donnée.

Il existe plusieurs moyens de stocker des données.

- Le format texte (`.txt`) est un choix assez évident pour les néophytes, il ne nécessite pas de logiciels supplémentaires. Mais, il est peu efficace, un fichier devient vite encombrant, il utilise beaucoup de ressources à la lecture et à l'écriture, et chaque table nécessite un fichier séparé qui est ouvert tour à tour.
- Le format texte optimisé pour les données (`.csv`) ne nécessite pas de logiciels supplémentaires. C'est un moyen efficace de stocker les données quand il est bien utilisé. Mais, il a les mêmes inconvénients que le format texte.
- Le classeur Excel ou Calc (`.xls`, `.xlsx` ou `.ods`) est équivalent à une base de données. Mais, le fichier devient encombrant et est peu pratique à utiliser, il utilise beaucoup de ressources pour l'utilisation, il n'est pas optimisé pour être utilisé sur un serveur et il doit être créé par un logiciel externe.
- Le format Microsoft DataBase (`.mdb`) est un format spécial base de données, il est entièrement fonctionnel sous Windows. Mais, il est impossible à utiliser avec des outils gratuits et doit être créé par un logiciel externe.
- Le format OpenDocument Database (`.odb`) est un format de bases de données libre. Mais, tout comme le précédent, il doit être créé par un logiciel extérieur.
- Enfin, le format SQL (*Structured Query Language* en anglais, `.sql`) est gratuit et libre. Il est issu d'un consensus entre les différentes technologies pour unifier leur utilisation. Sa connexion est simple (selon les systèmes) et les requêtes basiques sont simples mais peuvent être complexes suivant l'utilisation. Mais il nécessite un Système de Gestion de Base de Données (ou SGBD) comme PostgreSQL, MySQL ou ORACLE qu'il faut installer.

3.2 Notre utilisation

Afin de stocker les différentes données acquises par la partie électronique, nous devons utiliser une base de données avec un champ pour le temps, la température et la pression.

Pour notre site, nous utilisons MySQL, un Système de Gestion de Base de Données Relationnel (SGBDR) qui peut créer et gérer des relations entre différentes tables, couplé avec le module PDO de PHP pour se connecter à la base.

Pour pouvoir utiliser une base de données, il faut d'abord installer le serveur MySQL, dont l'installation sera détaillée dans la section 5.1.

Après l'installation, nous devons créer la base (que l'on nommera **tpe**). Pour ce faire, dans la console MySQL ou *via* phpMyAdmin (administration pour MySQL en ligne), on rentrera le requête suivante :

```
1 CREATE DATABASE tpe
```

La base est alors créée mais encore inutilisable car ne contenant pas de tables. Nous créons donc une table nommée **datalog_meteo** contenant plusieurs champs avec la requête :

```
1 CREATE TABLE datalog_meteo (
2   id int(10) unsigned NOT NULL AUTO_INCREMENT,
3   temps timestamp NULL DEFAULT CURRENT_TIMESTAMP,
4   temperature float DEFAULT NULL,
5   pression float DEFAULT NULL,
6   PRIMARY KEY (id)
7 ) ENGINE=InnoDB;
```

La requête crée une table avec les champs **id** (une valeur qui s'auto-incrémente pour différencier les enregistrements), **temps** (une valeur de temps), **temperature** (un nombre décimal) et **pression** (un nombre décimal également).

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> datalog_meteo	Afficher Structure Rechercher Insérer Vider Supprimer	~28	InnoDB	utf8_general_ci	16 Kio	-
1 table	Somme	28	InnoDB	utf8_general_ci	16 Kio	0 0

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	id	int(10)		UNSIGNED	Non	Aucune	AUTO_INCREMENT
2	temps	timestamp			Oui	CURRENT_TIMESTAMP	
3	temperature	float			Oui	NULL	
4	pression	float			Oui	NULL	

FIGURE 3.1 – Structure de notre base

La table est bien créée et utilisable mais est vide. Pour la remplir, nous utilisons la requête :

```
1 --- Valeurs pour tester la table ---
2 INSERT INTO `datalog_meteo` (`temps`, `temperature`, `pression`) VALUES
3   ('2014-12-10 21:00:43', 105, 100);
4
5 --- Valeurs réelles ---
6 INSERT INTO `datalog_meteo` (`temps`, `temperature`, `pression`) VALUES
7   ('2015-02-15 18:11:09', 21.09, 1010.46),
8   ('2015-02-15 18:12:38', 21.06, 1010.52),
9   ('2015-02-15 18:12:40', 21.07, 1010.51),
10  ('2015-02-15 18:12:43', 21.09, 1010.58),
11  ('2015-02-15 18:12:46', 21.1, 1010.52),
12  ('2015-02-15 18:12:48', 21.1, 1010.52),
13  ('2015-02-15 18:12:51', 21.1, 1010.54),
14  ('2015-02-15 18:12:54', 21.1, 1010.54),
15  ('2015-02-15 18:12:56', 21.09, 1010.49),
16  ('2015-02-15 18:12:59', 21.1, 1010.54),
17  ('2015-02-15 18:46:37', 21.1, 1011.14),
18  ('2015-02-15 18:46:40', 21.1, 1011.1),
19  ('2015-02-15 18:46:44', 21.1, 1011.07),
20  ('2015-02-15 18:46:48', 21.1, 1011.11),
21  ('2015-02-15 18:46:54', 21.1, 1011.19),
22  ('2015-02-15 18:46:56', 21.1, 1011.1),
23  ('2015-02-15 18:46:59', 21.1, 1011.15),
24  ('2015-02-15 18:47:02', 21.1, 1011.16),
25  ('2015-02-15 18:47:05', 21.1, 1011.1),
```

```
26 ('2015-02-15 18:47:07', 21.1, 1011.12),
27 ('2015-02-15 18:47:10', 21.1, 1011.1),
28 ('2015-02-15 18:47:13', 21.1, 1011.1),
29 ('2015-02-15 18:47:15', 21.1, 1011.13),
30 ('2015-02-15 18:47:18', 21.1, 1011.09),
31 ('2015-02-15 18:47:21', 21.1, 1011.11),
32 ('2015-02-15 18:47:23', 21.1, 1011.06),
33 ('2015-02-15 18:47:26', 21.1, 1011.15);
```

La requête `INSERT INTO (table) (champs)` indique à MySQL qu'il doit insérer dans les champs indiqués de la table les valeurs suivant le mot-clé `VALUES`.

Nous pouvons maintenant afficher les valeurs contenues dans la table grâce à la requête :

```
1 SELECT * FROM datalog_meteo;
```

Cette requête affiche tout le contenu de la table `datalog_meteo` sans exception. Nous pouvons cependant restreindre ceci en donnant les noms des champs nous étant utile, et, par exemple, un intervalle de temps ou une limite (une limite de x valeurs retournera au plus x valeurs).

Nous utiliserons par défaut les 20 dernières valeurs, la requête la plus utilisée est alors

```
1 SELECT temps, temperature, pression FROM datalog_meteo ORDER BY temps DESC LIMIT 20;
```

Nous demandons au serveur SQL de renvoyer les valeurs des champs `temps`, `temperature` et `pression` dans l'ordre de temps décroissant et dans la limite de 20 valeurs. Le serveur renvoie alors un ensemble de données ressemblant à la figure suivante (dépendant des données).

id	temps	temperature	pression
1	2014-12-10 22:00:43	105	100
275	2015-02-15 19:11:09	21.09	1010.46
276	2015-02-15 19:12:38	21.06	1010.52
277	2015-02-15 19:12:40	21.07	1010.51
278	2015-02-15 19:12:43	21.09	1010.58
279	2015-02-15 19:12:46	21.1	1010.52
280	2015-02-15 19:12:48	21.1	1010.52
281	2015-02-15 19:12:51	21.1	1010.54
282	2015-02-15 19:12:54	21.1	1010.54
283	2015-02-15 19:12:56	21.09	1010.49
284	2015-02-15 19:12:59	21.1	1010.54
285	2015-02-15 19:46:37	21.1	1011.14
286	2015-02-15 19:46:40	21.1	1011.1
287	2015-02-15 19:46:44	21.1	1011.07
288	2015-02-15 19:46:48	21.1	1011.11
289	2015-02-15 19:46:54	21.1	1011.19
290	2015-02-15 19:46:56	21.1	1011.1
291	2015-02-15 19:46:59	21.1	1011.15
292	2015-02-15 19:47:02	21.1	1011.16
293	2015-02-15 19:47:05	21.1	1011.1
294	2015-02-15 19:47:07	21.1	1011.12
295	2015-02-15 19:47:10	21.1	1011.1
296	2015-02-15 19:47:13	21.1	1011.1
297	2015-02-15 19:47:15	21.1	1011.13
298	2015-02-15 19:47:18	21.1	1011.09
299	2015-02-15 19:47:21	21.1	1011.11
300	2015-02-15 19:47:23	21.1	1011.06
301	2015-02-15 19:47:26	21.1	1011.15

id	temps ▼	temperature	pression
301	2015-02-15 19:47:26	21.1	1011.15
300	2015-02-15 19:47:23	21.1	1011.06
299	2015-02-15 19:47:21	21.1	1011.11
298	2015-02-15 19:47:18	21.1	1011.09
297	2015-02-15 19:47:15	21.1	1011.13
296	2015-02-15 19:47:13	21.1	1011.1
295	2015-02-15 19:47:10	21.1	1011.1
294	2015-02-15 19:47:07	21.1	1011.12
293	2015-02-15 19:47:05	21.1	1011.1
292	2015-02-15 19:47:02	21.1	1011.16
291	2015-02-15 19:46:59	21.1	1011.15
290	2015-02-15 19:46:56	21.1	1011.1
289	2015-02-15 19:46:54	21.1	1011.19
288	2015-02-15 19:46:48	21.1	1011.11
287	2015-02-15 19:46:44	21.1	1011.07
286	2015-02-15 19:46:40	21.1	1011.1
285	2015-02-15 19:46:37	21.1	1011.14
284	2015-02-15 19:12:59	21.1	1010.54
283	2015-02-15 19:12:56	21.09	1010.49
282	2015-02-15 19:12:54	21.1	1010.54

FIGURE 3.2 – Données de la base sans et avec LIMIT

Chapitre 4

Partie site Web

Afin de réaliser notre projet, nous avons dû créer un site Web pour accueillir toutes les données que l'Arduino aura obtenu grâce au capteur.

4.1 Deux technologies : HTML/CSS et PHP

Pour ce faire, il nous a fallu coder ce site avec le langage HTML pour la partie fixe du site et le langage PHP pour la partie du qui viendra à changer au cours du temps.

4.1.1 Partie statique : HTML et CSS

Pour créer un site Web, il faut le coder à la main¹. Pour cela, il existe deux langages que sont le HTML et le CSS. À l'heure actuelle, le HTML est en version 5 et le CSS en version 3. Le HTML et le CSS sont complémentaires. En effet, le premier nous sert pour le fond, c'est-à-dire le contenu ; le second pour la forme, l'apparence.

Le HTML est un langage balistique : il utilise des balises de la forme `<balise>` et `</balise>`. Il est donc très compréhensible. Voici un exemple de page HTML :

```
1 <!DOCTYPE html><!-- Spécification de la version du HTML -->
2 <html>
3   <head><!-- En-tête : toutes les informations non-visible par l'utilisateur -->
4     <meta charset="utf-8">
5     <title>Exemple de page HTML</title><!-- Titre de la page -->
6   </head>
7   <body><!-- Corps de la page -->
8     <h1>Un titre</h1>
9
10    <p>Un paragraphe avec un <a href="http://localhost">lien</a>.</p>
11  </body>
12 </html>
```

D'un autre côté, le CSS s'organise en propriétés, en classes et en identificateurs. Il est capable de mettre en forme le HTML. Pour associer une classe ou un identificateur à une balise, il suffit d'utiliser respectivement les arguments `class="ma-classe"` et `id="mon-identificateur"` sur une balise quelconque. Contrairement à une classe, un identificateur ne peut s'utiliser qu'une fois dans une page. Dans le fichier CSS, les classes sont précédées d'un point, les identificateurs d'un croisillon # et leurs propriété sont entre accolades.

Supposons un fichier CSS nommé `style.css` avec le contenu suivant.

```
1 .important { /* Définition d'une classe */
2   color: red;
3   font-weight: bold;
```

1. Ou en passant par un logiciel spécifique.

```

4 }
5
6 h1 { /* Redéfinition d'un style de balise ; ici, le titre h1 */
7     color: green;
8     text-decoration: underline;
9 }

```

À partir de cela, nous pouvons créer un fichier HTML utilisant ces classes.

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="utf-8">
5         <title>Un exemple avec de CSS</title>
6         <link href="style.css" rel="stylesheet" type="text/css"><!-- Inclusion du fichier CSS -->
7     </head>
8     <body>
9         <h1>Un titre</h1>
10
11         <p>Un mot <span class="important">très important</span> !</p>
12     </body>
13 </html>

```

Ainsi, nous pouvons voir le résultat sur la figure suivante.

Un titre

Un mot **très important** !

FIGURE 4.1 – Exemple de CSS

Notre site est alors constitué de ces deux langages. Vous pouvez en voir un aperçu sur la figure suivante ou en ligne sur <http://tpe.teguad.ovh>.



FIGURE 4.2 – Notre site sans et avec le CSS

4.1.2 Partie dynamique : PHP

Mais notre site a besoin d'être dynamique, de changer au cours du temps. Pour cela, nous utilisons le langage PHP. Ce langage s'exécute sur le serveur qui nous avons mis en place (voir chapitre 5) à l'inverse du HTML et du CSS qui s'exécute du côté client.

Un code PHP se trouve entre `<?php` et `?>`. À l'intérieur, nous pouvons mettre des variables, des structures conditionnelles, des boucles, etc. Le code suivant affiche un simple texte.

```

1 <?php echo "Un texte"; ?>

```

Le PHP sert à générer du code HTML statique. Ce langage va nous servir à ajouter et à récupérer les différentes données dans la base MySQL.

4.2 Ajout des données par l'Arduino

Afin que l'Arduino puissent transmettre ses données à la base, il va falloir créer un script qui va être chargé par celui-ci (voir section 2.4). Ce script va recevoir les arguments passés dans l'URL. Il y aura trois arguments :

- `temperature` contiendra la température ;
- `pression` contiendra la pression ;
- et `mdp` contiendra le mot de passe pour sécuriser l'accès à la base (ici, 123).

Ainsi, nous créons un fichier `arduino.php` à la racine du site. Le fichier `sql.php` qui contient tous le code pour accéder à la base MySQL est disponible en annexe.

```

1  <?php
2
3  require 'sql.php'; // Inclusion du fichier contenant les commandes pour accéder à la base
4
5  $data = array(); // Création d'un tableau contenant les données
6
7  if (isset($_GET['pass']) && $_GET['pass'] == '123') // On test si le mot de passe est bon.
8  {
9      $data['temps'] = 'NOW()'; // Le temps est maintenant.
10     /* Si il y a une température donnée, alors on la convertit. Sinon, rien. */
11     $data['temperature'] = (isset($_GET['temperature'])) ?
12         htmlspecialchars($_GET['temperature']) : NULL;
13     /* Idem pour la pression */
14     $data['pression'] = (isset($_GET['pression'])) ?
15         htmlspecialchars($_GET['pression']) : NULL;
16
17     insert($data); // Et on insert les données dans la base.
18 }

```

4.3 Récupération des données

L'objectif principal du site est de proposer un tableau de données. À l'aide de PHP, il faut coder une fonction qui crée le tableau recueillant les données venant de la base avec un nombre limité de valeurs pour éviter un tableau trop long qui serait illisible.

Il a donc fallu d'abord créer les en-tête des colonnes du tableau puis demander à la base d'aller chercher les valeurs. Nous avons donc créé une fonction que nous appellerons sur la page adéquate. Encore une fois, le fichier `connect.php` qui permet la connexion à la base se trouve en annexe.

```

1  <?php
2
3  require 'connect.php'; // Inclusion du fichier pour la connexion à la base
4
5  /* Fonction qui crée et affiche le tableau
6   - $limit le nombre de données que le tableau doit afficher */
7  function afficher_tableau($limit)
8  {
9      $db = $GLOBALS['db'];
10     ?>
11     <table>
12         <!-- En-tête du tableau -->
13         <thead>
14             <tr>
15                 <td>Date et Heure</td>
16                 <td>Température (&deg;C)</td>
17                 <td>Pression (hPa)</td>
18             </tr>
19         </thead>
20         <tbody>
21     <?php

```

```

22 $query = $db->query('SELECT temps, temperature, pression FROM `datalog_meteo` ORDER BY id DESC
    ↳ LIMIT '.$limit); // Requête SQL avec LIMIT
23 $meteo = $query->fetchAll(PDO::FETCH_ASSOC);
24
25 foreach($meteo as $data) // Parcours des données
26 {
27     $temps = preg_replace('#(\d{4})-(\d{2})-(\d{2}) (\d{2}:\d{2}:\d{2})#', 'Le $3/$2/$1 à $4',
    ↳ $data['temps']); // Conversion du temps
28
29     /* Affichage des données : création d'une ligne */
30     echo '<tr>';
31     echo '<td>'.$temps.'</td>';
32     echo '<td>'.$data['temperature'].'</td>';
33     echo '<td>'.$data['pression'].'</td>';
34     echo '</tr>';
35 }
36 ?>
37 </tbody>
38 </table>
39 <?php
40 }

```

Ainsi, les valeurs s'ajoutent petit à petit dans les cases avec l'heure à laquelle ont été prises les données, la température à cette heure précise et la pression à ce même moment. Par exemple, en créant une page avec la fonction `afficher_tableau(20)`, nous pourrions voir un tableau des 20 derniers relevés.

4.4 Le graphique de données

Enfin, pour que les données soient plus visibles, nous avons décidé de créer un graphique. Sur Internet, nous avons repéré Highchart : c'est une bibliothèque JavaScript faisant des graphiques vectoriels en HTML.

À partir des démonstrations proposés sur le site de Highchart², nous pouvons créer un graphique pour les températures. Afin de faire apparaître ces courbes, il faut d'abord dessiner les axes (heures en abscisse, température en ordonnée), les graduer et d'afficher la courbe.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Graphique avec Highcharts</title>
6     <!-- Inclusion des bibliothèques -->
7     <script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
8     <script src="http://code.highcharts.com/highcharts.js"></script>
9     <script src="http://code.highcharts.com/modules/exporting.js"></script>
10  </head>
11  <body>
12    <div id="graphique"></div><!-- Bloc pour le graphique -->
13
14    <script>
15      $(function() {
16        $('#graphique').highcharts({
17          chart: {
18            type: 'line'
19          },
20          title: {
21            text: 'Graphique des températures'
22          },
23          xAxis: { // Abscisses

```

2. Voir <http://www.highcharts.com/demo>.

```

24     categories: ['14 h', '14 h 15', '14 h 30', '14 h 45', '15 h', '15 h 15',
25               ↪ '15 h 30', '15 h 35', '16 h', '16 h 15', '16 h 30', '16 h 45']
26   },
27   yAxis: { // Ordonnées
28     title: {
29       text: 'Température (°C)'
30     }
31   },
32   plotOptions: {
33     line: {
34       dataLabels: {
35         enabled: true
36       },
37     },
38   },
39   series: [{ // Valeurs
40     name: 'Lycée',
41     data: [21, 23, 22, 21.5, 21.6, 22, 23.3, 22.1, 21.5, 22.6]
42   }]
43 });
44 </script>
45 </body>
46 </html>

```

Nous pouvons ainsi observer le résultat suivant.

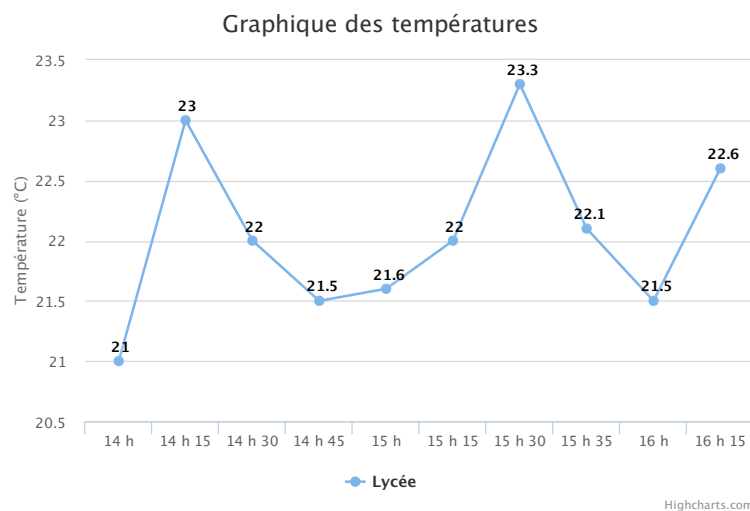


FIGURE 4.3 – Graphique avec Highchart

Il faut donc que nous codions une fonction PHP qui récupère les données et qui les adapte au code pour Highchart. Ce graphique mettra en image l'évolution de la température et de la pression en fonction du temps. Ce code, un peu long, est lui aussi disponible en annexe.

Chapitre 5

Installation du serveur

Pour finaliser notre installation, nous avons mis en place un serveur. Un serveur est une machine informatique permettant de réceptionner des informations et de les envoyer par la même occasion à un client.

5.1 Logiciels utilisés

Pour créer notre serveur, nous utilisons un ordinateur où nous avons installé Linux, Apache, MySQL et PHP. Le système d'exploitation Linux permet la mise en place d'un serveur très facilement, nous avons pris la distribution Linux Mint. Apache est un logiciel appelé « serveur HTTP », il permet, lorsqu'il est installé sur un ordinateur, de faire un serveur Web. Ainsi, si « serveur HTTP » désigne toujours un logiciel, « serveur Web » peut aussi bien désigner le logiciel, en l'occurrence Apache. Le SGBD¹ MySQL sert à gérer les bases de données. Enfin, PHP nous sert pour le côté dynamique du site Web.

Dans notre TPE, l'utilisation d'un serveur HTTP nous permet de faire fonctionner la base de données vu précédemment avec le site Web pour en ressortir les données.

Avec Linux, l'installation des logiciels est très facile. Nous utilisons le logiciel XAMPP² : il s'agit d'un ensemble de logiciels avec Apache, MySQL, PHP et Perl (nous n'utiliserons pas ce dernier). Après l'avoir téléchargé, il suffit d'exécuter le `.run` pour l'installer. Ensuite, on peut utiliser les commandes suivantes dans un terminal pour démarrer et arrêter le serveur Apache et MySQL.

```
1 sudo /opt/lampp/lampp start # Pour démarrer
2 sudo /opt/lampp/lampp stop  # Pour arrêter
```

Ensuite, par un navigateur Web, nous pouvons y accéder avec l'adresse `http://localhost`. Nous pouvons voir la page d'accueil. Il nous faut maintenant ajouter les fichiers HTML/CSS et PHP du site. Pour cela, il faut créer un nouveau dossier dans `/opt/lampp/htdocs` en ayant les permissions du superutilisateur et transférer tous nos fichiers à l'intérieur.

```
1 cd /opt/lampp/htdocs # On se déplace dans le dossier du serveur Web.
2 sudo mkdir tpe        # On crée un dossier tpe.
3 cd tpe
4 sudo cp -r nos-fichier/ /opt/lampp/htdocs/tpe # On copie nos-fichier dans tpe.
```

Nous pouvons également le faire en utilisant l'interface graphique. Ensuite, les fichiers sont accessibles par le navigateur à l'adresse `http://localhost/tpe`.

1. Système de gestion de bases de données.

2. Téléchargeable sur leur site : <https://www.apachefriends.org/download.html>.

5.2 Mise en place du réseau

Pour relier ce serveur Web avec notre Arduino, nous devons utiliser un *switch*. Un *switch* désigne un commutateur réseau, il s'agit d'un équipement ou un appareil qui permet l'interconnexion de différents appareils communicants entre eux. Il nous permet ici, grâce à l'Arduino Ethernet Shield, d'interconnecter nos différentes parties avec des câbles Ethernet. L'Arduino est donc relié au *switch* tout comme l'ordinateur qui héberge le serveur Web.

Pour connecter différents appareils, il faut utiliser des adresses IP pour que nous puissions reconnaître les appareils. Il existe des adresses IP de version 4 (sur 32 bits, soit 4 octets) et de version 6 (sur 128 bits, soit 16 octets). La version 4 est actuellement la plus utilisée : elle est généralement représentée en notation décimale avec quatre nombres compris entre 0 et 255, séparés par des points, ce qui donne par exemple 212.85.150.134.

Nous sommes en réseau local donc nous devons utiliser l'adresse IP suivante 192.168.1.X en remplaçant le X par 1, 2 ou 3 en fonction de la configuration du PC et de l'Arduino. Dans notre cas, 192.168.1.1 va représenter le serveur, 192.168.1.2 l'Arduino et 192.168.1.3 le client. Tout cela peut être représenté par la figure suivante.

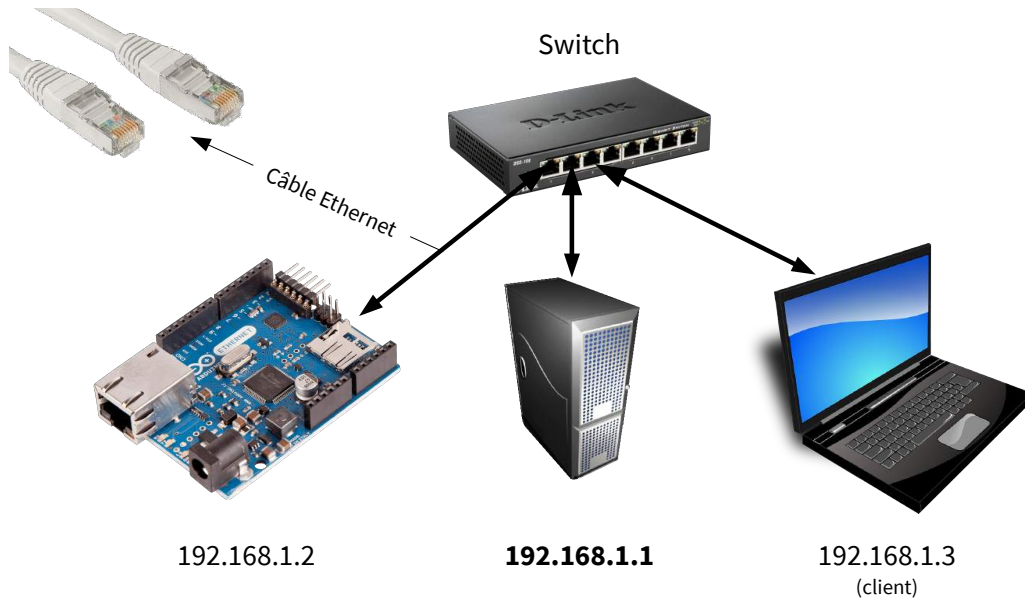


FIGURE 5.1 – Schéma de notre réseau local

Il nous reste plus qu'à configurer le PC et l'Arduino avec les bonnes adresses IP. Concernant l'Arduino, il suffit d'invoquer la commande `Ethernet.begin(MAC, IP)` avec une adresse MAC quelconque et l'IP donnée. Pour notre serveur, il faut aller dans « Paramètres système » puis cliquer sur « Réseau ». Ensuite, il faut aller dans la partie « Filiaire » puis cliquer sur le bouton « Option... ». Enfin, dans l'onglet « Paramètre IPv4 », nous pouvons insérer la configuration en sélectionnant « Manuel » puis en ajoutant l'IP en cliquant sur « Ajouter » comme le montre la figure 5.2.

À l'avenir, si nous utilisons réellement notre projet, l'Arduino et le serveur n'iraient plus sur un switch mais seraient branchés sur une *box*, le site et la base de données seraient hébergés dans un *data-center* pour que l'utilisateur final y ait accès depuis n'importe quelle connexion Internet. Ils n'auront par conséquent plus les mêmes adresses IP.

Modification de Ethernet automatique

Nom de la connexion : Ethernet automatique

Général Ethernet Sécurité 802.1x Paramètres IPv4 Paramètres IPv6

Méthode : Manuel

Adresses

Adresse	Masque de réseau	Passerelle
192.168.1.2	255.255.255.0	192.168.1.1

Ajouter Supprimer

Serveurs DNS :

Domaines de recherche :

ID de client DHCP :

☐ Requiert un adressage IPv4 pour que cette connexion fonctionne

Routes...

Annuler Enregistrer...

FIGURE 5.2 – Configuration réseau du serveur

Chapitre 6

Résultats finaux

Après quelques essais, nous pensons avoir plutôt bien réussi notre projet. Mais il subsiste quelques défauts : une précision du capteur qui n'est pas parfaite mais satisfaisante pour une utilisation domestique, l'Arduino et son *shield* ne peut fonctionner sans la connexion avec l'ordinateur.

Les données obtenus s'affiche très bien sur le site et nous pouvons changer la fréquence d'envois des données.

[image à ajouter]

Conclusion

Webographie

- [1] ESKIMON et OLYTE. *Arduino : Premiers pas en informatique embarquée* — Zeste de Savoir. 2015. URL : <https://zestedesavoir.com/tutoriels/537/arduino-premiers-pas-en-informatique-embarquee/>.
- [2] WIKIPÉDIA. *Qualité métrologique des appareils de mesure* — Wikipédia, l'encyclopédie libre. 2015. URL : http://fr.wikipedia.org/wiki/Qualit%C3%A9_m%C3%A9trologique_des_appareils_de_mesure.
- [3] SEEED STUDIO. *Grove - Barometer Sensor (BMP180)*. 2014. URL : http://www.seeedstudio.com/wiki/Grove_-_Barometer_Sensor_%28BMP180%29.
- [4] WIKIPÉDIA. *I²C* — Wikipédia, l'encyclopédie libre. 2014. URL : <http://fr.wikipedia.org/wiki/I2C>.
- [5] SEEED STUDIO. *Ethernet Shield V2.0*. 2014. URL : http://www.seeedstudio.com/wiki/Ethernet_Shield_V2.0.
- [6] WIKIPÉDIA. *Serveur HTTP* — Wikipédia, l'encyclopédie libre. 2014. URL : http://fr.wikipedia.org/wiki/Serveur_HTTP.
- [7] WIKIPÉDIA. *Adresse IP* — Wikipédia, l'encyclopédie libre. 2014. URL : http://fr.wikipedia.org/wiki/Adresse_IP.

Annexe A

Code complet pour l'Arduino

```
1  /* Inclusion des bibliothèques */
2  #include <Wire.h>
3  #include <SPI.h>
4  #include <Barometer.h>
5  #include <EthernetV2_0.h>
6
7  #define DEBUG true // Pour déboguer
8
9  #define W5200_CS 10
10 #define SDCARD_CS 4
11
12 #define NB_RELEVES 10 // Nombre de relevés par passage
13 #define INTERVALLE 15 * 60 * 1000 // Intervalle entre les envois de données
14 #define TEMPS INTERVALLE / NB_RELEVES // Temps entre chaque relevé
15
16 Barometer capteur; // Déclaration du capteur
17
18 byte MAC[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
19 IPAddress IP(192, 168, 1, 3); // Adresse IP de l'Arduino
20
21 EthernetClient client; // Déclaration du client Ethernet
22
23 char serveur[] = "192.168.1.2"; // Adresse IP du serveur
24
25 unsigned long tempsDerniereConnexion;
26 boolean dernierConnecte(false);
27
28 void setup()
29 {
30     capteur.init(); // Initialisation du BMP180
31
32     /* Désactivation de la carte SD */
33     pinMode(SDCARD_CS, OUTPUT);
34     digitalWrite(SDCARD_CS, HIGH);
35
36     delay(1000);
37
38     Ethernet.begin(MAC, IP); // Démarrage de la connexion Ethernet
39
40     if (DEBUG)
41     {
42         Serial.begin(9600); // Démarrage de la liaison série
43         Serial.print("Mon adresse IP : ");
44         Serial.println(Ethernet.localIP());
45     }
46 }
47
48 void loop()
```

```

49 {
50   /* Affichage des résultats des requêtes */
51   if (DEBUG)
52   {
53       while (client.available())
54       {
55           char c(client.read());
56           Serial.print(c);
57       }
58   }
59
60   if (!client.connected() && dernierConnecte)
61   {
62       Serial.println("Déconnexion.");
63
64       client.stop();
65   }
66
67   /* moyennes : tableau de 2 cases
68   - 1re : moyenne de la température
69   - 2de : moyenne de la pression */
70   float moyennes[2] = {};
71
72   /* Relevés des données */
73   for (int i(0) ; i < NB_RELEVES ; i++) // On fait la somme de 10 relevés de température et de
74   ↪ pression.
75   {
76       moyennes[0] += capteur.bmp085GetTemperature(capteur.bmp085ReadUT()); // Température
77       moyennes[1] += capteur.bmp085GetPressure(capteur.bmp085ReadUP()); // Pression
78
79       delay(TEMPS);
80   }
81
82   moyennes[0] = moyennes[0] / NB_RELEVES;
83   moyennes[1] = moyennes[1] / NB_RELEVES / 100; // Division par 100 pour convertir les Pa en hPa
84
85   /* Affichage des valeurs */
86   if (DEBUG)
87   {
88       Serial.print("Moyennes : ");
89       Serial.print(moyennes[0]);
90       Serial.print(" °C et ");
91       Serial.print(moyennes[1]);
92       Serial.println(" hPa");
93   }
94
95   if (!client.connected() && (millis() - tempsDerniereConnexion > INTERVALLE))
96   {
97       if (client.connect(serveur, 80)) // Connexion au port 80
98       {
99           if (DEBUG)
100             Serial.println("Connexion.");
101
102           /* Envoi de la requête */
103           client.print("GET /arduino.php?pass=123&temperature=");
104           client.print(moyennes[0]);
105           client.print("&pression=");
106           client.print(moyennes[1]);
107           client.println(" HTTP/1.1");
108           client.println("Host: 192.168.1.2");
109           client.println("User-Agent: Arduino Ethernet");
110           client.println("Connection: close");
111           client.println();
112
113           tempsDerniereConnexion = millis();

```



```
113     }
114     else
115     {
116         if (DEBUG)
117         {
118             Serial.println("Erreur de connexion.");
119             Serial.println("Déconnexion.");
120         }
121
122         client.stop();
123     }
124 }
125
126 dernierConnecte = client.connected();
127 }
```