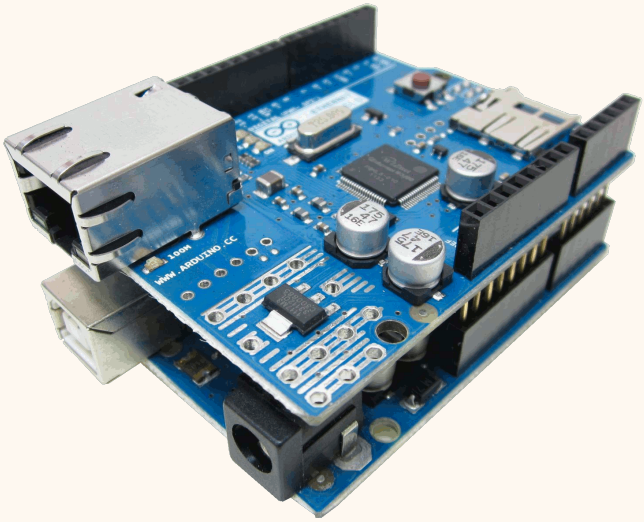
Travaux personnels encadrés

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 ?

ARDUINO
OPEN-SOURCE
COMMUNITY

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 utilisés pour nous rendre le quotidien meilleur. Ce domaine utilise quasiment tout les techniques existantes : nous pouvons trouver aussi bien de l'électronique que de la physique ou encore de l'informatique. Ces dispositifs perment, entre autre, de contrôler notre maison depuis l'extérieur : fermeture et ouverture des volets, déclenchement du chauffage, sécuriser notre habitat, ...Mais il peut également servir à augmenter notre comfort 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 décidés 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 problèmatique, 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 deliverai, 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 détaiils notre démarche pour créer cet objet avec la partie électronique et site Web. 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

Table des figures

Chapitre 1

Présentation du projet

Chapitre 2

Partie électronique

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

2.1 Une plateforme pour reçevoir les données

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

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

2.1.1 Qu'est-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 des contrôler différents récepteur 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 commandés à partie d'un code compilé sur un ordinateur puis téléversé sur un microcontrôleur.

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

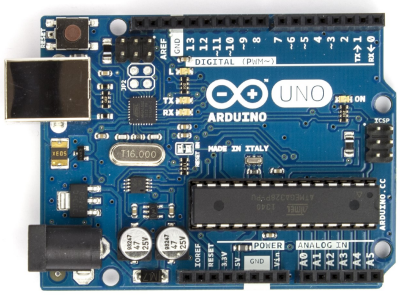


FIGURE 2.1 – Un Arduino Uno R3

2.1.2 Programmation d'un Arduino

Pour programmer un Arduino, il faut utiliser un ordinateur avec l'IDE [1] du constructeur. Après l'avoir installé et configuré pour qu'il programme la bonne carte, nous pouvons y insérer du code.

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.

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

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, on peut téléverser le code suivant à l'Arduino.

```
#define LED 13 // On stocke l'entrée dans un constante.

/* Phase d'initialisation */
void setup()
{
  pinMode(LED, OUTPUT); // On dit que la LED est une sortie.
}

/* Boucle infinie */
void loop()
{
  digitalWrite(LED, HIGH); // On éteint la LED.
  delay(1000);             // On attend 1000 ms = 1 s.
  digitalWrite(LED, LOW);  // On allume la LED.
  delay(1000);             // On attend 1 s.
}
```

Comme vous pouvez le voir, le code est relativement simple. C'est pourquoi les plateformes Arduino sont conseillés 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, hydrolique, ... Les capteurs sont utilisés quasiment partout et il en existe beaucoup : certains peuvent capteur 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 petit valeur mesurable,
- la résolution : c'est la plus petite variation que la capteur est capable de mesurées. 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 la rapport entre la variation du signal d'entrée et la variation du signal du sortie ;
- l'exactitude : elle indique le pourcentage d'erreurs commises ;
- la justesse : c'est l'aptitude à donner un valeur juste, c'est l'écart entre le résultat moyen et la valeur vraie ;
- la fidélité : elle définit la dispersion des valeurs relevées.

En fonction de ces différentes caractéristiques, nous pouvons choisir un capteur.

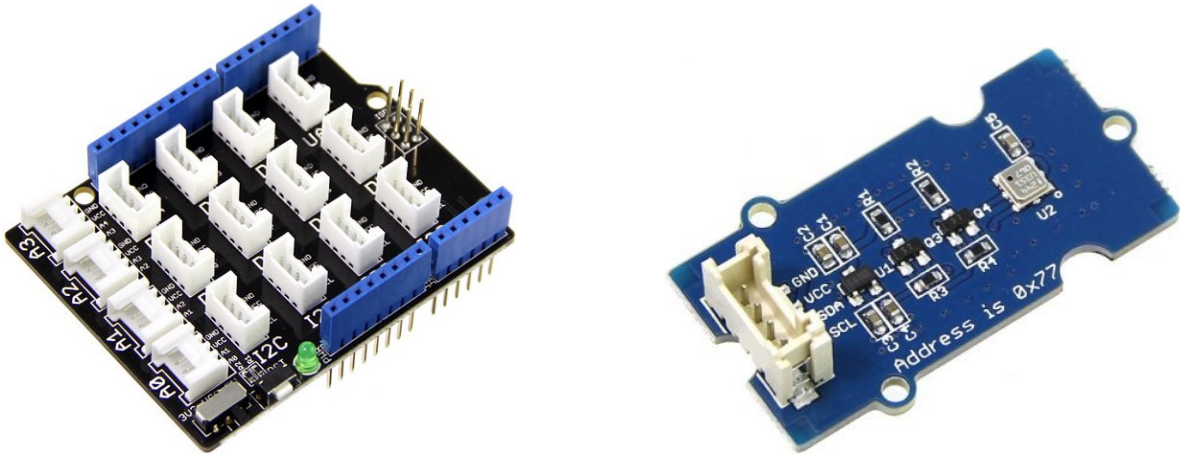


FIGURE 2.2 – Le Base Shield V2 et le BMP180

2.2.2 Capteur utilisé

Comme vu dans le premier chapitre, notre capteur se doit de capter le température ainsi que la pression. Au lycée, nos deux professeurs nous ont orientées 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 cablage facile sans rajout de composants électroniques.

Protocole I²C

Le capteur BMP180 se pilote à l'aide du protocole I²C. Ce protocole est un bus série crée 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 bibliothéque [2] qui va nous faciliter le travail. Il suffit alors de décompresser l'archive dans le dossier des bibliothéques Arduino.

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

```
1 #include <Barometer.h> // On inclus la bibliothéque 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 {
```

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

```
  Serial.begin(9600); // On initialise le communication série
  capteur.init();     // et le capteur.
}

void loop()
{
  /* On stocke la température et la pression. */
  temperature = capteur.bmp085GetTemperature(capteur.bmp085ReadUT());
  pression = capteur.bmp085GetPressure(capteur.bmp085ReadUP());

  /* On affiche les données. */
  Serial.print("Température : ");
  Serial.print(temperature);
  Serial.println(" °C");
  Serial.print("Pression : ");
  Serial.print(pression);
  Serial.println(" Pa");
  Serial.println();

  delay(1000);
}
```

Comme nous allons envoyer les données sur le serveur tous les quartes 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 extrèmes 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.

```
// ...
float moyenneTemperature(0);

for (int i(0) ; i < 10 ; i++)
{
  moyenneTemperature += capteur.bmp085GetTemperature(capteur.bmp085ReadUT());
  delay(90000);
}

moyenneTemperature /= NB_RELEVES;
// ...
```

2.3 Faire communiquer 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 lancée des requêtes TCP/UDP sur un réseau Ethernet. Il faut bien évidement le relier à un *switch* ou à une *box* par un câble Ethernet.

Pour programmer ce *shield*, il faut aussi utiliser la bibliothéque [3] proposée par Seeed Studio.

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

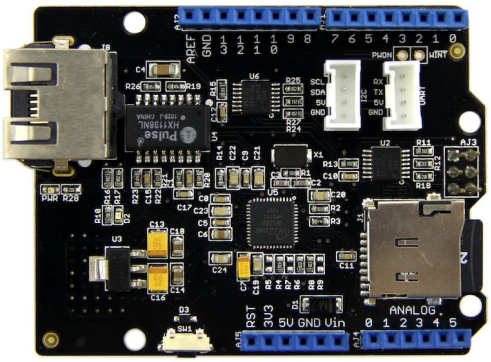


FIGURE 2.3 – L'Ethernet Shield V2.0

Bibliographie

[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. *I^2C — 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.

Annexe A

Code complet pour l'Arduino

```
#include <Wire.h>
#include <SPI.h>
#include <Barometer.h>
#include <EthernetV2_0.h>

#define DEBUG true // Pour déboguer

#define W5200_CS 10
#define SDCARD_CS 4

#define NB_RELEVES 10 // Nombre de relevés par passage
#define INTERVALLE 1.5 * 1000 // Intervalle entre les envois de données
#define TEMPS_INTERVALLE / NB_RELEVES // Temps entre chaque relevé

Barometer capteur;

byte MAC[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress IP(192, 168, 1, 3);
IPAddress DNS(192, 168, 1, 1);

EthernetClient client;

char serveur[] = "192.168.1.2";

unsigned long tempsDerniereConnexion;
boolean dernierConnecte(false);

void setup()
{
  capteur.init();

  pinMode(SDCARD_CS, OUTPUT);
  digitalWrite(SDCARD_CS, HIGH);

  delay(1000);

  Ethernet.begin(MAC, IP);

  if (DEBUG)
  {
    Serial.begin(9600);
    Serial.print("Mon adresse IP : ");
    Serial.println(Ethernet.localIP());
  }
}

void loop()
{
  if (DEBUG)
```

```
50    {
51      while (client.available())
52      {
53        char c(client.read());
54        Serial.print(c);
55      }
56    }
57
58    if (!client.connected() && dernierConnecte)
59    {
60      Serial.println("Déconnexion.");
61
62      client.stop();
63    }
64
65    float moyennes[2] = {};
66
67    for (int i(0) ; i < NB_RELEVES ; i++)
68    {
69      moyennes[0] += capteur.bmp085GetTemperature(capteur.bmp085ReadUT());
70      moyennes[1] += capteur.bmp085GetPressure(capteur.bmp085ReadUP());
71
72      delay(TEMPS);
73    }
74
75    moyennes[0] = moyennes[0] / NB_RELEVES;
76    moyennes[1] = moyennes[1] / NB_RELEVES / 100;
77
78    if (DEBUG)
79    {
80      Serial.print("Moyennes : ");
81      Serial.print(moyennes[0]);
82      Serial.print(" °C et ");
83      Serial.print(moyennes[1]);
84      Serial.println(" hPa");
85    }
86
87    if (!client.connected() && (millis() - tempsDerniereConnexion > INTERVALLE))
88    {
89      if (client.connect(serveur, 80))
90      {
91        if (DEBUG)
92          Serial.println("Connexion.");
93
94        client.print("GET /arduino.php?pass=123&temperature=");
95        client.print(moyennes[0]);
96        client.print("&pression=");
97        client.print(moyennes[1]);
98        client.println(" HTTP/1.1");
99        client.println("Host: 192.168.1.2");
100       client.println("User-Agent: Arduino Ethernet");
101       client.println("Connection: close");
102       client.println();
103
104       tempsDerniereConnexion = millis();
105     }
106     else
107     {
108       if (DEBUG)
109       {
110         Serial.println("Erreur de connexion.");
111         Serial.println("Déconnexion.");
112       }
113
114       client.stop();
```

```
    }
  }

  dernierConnecte = client.connected();
}
```