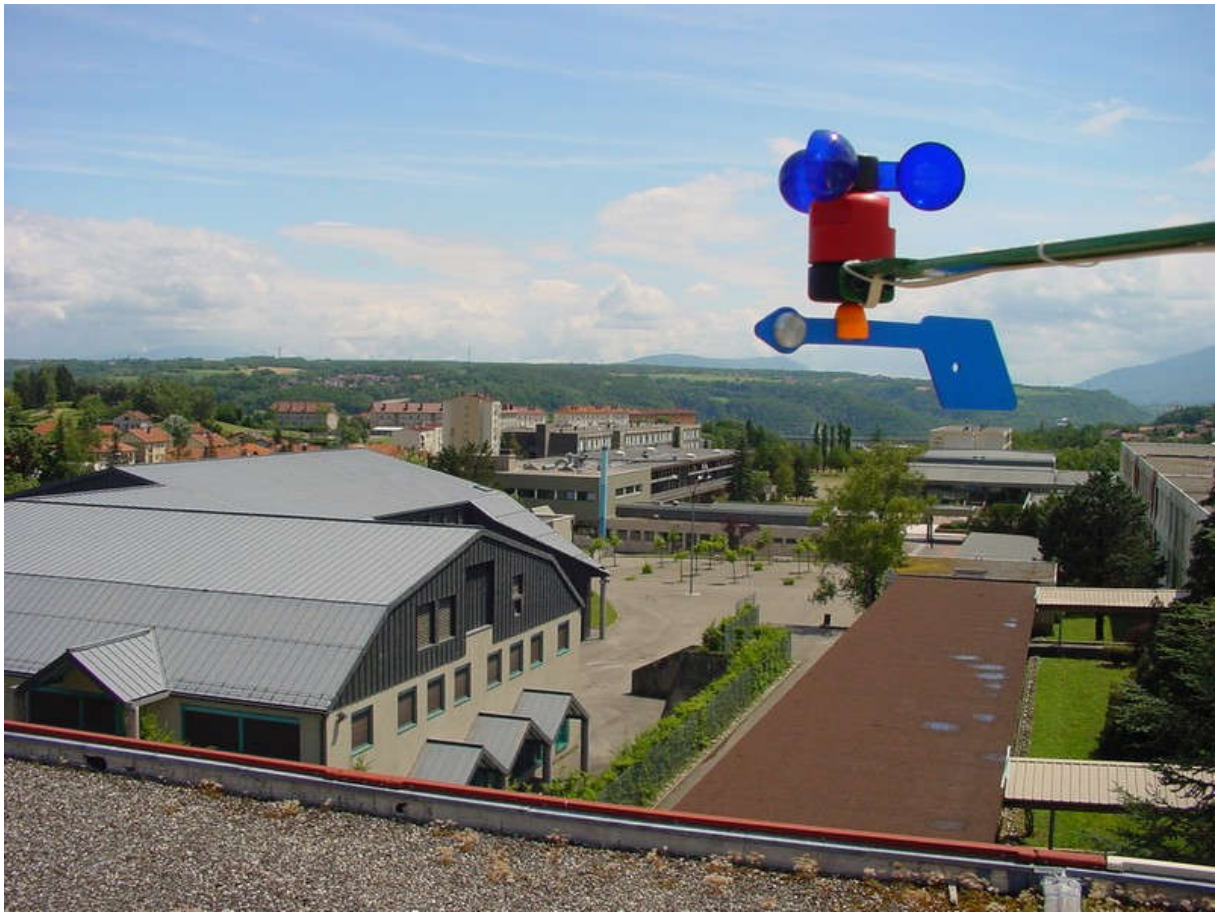
COULLON Camil Tle S1

Dossier-projet de l'épreuve orale du projet d'Informatique et des Sciences du Numérique :

La station météorologique Delphi



http://isnstex.free.fr/

Problématique : Comment exploiter, organiser et présenter les données recueillies par une station météo ?

Membres de l'équipe sur ce projet :

Camil COULLON Swan LAUNAY Judith MILLET

2017-2018

I- Introduction commune

La météo est un phénomène à la fois remarquable, mais aussi d'une grande complexité. Cette complexité se traduit par une analyse théorique approximative et une prévision qui nécessite une très grande puissance de calcul. En effet, prédire le comportement de la masse d'air nécessite la prise en compte d'une multitude de paramètres différents (température, humidité, ensoleillement, influence humaine… etc). En revanche, d'un point de vue expérimental (d'analyse en temps réel) il est possible de comprendre en partie ce qu'il est en train de se passer à l'échelle locale. C'est pourquoi on trouve de nombreuses stations météos disséminées un peu partout en France, qui enregistrent des données : soient de sorte à pouvoir prévoir le temps, soient de sorte à analyser l'évolution du dérèglement climatique au fil des années.

Présentation et limites du projet

Au début du projet,

Nous avions à notre disposition une station météorologique. Cette station comporte un ensemble de capteurs tels qu'un capteur de température, d'humidité, de pression, de vitesse du vent, ainsi qu'une girouette pour la direction du vent. Ils communiquent leurs données à une carte Arduino, qui les récupère et les envoie sous forme d'un URL à la base de données. La base de données s'occupe ensuite de stocker ces données et de les organiser dans un tableau. Ces données sont ensuite récupérées par la page d'accueil du site (index.php) qui les affichent sous forme graphique. En effet avec la place grandissante qu'occupe internet dans notre quotidien, un des objectifs que nous nous sommes fixés était d'utiliser comme support principal le Web. Une page web ne nécessite pas de prérequis d'un point de vue logiciel car on retrouve sur à peu près tous les appareils modernes (ordinateur, smartphone, tablette) un navigateur web.

Pour résumer, le but du projet réside dans le fait d'établir une liaison entre un site web et une station météo pour ainsi traiter et présenter les relevés de façon ergonomique.

Répartition des tâches

De sorte à répartir équitablement le travail en tous les membres de l'équipe, tout en laissant une possibilité d'extension du projet, nous avons décidé de découper la problématique en trois parties :

Ainsi, je me suis occupé du dialogue avec la station ainsi que du contact d'une page PHP afin de transmettre des données facilement exploitables.

Judith MILLET, quant à elle, s'est occupée de tout ce qui est en rapport avec la base de données, et notamment, de la mise en place des différentes pages PHP permettant l'approvisionnement de cette base de données.

Enfin, *Swan LAUNAY* devait proposer une présentation graphique de ces données de façon ergonomique, ou plus largement de donner à l'utilisateur les outils pour consulter/exploiter nos données. J'avais également la possibilité de proposer un service de prévision météo basé sur l'historique des relevés.

Au final on peut schématiser les flux d'informations, et les différentes parties du projet comme

ceci :

Illustration 1: Modélisation de la totalité des échanges entre les différentes parties

Environnement de travail

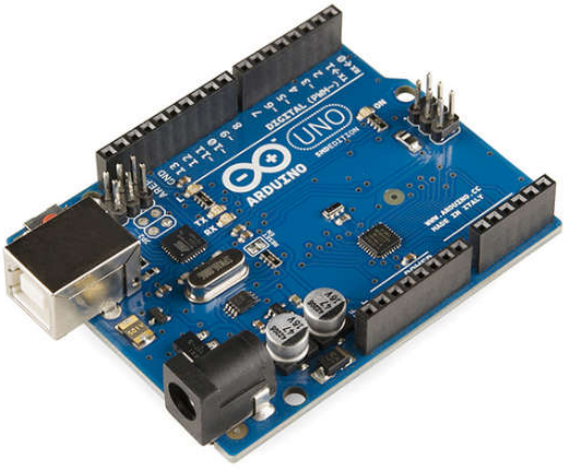
La réalisation de ce projet a fait appel à différents services et différentes ressources.
Les langages à la base de notre projet sont :

- PHP
- HTML/CSS
- SQL
- Langage Arduino (une variante du C)
- JavaScript (/JQuery)

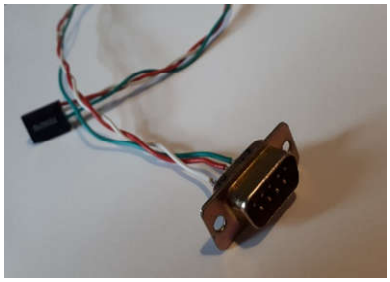
De sorte à faciliter les modifications de chacun, nous sommes passés par un système de versionnage de code à l'aide de la plateforme GitHub (un hébergeur de repositories), et du logiciel Git. A savoir que lors de chaque commit (paquet de modifications) envoyé sur GitHub, les fichiers modifiés étaient automatiquement envoyés sur notre serveur de développement : http://akashita.mooo.com/isn. Ce serveur se situe à mon domicile et tourne sur la dernière version de Debian (une distribution Linux). La partie d'hébergement web est gérée par Apache 2.4.25, PHP 5.6 et MySQL 5.7, ces services sont dans des versions suffisamment récentes pour interpréter correctement les nouveautés de ces langages.

I- Acquisition des donnes, commutation avec la station.

A- Matériel utilisé (Hardware).



> Le plus judicieux était d'utiliser un Arduino car l'interface Arduino IDE est très utile pour des projets de notre cas. Nous avions songé à utiliser un Raspberry pie mais l'interface aurait été trop compliquer.

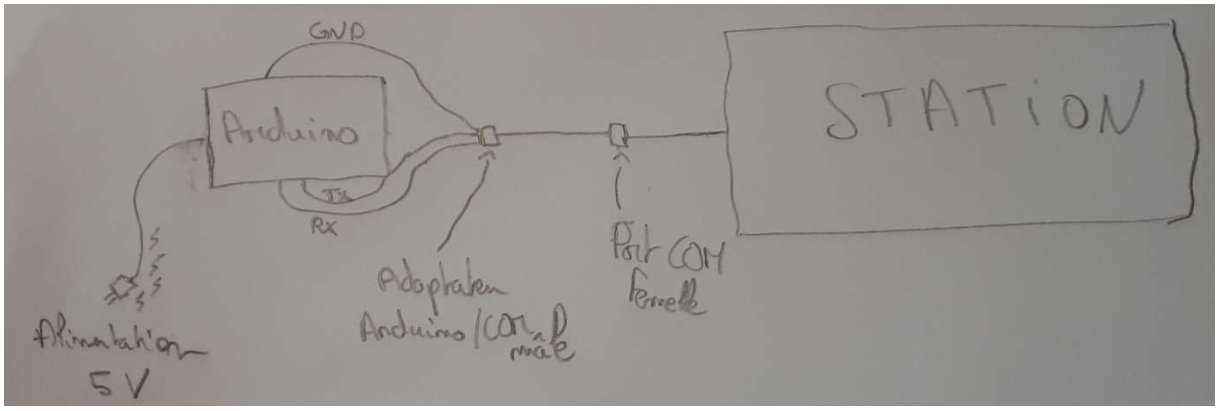


Pour pouvoir acquérir les donnes de la station nous avons choisi d'utiliser une liaison filaire avec l'adaptateur port COM/Arduino montrée ci-contre. Le port COM est ensuite branche à la station.

Pour relier l'Arduino et l'adaptateur, on utilise des « jumpers ».

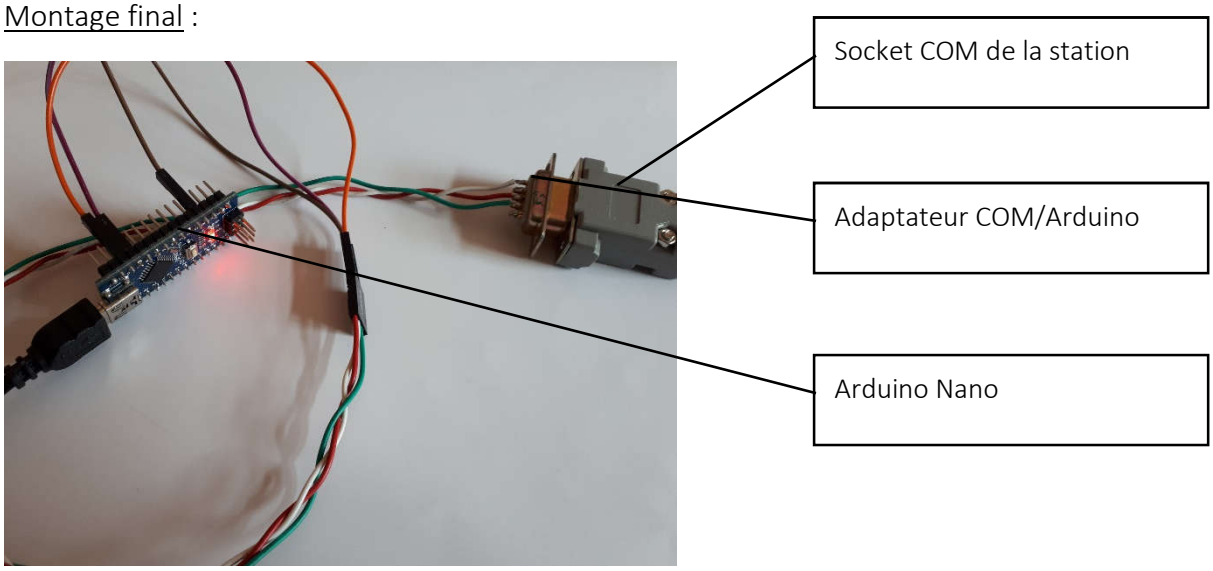
B- Schématisation du problème.

Schéma communication Station/Arduino :



Comme on peut le voir ci-dessus, il y a trois fils qui sortent de l'adaptateur et qui sont branche sur l'Arduino. L'Arduino doit être branchée au ground de la station (ground commun) et le pin RX de l'Arduino au pin TX de la station et inversement, le pin TX de l'Arduino au pin RX de la station.

Montage final :



C- Software serial.

Pour communiquer avec la station, il fallait que je puisse envoyer des commandes par la « voie série » de la station, le problème est que l'Arduino est capable de communiquer qu'avec une seule voie série. Il fallait donc que je crée une voie série Arduino/station différente de la voie série Arduino/ordinateur. Il existe une bibliothèque Arduino qui permet d'utiliser deux voies séries en même temps, appelée « Software serial ». Je me suis donc inspiré de l'exemple ci-dessous pour bâtir mon programme.

SoftwareSerialExample

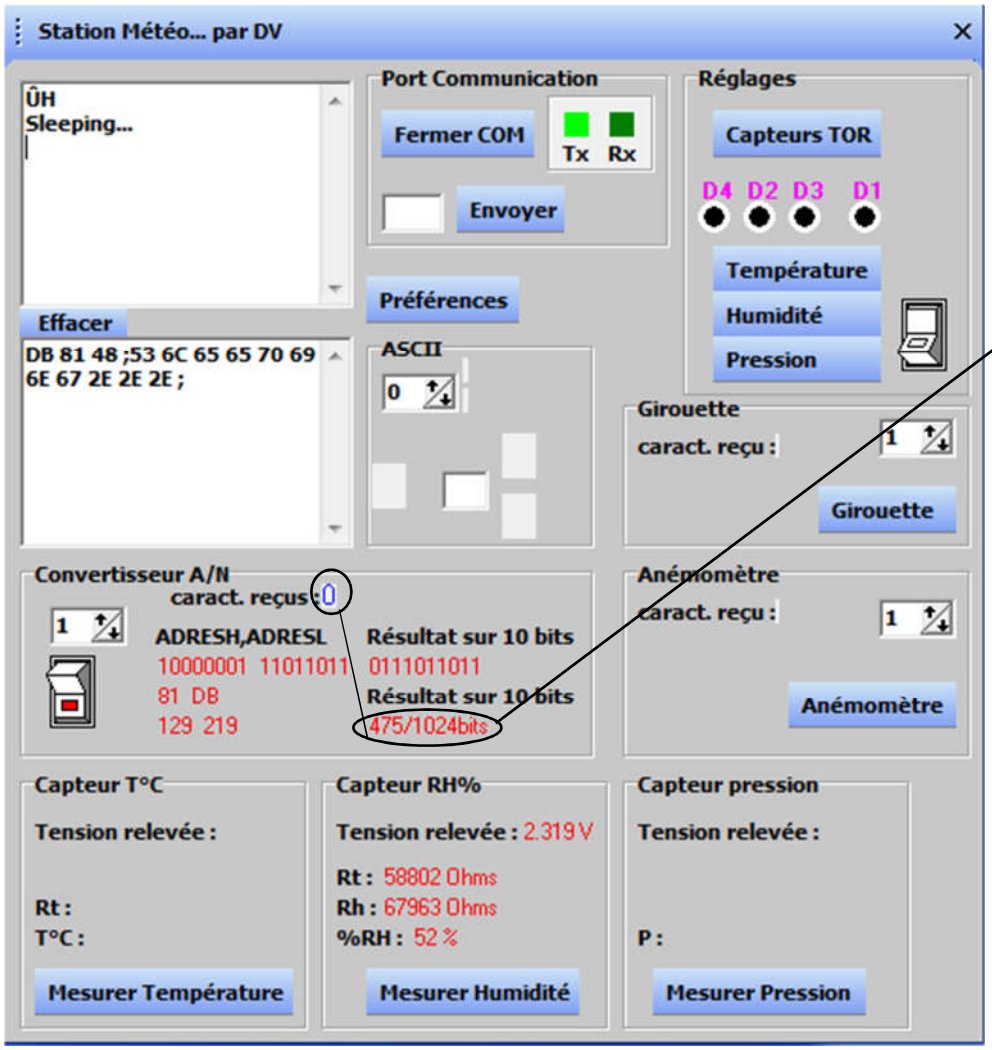
```
 This example code is in the public domain.

 */
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }


  Serial.println("Goodnight moon!");

  // set the data rate for the SoftwareSerial port
  mySerial.begin(4800);
  mySerial.println("Hello, world?");
}
```

```
void loop() { // run over and over
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
}
```

II- Traiter les donnes reçus.

A- CAN

Pour envoyer les donnes de température, d'humidité et de pression atmosphérique la station utilise un convertisseur analogique numérique. En interrogant la station sur l'humidité par exemple, on obtient ceci :



Résultat sur 10 bits qui sera renvoyé par la station dans le buffer de l'Arduino lors d'une demande de mesure d'humidité.

La station nous renvoie 2 octet de 8 bits lorsque nous l'interrogeons et qu'elle a besoin d'utiliser le CAN (uniquement pour la température, la pression et l'humidité). Pour avoir donc un mot sur 10 bits il fallait trouver une solution. J'ai remarqué que c'est l'octet de poids faible qui était transmis en premier puis l'octet de poids fort. J'ai vu aussi que sur l'octet de poids fort, la station prenait que les deux premiers bits pour les « additionner » avec les 8 bits de l'octet de poids

faible. J'ai donc dû inverser le poids des octets et faire un « et logique » entre l'octet de poids faible et « 00000011 » pour avoir les 2 premiers bits de l'octet de poids fort.

```
b1 = stationserial.read();
b2 = stationserial.read();

while (stationserial.available()) stationserial.read();

Serial.print(b1);
Serial.print(" - ");
Serial.print(b2);
Serial.println();

b1 = Bit_Reverse(b1);
b2 = b2 & byte(192);
b2 = Bit_Reverse(b2);
int b = b2*256;
int bfinal = b1+b;
Serial.println(bfinal);
```

```
byte Bit_Reverse(byte Monbyte) {
  byte  rtn = 0;
  for (byte i=0; i<8; i++) {
    bitWrite(rtn, 7-i, bitRead(Monbyte, i));
  }
  return rtn;
}
```

La fonction Bit_Reverse(), parcours chaque bit de l'octet et l'écrit dans l'autre sens.

B- Transformer les donnes.

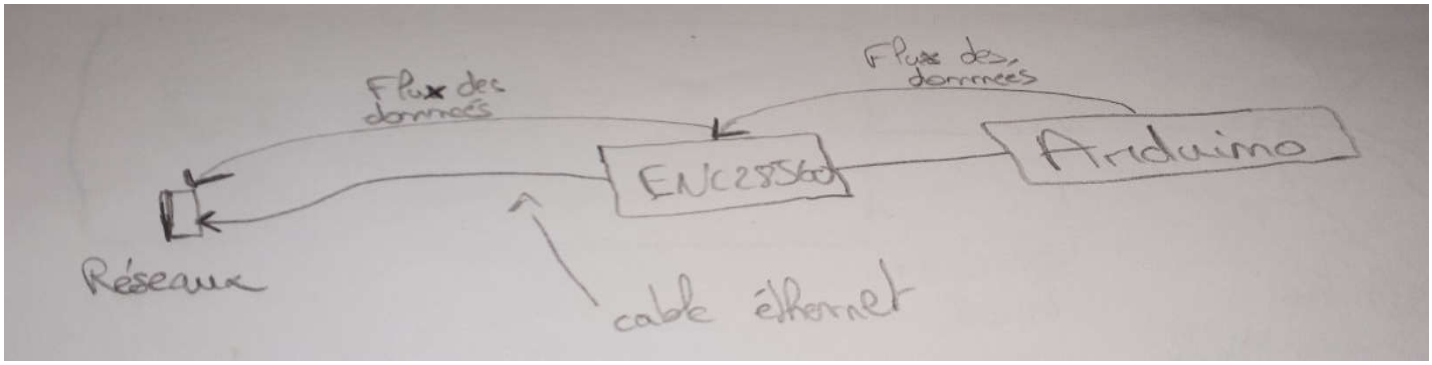
Les données que nous renvoie la station sont de type « byte » et ne sont donc pas compréhensible par l'utilisateur. J'ai du donc aller chercher dans les programmes sources de la station les calculs effectués sur les « bytes » pour qu'ils deviennent compréhensibles par l'utilisateur. Cette tache fut très difficile pour moi et m'as pris beaucoup de temps car j'ai dû comprendre le code de la station qui n'est pas le miens, ce qui est un exercice assez difficile. Ensuite, j'ai dû adapter ces formules qui sont écrites en langage Delphi en C-basics d'Arduino. Enfin, j'ai stocké ces valeurs dans des variables que j'utiliserai par la suite.

III- Communiquer les données.

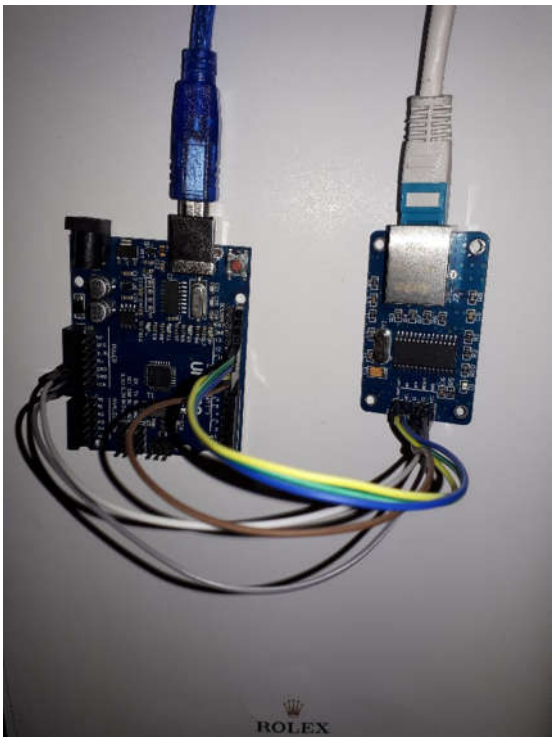
A- Hardware.

> Pour transmettre les donnes acquises par la station nous avons décidé d'utiliser un protocole Ethernet qui contacte une page PHP et qui envoie les donnes à travers l'URL (méthode GET). Pour ce faire j'ai utilisé le module Ethernet, ENC28J60. Ce module est un bon compromis par rapport au Shield Ethernet car il est plus compact et la bibliothèque est plus facile d'accès.

B- Schématisation.



Après quelque recherche sur internet j'ai trouvé sur une page GitHub, l'ordre des branchements.



Physical Installation

PIN Connections (Using Arduino UNO):

```
VCC -   3.3V
GND -    GND
SCK - Pin 13
SO  - Pin 12
SI  - Pin 11
CS  - Pin 10 # Selectable with the ether.begin() function
```

Le problème est que lorsque j'exécutait mon programme rien ne ce passait. L'erreur venait du fait qu'il fallait alimenter le module en 5V et non pas en 3.3V. Enfin, on connecte le module Ethernet sur notre réseau via un câble Ethernet.

C- Programme (Software).

Pour commencer je n'avais aucune idée sur quoi partir. J'ai donc commencé à chercher des exemples sur les forum Arduino pour déjà essayer de bien comprendre le fonctionnement du module. Mais vu que mes recherches n'aboutissaient à rien je suis donc aller me renseigner directement sur le fonctionnement de la bibliothèque Ethercard associé au module, que j'ai dû télécharger car ce n'est pas une bibliothèque native d'Arduino. En parcourant les classes de la bibliothèque, j'ai pu appeler uniquement les fonctions qui m'intéressait. Le site ou j'ai pris la bibliothèque, explique aussi très bien le rôle de chaque fonction et de chaque classe : https://jeelabs.org/pub/docs/ethercard/. Le programme contacte une page PHP crée par Judith Millet, et renvoie des valeurs à travers l'url (Méthode GET).

Setup qui permet d'initialiser la communication avec la station et d'initialiser le module Ethernet.

```
void setup()
{
    // Ouvre la voie série avec l'ordinateur
    Serial.begin(9600);
    // Ouvre la voie série avec la station
    stationserial.begin(4800);

   if (ether.begin(sizeof Ethernet::buffer, mymac) == 0) // test pour savoir si carte est branchee
     Serial.println(F("Probleme d'acces a la carte ethernet"));
   if (!ether.dhcpSetup()) // test pour savoir si serveur dhcp est existant
      Serial.println(F("erreur DHCP"));

  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
  ether.printIp("DNS: ", ether.dnsip);

  char websiteIP[] = "192.168.0.13"; // ardesse IP qui recoit la requette
  ether.parseIp(ether.hisip, websiteIP);

  ether.printIp("SRV: ", ether.hisip);
}
```

```
ether.packetLoop(ether.packetReceive());
String data = String("index.php?T=") + String (Tt) + String("&P=")+ String(Pmbar)+ String("&A=")+ String(Va)+ String("&H=")+ String(Hr)+ String(

char __data[100];
data.toCharArray(__data, sizeof(__data));
Serial.begin(9600);
Serial.println(F("\n[webClient]"));
Serial.println(__data);
```

// Fonction qui transforme le String en char.

```
if (millis() > timer) { // boucle qui envoie les donnees toutes les 10 minutes
  timer = millis() + 600000;
  Serial.println();
  Serial.print("<<< REQ ");
  ether.browseUrl(PSTR("/ethernetcard/"), __data, "192.168.0.13", my_callback);
```

« String data » est un string qui recense toutes les valeurs à renvoyer dans la page PHP. La fonction data.toCharArray() est une fonction qui transforme String data en char car la fonction ether.BrowseURL() ne renvoie que des chars dans l'URL.

IV- Problèmes rencontrés.

Les problèmes majeurs que j'ai rencontré lors de ce projet ont été dans la partie communication entre l'Arduino et la station. En effet, il y avait toujours un problème, soit la station ne fonctionnait pas, soit la bibliothèque software serial ne fonctionnait pas avec la version 1.8 d'Arduino. De plus, la station ne peut communiquer qu'avec un port série, et lorsqu'il fallait que j'exécute l'application de la station chez moi pour pouvoir comprendre son fonctionnement il fallait que je connecte la station a mon pc via un adaptateur PL232 USB /COM. Ayant Windows 10, aucun driver n'existait pour mon adaptateur. J'ai donc dû me faire prêter un pc avec un socket COM pour que je puisse enfin interagir avec la station via l'application consacré.

V- Sources.

- Sources utiliser :

- https://jeelabs.org/pub/docs/ethercard/.
- varrel.fr/station/
- https://github.com/camilcll/ISN.
- Forums Arduino