UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Station Météo Autonome

TW51 – A2016

SUTER Gaétan
GUOLIANG Li
VANDEPOORTER Valentin

Département Génie Mécanique et Conception
Filière Conception des Systèmes Mécatroniques

Responsables de l'UV
AVISSE Bruno
NOBILE Claude

Sommaire

1. Introduction

Nous réalisons dans le cadre de ce projet une station météo à base Arduino pour le club de voile du bassin de Champagney. Cette station météo doit mesurer différents paramètres au bord du lac et les rendre disponibles via internet aux membres du club, et ce afin que ces derniers puissent juger des conditions de navigations avant de se rendre sur place.

Ce projet est la continuation d'un projet de mécatronique précédant dans lequel les différents composants ont été choisis et commandés. Le code en C permettant le fonctionnement de la station avait également été fait.

Nous l'avons donc poursuivit durant ce semestre afin qu'il soit pleinement opérationnel et installable à son terme. Ce rapport, relate les différentes opérations effectuées pour arriver à l'aboutissement de ce projet.

2. Objectifs

L'objectif est d'obtenir à la fin de ce projet une station météo fonctionnelle et prête à être installée sur place.

Du point de vue fonctionnel, elle devra répondre au cahier des charges suivant :

Fonction
• Être capable de mesurer : - la température - la vitesse du vent - la direction du vent
• Pouvoir prendre une photo de la surface du lac
• Résister à l'environnement extérieur (été)
• Etre démontable en Hiver
• Etre autonome électriquement
• Transmettre les données recueillies sur internet
• Pouvoir consulter les données en ligne via un mot de passe

3. L'existant

Lorsque nous avons repris le projet, le matériel suivant était présent :

- L'Arduino
- Le Shiled Dragino Yun
- La webcam
- Divers câbles
- Le mât portant la girouette et l'anémomètre
- La batterie
- Le régulateur de tension
- Le panneau solaire

La sonde de température de type TMP36 avait été cassée et il fallait en recommander une. Le boitier 3G quant à lui n'était pas présent.

Sur le plan énergétique, le panneau solaire relié à une batterie de 12V via un régulateur de tension permet de rendre la station autonome en électricité.

Le projet initial prévoyait de transmettre les données sur un Dropbox via un service payant Teembo à 7$ par mois.

Le code permettant l'exécution des différentes mesures, la prise de photos et l'envoi des données sur Dropbox était présent, bien que des erreurs soient présentes au sein de la partie mesure.

4. Avancée

La première action a été de réunir le matériel manquant. Une nouvelle sonde de température TMP36 a été commandée. Le boitier 3G n'a pas été commandé tout de suite car nous voulions que la partie mesure fonctionne avant de nous engager dans un abonnement mensuel pour la carte SIM du boitier 3G.

Le code C de l'Arduino permettant d'acquérir les mesures doit être corrigé et le code permettant le transfert des données complètement changé. En effet, la solution retenue précédemment nous parait trop onéreuse. Après avoir étudié certaines pistes et méthodes, nous en avons retenu une : envoyer les mesures directement sur un site internet disposant d'une base de données.

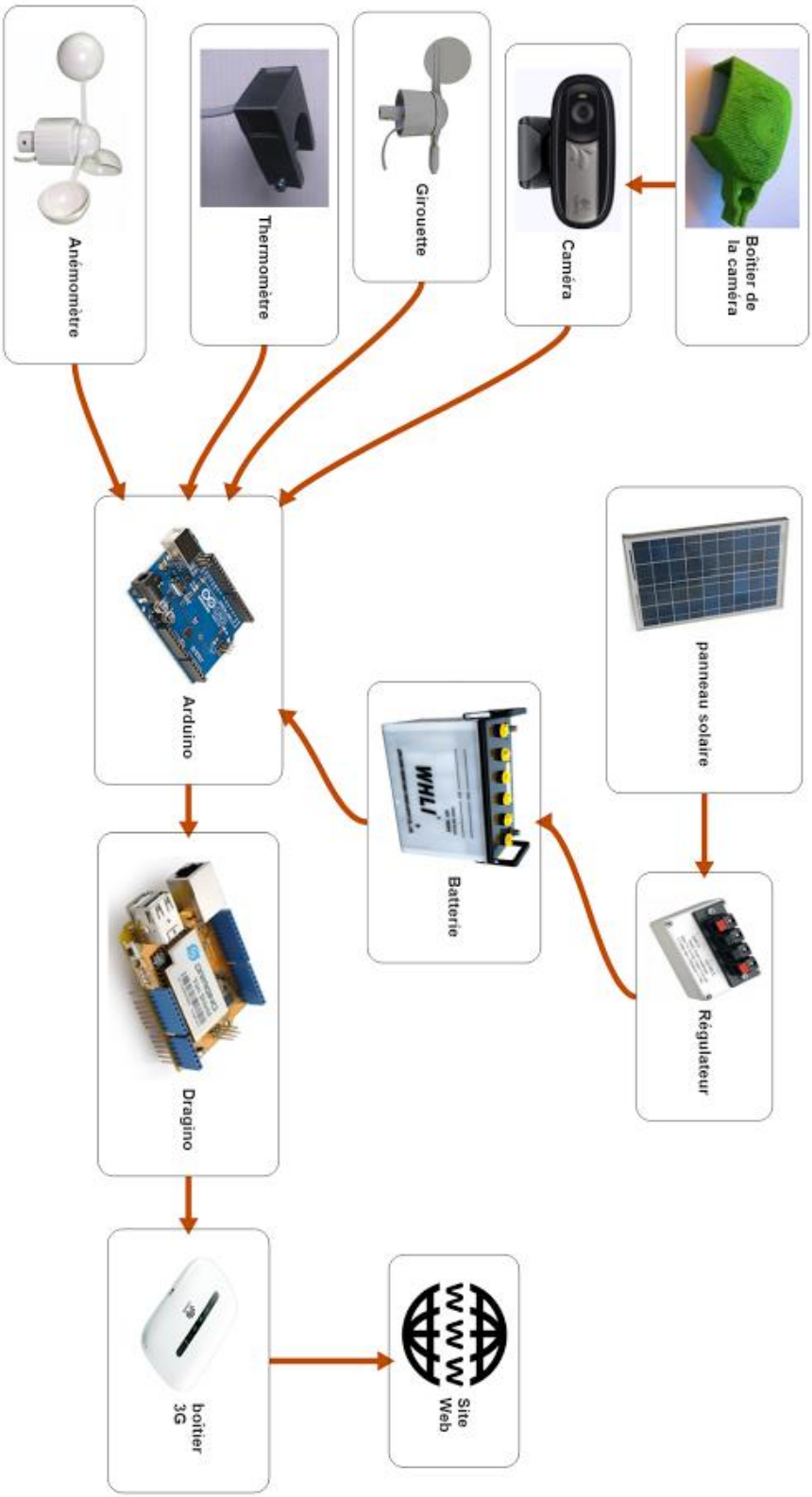
Les mesures seront stockées dans la base de données ce qui permettra de faire des graphiques d'évolution et les résultats seront affichés sur une page du site internet. Le coût de l'hébergement du site internet est beaucoup plus accessible à environ 12 euros par an.

En parallèle, nous avons également décidé de fabriquer des boitiers de protections pour les différents composants, afin qu'ils puissent résister aux conditions extérieures et permettre du même coup une installation plus rapide.

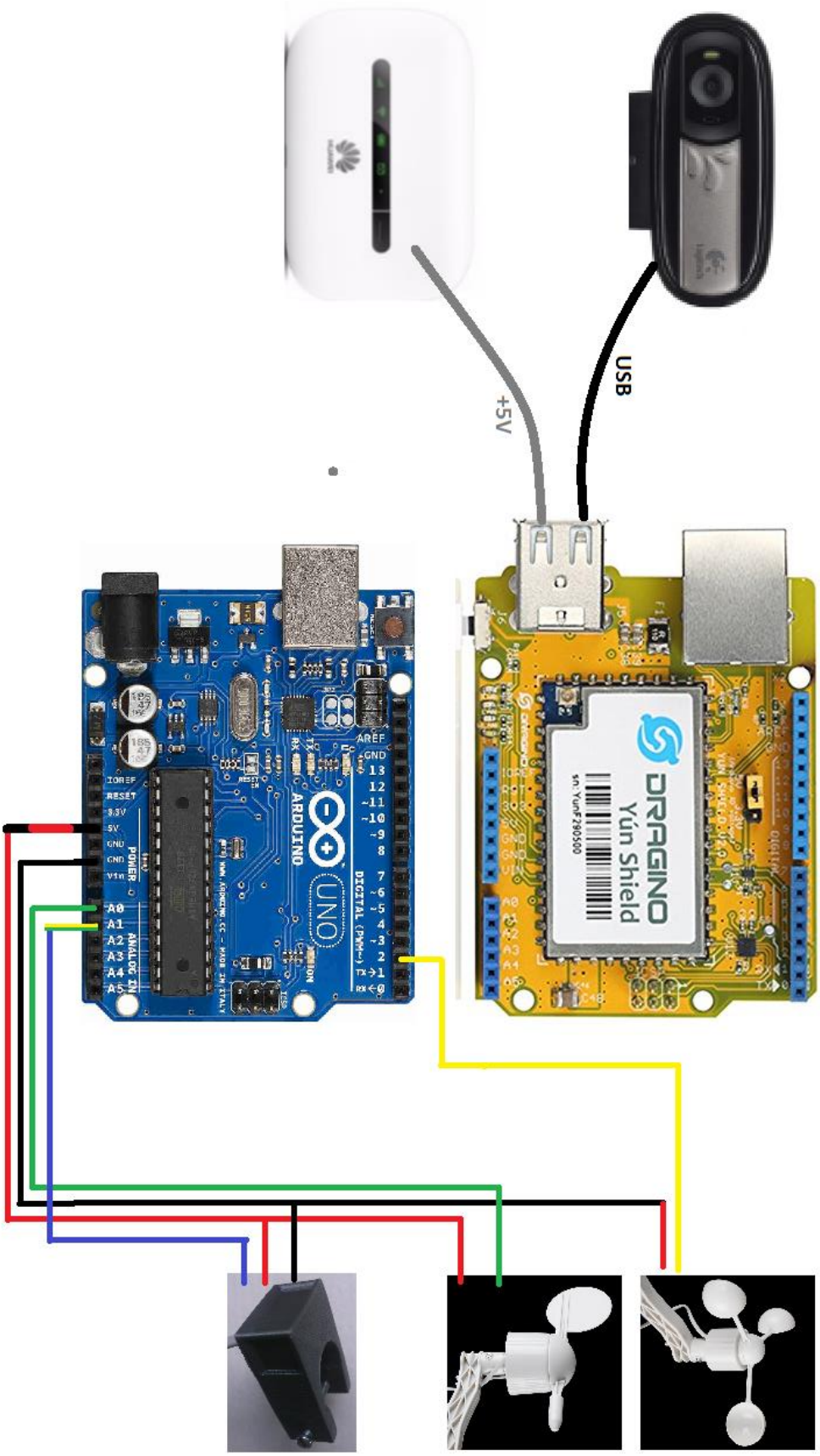
Ce qui nous mène au planning suivant :

Septembre - Octobre	Novembre	Décembre - Janvier
Réalisation de la partie acquisition de mesure	Réalisation de la partie transmission de données (transmission et site internet)	Installation de la station et ajustements
Conception et création des boitiers		

4.1. Architecture du projet



4.2. Connexions



4.3. Programme Arduino

Ce programme sera embarqué dans la station météo, et plus précisément dans la mémoire de l'Arduino, toutes les informations mesurées passeront par lui, deplus il contrôlera via une bibliothèque spécifique le processeur linux qui lui est associé, ce qui fait qu'il sera en charge de la transmission des données vers le site web, que nous verrons un peu plus loin.

Il basé sur une architecture basique que l'on peut trouver sur un Arduino quelques soit son modèle. Néanmoins, au vu de la taille du programme, de nombreux ajouts sont présents, ils ne sont pas essentiels mais aident grandement à la structuration du code.

Nous débutons par déclarer la bibliothèque Process.h qui nous permet de communiquer avec le linux attaché en tant que Shield, ce dernier est une carte additionnelle laissant la possibilité d'upgrader son Arduino en une multitude d'usages différents. Son fonctionnement revient à pouvoir lancer des commandes Linux depuis le programme écrit en C++.

Une seconde librairie, nommée Console.h permet durant le développement d'accéder à des retours d'informations via la console de l'IDE, comme l'on peut le faire avec l'objet Serial si l'Arduino est encore connecté au PC.

Dans un souci de clarté, nous uniformisons les notations, les #define seront inscrit en majuscules ; les fonctions seront nommées tout attaché avec la première lettre de chaque mot en majuscule, sauf le premier, par exemple : uneFonction ; et les variables seront écrites en minuscules et les mots séparés par des underscore (_), par exemple : voici_une_variable. Ceci permet ainsi immédiatement de savoir le type concerné.

```
//librairies nécessaires
#include <Process.h>

//librairies pour le développement
#include <Console.h>

//définition de define (paramétrage des entrés/sorties)
#define DO_BUZ 7 //pour implantation d'un buzzer (dev)
#define DI_ANEMOMETRE_INT 2
#define AI_GIROUETTE 0
#define AI_TEMPERATURE 1
#define DUREE_LECTURE_CAPTEUR 30

#define HEURE_DEBUT 8
#define HEURE_FIN 20

//variables générales
int direction_analog_in[] = {786,405,460,84,92,65,184,127,286,243,630,59
char path_photo_dragino[] = "/mnt/sda1/meteo.png"; //photo_meteo_2
char path_login_photo_web[] = "ftp://stationm:utbmcsm2015@station-meteo-
char path_donnes_web_csv2[] = "http://station-meteo-csm.nhvvs.fr/putdata
int date_heure[6]; //contient annee-mois-jour-heure-minute-seconde
volatile int compteur_vent;
float donnees[3]; //contient température, vitesse du vent et direction
int numero_boucle; //numero d'iteration de la boucle principale
unsigned long tempo = 0;
int derniere_mesure = 0;

//déclarations fonctions
void prendrePhoto();
void envoiPhoto();
void envoiCSV();
void majTime();
void incrementationCompteurVent();
void lectureCapteurs(int duree);
int trouverDirectionProche(int analogiqueGirouette);
String ecrireNombre(int in);
String ecrireTimeComplet();
String ecrireTimeDonneesCSV();
void logC(String txt, int nb);
void cycleMesure();
```

Après les bibliothèques viennent ensuite trois séries de déclarations.

La première défini des mots-clefs de type #define, ces derniers, au moment de la compilation, seront remplacés par la valeur qui leur est attribué. Cela permet de changer rapidement des paramètres du programme sans occuper de la mémoire pendant l'exécution de celui-ci. Ils sont utilisés pour fixer les pins d'entrée/sortie et une grandeur variable, mais ne peuvent contenir que des nombres entiers.

Ensuite nous déclarons les variables globales qui seront utilisées au cours de l'exécution du programme pour gérer les mesures, les données, et la transmission. Nous y trouvons à la fois des constantes, telles que les adresses des différents emplacements de dépose des photos prises, ou de la page qui reçois les données. Il y également quelques variables qui seront utiles tous au long du programme, par exemple le tableau date_heure[6], qui contient toutes les informations temporelles, régulièrement mises à jour et utilisées à de nombreux moments.

L'exécution d'un programme commence par le haut et le parcours vers le bas, il est donc nécessaire lorsque l'on appelle une fonction que le programme sache si elle existe et comment est-elle définie. C'est pourquoi les fonctions sont à créer avant la fonction principale sous Arduino. Mais dans un souci de rigueur et de cohérence, nous nous contenterons d'inscrire ici les prototypes (type de retour, nom et variables d'entrée) annonçant au programme qu'elles existent bien, mais les écrivant plus bas.

4.3.1. Le programme débute

Le programme de l'Arduino débute donc sur la fonction setup, executée une seule fois au lancement, elle a pour rôle d'initier et de préparer les objets, méthodes et composants qui seront utilisés après.

```
45  void setup()
46  {
47    Bridge.begin();
48    Console.begin();
49    //while(!Console);
50    logC("Fin Bridge et Console setup", 0);
51
52    logC("Debut setup", 0);
53    numero_boucle = 0;
54    tempo = 0;
55
56    //pinModes
57    pinMode(DO_BUZ, OUTPUT);
58    pinMode(DI_ANEMOMETRE_INT, INPUT_PULLUP);
59    attachInterrupt(0,incrementationCompteurVent, FALLING);
60
61    tone(DO_BUZ,500,500);
62    logC("Fin du setup", 0);
63  }
```

Les deux premières fonctions appelées sont les deux moyens de communication avec le Linux et l'utilisateur. Cette précipitation n'a pour but que permettre au plus vite le retour d'informations en vue d'un éventuel débogage. C'est d'ailleurs dans ce but que la ligne `//while(!Console);` est présente, elle permet de suspendre le programme le temps que la console soit ouverte, lorsqu'elle n'est pas mise en commentaire.

Nous initialisation ensuite quelques variables, et paramétrons les entrées sorties utilisées.

La fonction tone(), permet la génération d'un son sur un buzzer branché à l'Arduino, ceci dans un but de suivi de l'avancement du programme, c'est plus simple à mettre en place que d'y connecter un écran LCD.

La fonction logC(), permet de centraliser en une fonction les différents moyens de suivre l'action du programme, tels que l'affichage sur la console (comme actuellement), ou aussi d'écrire dans un fichier. Elle prend en paramètres une chaine de caractères accompagnés d'un nombre entier (un troncage est réalisé en cas de nombre réel).

4.3.2. La boucle principale

```
66  void loop()
67  {
68    logC("Loop Debut numero",numero_boucle);
69    majTime(); //mise à jour de l'heure et la date.
70
71    if(constrain(date_heure[3],HEURE_DEBUT,HEURE_FIN) == date_heure[3])//si l'on est dans la période d'activation
72    {
73      logC("Prise de mesures", 0);
74      cycleMesure();
75    }
76    else if(date_heure[3] == HEURE_DEBUT-1) //à moins d'une heure
77    {
78      //on attend un nombre de minutes jusqu'à l'heure pile
79      tempo = 60*(60-date_heure[4]+1);
80      tempo = tempo * 1000;
81      Console.print("Pause partielle :");
82      Console.println(tempo);
83      delay(tempo);
84    }
85    else
86    {
87      //on attend une heure
88      tempo = 60*60000;
89      Console.print("Pause heure :");
90      delay(tempo);
91    }
92    tone(DO_BUZ,1000,50);
93    logC("Loop Fin",0);
```

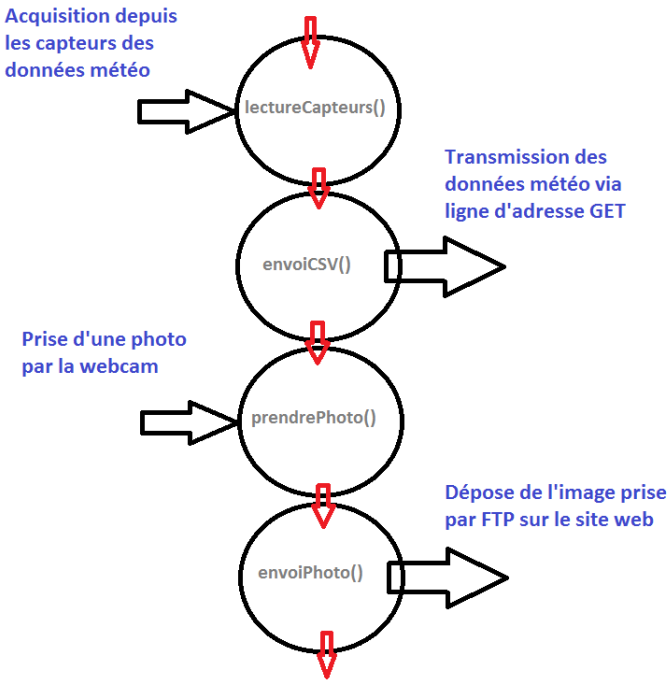
Cette boucle, ne contient qu'une utilité, en fonction de l'instant présent, elle décide si le programme s'exécute ou se met en veille. Pour cela elle compare l'heure actuelle au créneau qui lui est défini plus haut, alors trois cas se présentent.

Si elle est en dehors de la fourchette horaire, alors elle fait une pause d'une heure avant de tester à nouveau. Mais si elle est à moins d'une heure du début de la mise en route, alors elle n'attend qu'un nombre précis de minutes pour que le cycle reprenne à l'heure pile.

Si l'heure est dans le créneau de fonctionnement, alors elle lance la fonction cycleMesure(), qui est chargée de gérer les différentes étapes de la chaine d'acquisition et de transmission.

A partir de ce point, le programme suit un comportement linéaire, il exécutera les étapes successivement, telles qu'elles sont présentées ci-dessous :

```
96  void cycleMesure()
97  {
98    lectureCapteurs(DUREE_LECTURE_CAPTEUR);
99    envoiCSV();
100   prendrePhoto();
101   envoiPhoto();
102
103   Console.println(ecrireTimeDonneesCSV());
104 }
```



4.3.3. Lecture des capteurs

Nous avons trois capteurs à interroger, par deux méthodes différentes.

La vitesse du vent est obtenue par une interruption, c'est-à-dire que dès que la broche D2 est mise à état LOW, cela indique que l'anémomètre a fait un tour. Compter le nombre de tours en un certain temps, permet donc d'obtenir la vitesse.

```
177 void incrementationCompteurVent()
178 {
179   compteur_vent = compteur_vent +1;
180 }
```

L'interruption est un programme très bref, que l'Arduino exécute dès que le signal (ici la broche D2 à LOW) est déclenché. Il met en pause le programme principal le temps qu'il lui faut pour exécuter celui-là.

Nous mettons à zéro le compteur juste après avoir enregistré l'instant absolu (avec la fonction millis() qui donne le nombre de millisecondes écoulées depuis le lancement du programme), et nous relevons la différence juste avant le calcul de la vitesse.

Les mesures de température et de direction sont deux valeurs analogiques, au sein d'une boucle FOR, nous relevons plusieurs valeurs successives avant d'en faire la moyenne et de les convertir en unités courantes.

Une imperfection est apparue dans la mesure de valeurs analogiques pour la température, il arrive que, pour une raison inconnue, les valeurs lues soient soudainement très éloignées de ce qu'elles devraient être. Pour pallier ce problème, un lissage des valeurs est nécessaire, nous restreignons l'évolution d'une mesure vis-à-vis de la précédente. Le changement de température étant un phénomène lent, nous pouvons nous le permettre.

Pour cela, nous comparons la valeur lue à la précédente, puis si cette différence est trop importante, nous limitons son changement, mais ne l'annulons pas. Ce qui permet en cas de changement brusque, ou lors de l'allumage, de prendre une centaine de mesures pour s'ajuster à sa nouvelle valeur (quelques minutes, selon les paramètres).

Pour obtenir la direction du vent, nous n'avons pas de fonction linéaire. Il est donc nécessaire de comparer la moyenne des valeurs lues à la table de la documentation pour en définir la direction.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Ici en fonction du voltage mesuré au pin analogique.

Mais la moyenne pourrait ne pas être exactement la valeur contenue dans le tableau, c'est pourquoi nous balayons le tableau afin de trouver la direction ayant la

valeur la plus proche que celle que nous avons mesuré.

```
int trouverDirectionProche(int analogiqueGirouette)
{
  int p, rtr=-1, difference_min = 1024;
  for(p=0;p<16;p++)
  {
    //on compare la valeur lue avec celle du tableau et si est est plus petite que
    //celle enregistrée alors c'est la nouvelle direction
    if( abs(analogiqueGirouette - direction_analog_in[p])<difference_min)
    {
      difference_min = abs(analogiqueGirouette - direction_analog_in[p]);
      rtr = p;
    }
  }
  return rtr;
}
```

4.3.4. Utilisons le noyau Linux

Il nous reste trois étapes, toutes seront réalisées grâce au shield Draguino. En effet, nous allons par trois fois exécuter l'action avec une commande linux.

Pour prendre une photo, par exemple, nous créons un objet nommé *photo* à partir de la bibliothèque Process.

```
void prendrePhoto()
{
  logC("prendrePhoto Process Debut",0);
  Process photo;
  photo.begin("fswebcam");
  photo.addParameter(path_photo_dragino);
  photo.addParameter("-S 20");
  photo.addParameter("-r 1280x720");
  photo.run();
  while (photo.available()>0) {
    char c = photo.read();
    Console.print(c);
  }
  logC("prendrePhoto Process Fin",0);
}
```

Cet objet contiendra tous les éléments d'une commande Linux :

- fswebcam est le programme installé sur le linux qui gère l'usage de camera
- path_photo_dragino est l'emplacement où sera stockée la photo prise
- -S 20 : laisse passer 20 frames avant de prendre la photo, ceci dans le but de laisser du temps au capteur de s'acclimater à la lumière

- -r 1280x720 : défini la résolution de l'image prise, maximale

La boucle While qui suit permet d'afficher les éventuels retours issus de l'exécution de la commande.

Le programme suit le même modèle pour envoyer les données et la photo sur le site web.

```
140 void envoiCSV()
141 {
142   logC("envoiLiDonneesCSV Process Debut",0);
143   Process envoi;
144   envoi.begin("curl");
145   String adresse = path_donnes_web_csv2;
146   adresse = String(adresse + "?ligne=");
147   //Console.println(adresse);
148   adresse += ecrireTimeDonneesCSV();
149   Console.println(adresse);
150   envoi.addParameter(adresse);
151   adresse = "";
152   envoi.run();
153   logC("envoiLiDonneesCSV Process Fin",0);
154 }
```

```
123 void envoiPhoto()
124 {
125   logC("envoiPhoto Process Debut",0);
126   Process envoi;
127   envoi.begin("curl");
128   envoi.addParameter("-T");
129   envoi.addParameter(path_photo_dragino);
130   envoi.addParameter(path_login_photo_web);
131   envoi.run();
132   while (envoi.available()>0) {
133     char c = envoi.read();
134     Console.print(c);
135   }
136   logC("envoiPhoto Process Fin",0);
137 }
```

Pour envoyer la photo, nous nous connectons via le protocole FTP, utilisé par la commande *curl* suivi du paramètre –T, indiquant le transfert de fichier, et les deux chemins, de la photo et de sa destination. Les identifiants de connexion sont mis dans l'adresse de destination. Pour rappel :

```
20 char path_login_photo_web[] = "ftp://stationm:utbmcsm2015@station-meteo-csm.nhvvs.fr/httpdocs/";
```

Communiquer les données au site web, est plus complexe. Depuis le linux, nous lui demandons de charger une page internet spécifique. Nous complétons l'adresse avec des données à transmettre. Les données sont regroupées (concaténées) en une seule chaine de caractères et séparées par un délimiteur. En effet, nous sommes limités à l'envoi d'une seule variable en paramètre.
L'adresse chargée est de par exemple :
http://station-meteo-csm.nhvvs.fr/putdata.php?ligne=170119045911,-12.89,0.00,270.00
Nous y avons mis successivement, l'instant à la prise de mesures, la température, la vitesse du vent et enfin sa direction.

En complément, il y a également diverses fonctions de type utilitaires. Nous avons une fonction qui centralise et gère la construction du journal (logC), où ici l'on affiche sur la console les informations qui lui sont données.

```
296 void logC(String txt, int nb)
297 {
298   Console.print(millis()/1000.0);
299   Console.print(" >=> ");
300   Console.print(txt);
301   Console.print(" :: ");
302   Console.println(nb);
303 }
```

```
273 String ecrireTimeComplet()
274 {
275   String rtr = "";
276   int i;
277   for(i=0;i<6;i++)
278   {
279     rtr = rtr+ecrireNombre(date_heure[i]);
280   }
281   return rtr;
282 }
```

Ou encore une fonction qui encode le temps de manière unique et sous la forme d'une chaine de 14 caractères. Mais aussi une fonction similaire, qui retourne un objet String (presque comme une chaine de caractères) avec les différentes informations à transmettre, dont l'instant présent. Le résultat de ces deux fonctions est utilisé pour l'envoi des données au site.

```
284 String ecrireTimeDonneesCSV()
285 {
286   String rtr = ecrireTimeComplet();
287   int i;
288   for(i=0;i<3;i++)
289   {
290     rtr = rtr+","+donnees[i];
291   }
292   //rtr = rtr + ";";
293   return rtr;
294 }
```

```
262 String ecrireNombre(int in) //éc
263 {
264   String rtr = "";
265   if(in<10)
266   {
267     rtr = "0";
268   }
269   rtr = rtr+in;
270   return rtr;
271 }
```

Une fonction supplémentaire, ajoute juste un zéro devant les nombres inférieurs à 10, cela nous permet de maintenir un nombre de caractères constant dans l'affichage de la date et de l'heure. Date et heures, qui sont mis à jour avec cette fonction ci-dessous, qui interroge le linux de l'heure avec la commande date, et qui analyse le retour pour en déduire les différents éléménts, de l'année à la seconde.

```
156 void majTime() //mise à jours des infos contenues dans
157 {
158   logC("majTime Process Debut",0);
159   Process pTime;
160   pTime.begin("date");
161   pTime.addParameter("+%T-%D");
162   pTime.run();
163   while(pTime.available() > 0)
164   {
165     String getText = pTime.readString();
166     date_heure[3] = getText.substring(0,2).toInt(); //heure
167     date_heure[4] = getText.substring(3,5).toInt(); //minute
168     date_heure[5] = getText.substring(6,8).toInt(); //seconde
169     date_heure[1] = getText.substring(9,11).toInt(); //mois
170     date_heure[2] = getText.substring(12,14).toInt(); //jour
171     date_heure[0] = getText.substring(15,17).toInt(); //annee
172   }
173   logC("majTime Process Fin",0);
174 }
```

4.4. Site Web

Un site a été créé pour afficher les mesures prises par la station météo, il est accessible à cette adresse : station-meteo-csm.nhvvs.fr

4.4.1. Bases de connaissance

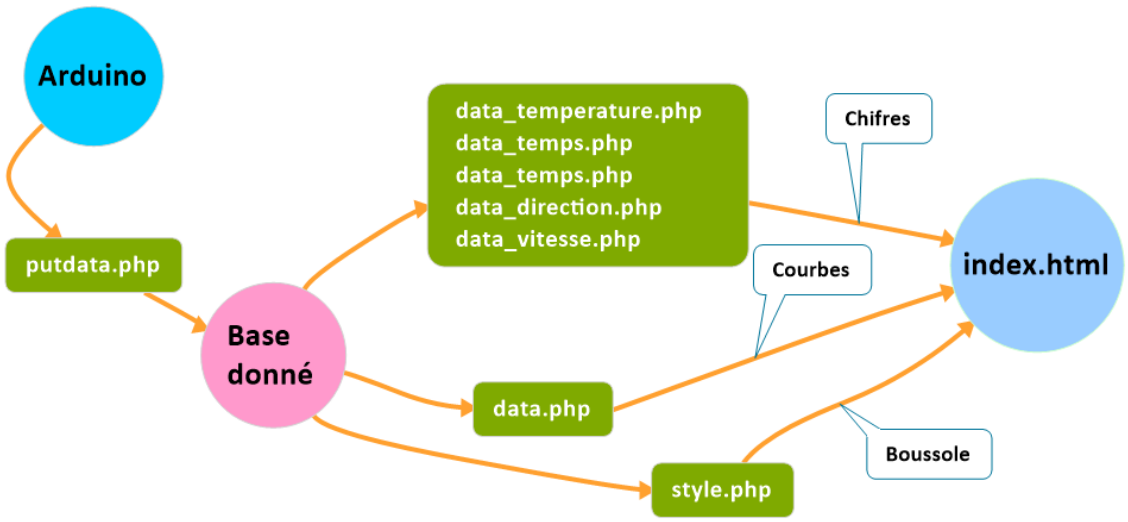
Mots clés :

- **Linux(Ubuntu)** : Paramétrer le VPS pour construire le site et contrôler des risques de sécurité. Action de base.
- **Nginx** : un logiciel libre de serveur Web (ou HTTP) ainsi qu'un proxy inverse, utilise les changements d'état pour gérer plusieurs connexions en même temps. Le traitement de chaque requête est découpé en de nombreuses mini-tâches et permet ainsi de réaliser un multiplexage efficace entre les connexions.
- **MySQL/MariaDB(Database)** : MySQL est un système de gestion de base de données relationnelles. MariaDB est un fork communautaire de MySQL. On

utilise MariaDB pour sauvegarder éternellement des données, comme vitesse du vent, température, direction du vent.

- **PHP**: *Hypertext Preprocessor,* un langage de programmation libre7, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP6, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale.
- **FTP**: File Transfer Protocol (protocole de transfert de fichier), ou FTP, est un protocole de communication destiné au partage de fichiers sur un réseau TCP/IP.
- **Highcharts**: Une librairie javascript pour les graphs.
- **Javascript**: un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs
- **Html+css**: (HyperText Markup Language) est la dernière révision majeure du HTML (format de données conçu pour représenter les pages web). Css sont des feuilles de style en cascade, généralement appelées **CSS** de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Pour afficher le web page.
- **JSON**: un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple.

4.4.2. Le fonctionnement global



Le fonctionnement ensuite s'effectue via 8 fichiers

1. putdata.php : récupérer des données sous la forme correspondante et puis les enregistrer dans la base de données. Cette action est le processus ②.
2. Tous ensemble du process ③ nous permets d'enregistrer des données dans notre espace d'Internet.
3. data.php : consulter tous les chiffres nécessaires dans la base de données et puis l'action ⑤ est de transformer sous la forme de JSON pour l'affichage par HIGHCHARTS plus tard.

   data_direction.php: consulter seulement la direction du vent dans la base de données

   data_temperature.php: consulter seulement la température dans la base de données

   data_temps.php: consulter seulement le temps d'enregistrement dans la base de données

   data_vitesse.php: consulter seulement la vitesse du vent dans la base de données

   Le process ④ est l'action de consulter via PHP et Mysql
4. En fin, le process ⑥, quand nous allons au index.html, la page web principale nous montres des chiffres et des graphes en utilisant les règles qui sont définies dans style.php.

4.4.3. Initialisation de l'environnement

Paramétrer le service de MYSQL

Les données seront stockées dans une base de données, c'est le support idouane pour la gestion de grand nombre de données régulières.

```
mysql -u root -p // se connecter par le compte d'administrateur

    CREATE DATABASE meteo //créer une base de donné meteo
    CREATE USER 'meteo'@'localhost' IDENTIFIED BY 'meteo';
    GRANT ALL ON meteo.* TO 'meteo'@'localhost'; //authentifier des modifications du compte
    flush privileges; //utiliser les paramètres après avoir créé un compte nouvel
sudo /etc/init.d/mysql restart
```

putdata.php

```
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo"); //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$morceaux = explode(",", $_GET['ligne']); //récupérer le chiffre entré dans un tableau selon le semble <<,>>
$time = $morceaux[0];
//mettre le premier morceau, le temps dans la variable time
$tem = $morceaux[1];
//mettre le deuxième morceau, le temps dans la variable tem
$vit = $morceaux[2];
//mettre le troisième morceau, le temps dans la variable vit
$dire = $morceaux[3];
//mettre le quarantième morceau, le temps dans la variable dire
$annee = substr($time, 0,2);
//mettre le morceau de 0 à 1 de la variable time dans une autre variable annee
$mois = substr($time, 2,2);
//mettre le morceau de 2 à 3 de la variable time dans une autre variable mois
$jour = substr($time, 4,2);
//mettre le morceau de 4 à 5 de la variable time dans une autre variable jour
$heure = substr($time, 6,2);
//mettre le morceau de 6 à 7 de la variable time dans une autre variable heure
$minute = substr($time, 8,2);
//mettre le morceau de 8 à 9 de la variable time dans une autre variable minute
$seconde = substr($time, 10,2);
//mettre le morceau de 10 à 11 de la variable time dans une autre variable seconde
$timec='20'.$annee.'-'.$mois.'-'.$jour.' '.$heure.':'.$minute.':'.$seconde;
//réformer l'affichage du temps
echo $timec.'<br/>'; //afficher le temps
$sql = 'INSERT INTO `meteos`(`id`, `vitesse`, `direction`, `temperature`, `created_at`, `updated_at`) VALUES (NULL, '.$vit.','.$dire.','.$tem.',\''.$timec.'\',NOW() )';
echo $sql.'<br/>'; //enregistrer dans la base de donné
mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error()); //assurer qu'il n'y a pas d'erreur
mysql_close(); //quitter le processus
?>
```

data_direction.php

```php
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$sth = mysql_query("SELECT direction FROM `meteos` ORDER BY `created_at` DESC LIMIT 1");
//sélectionner la colonne du temps dans la base de donné meteo
$row = mysql_fetch_array($sth);
//enregistrer dans une variable row
echo "<font size=\"10rem\" >La direction du vent: </font>";
echo "<font size=\"10rem\" >".$row['direction']."</font>";
//affricher la donné avec des paramétres de HTML
mysql_close($con); //quitter le processus
?>
```

data_temps.php

```php
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$sth = mysql_query("SELECT created_at FROM `meteos` ORDER BY `created_at` DESC LIMIT 1");
//sélectionner la dernière chiffre du temps dans la base de donné meteo
$row = mysql_fetch_array($sth);
//enregistrer dans une variable row
echo "<font size=\"10rem\" >Le temps: </font>";
echo "<font size=\"10rem\" >".$row['created_at']."</font>";
//affricher la donné avec des paramétres de HTML
mysql_close($con); //quitter le processus
?>
```

data_vitesse.php

```php
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$sth = mysql_query("SELECT vitesse FROM `meteos` ORDER BY `created_at` DESC LIMIT 1");
//sélectionner la dernière chiffre de la vitesse dans la base de donné meteo
$row = mysql_fetch_array($sth);
//enregistrer dans une variable row
echo "<font size=\"10rem\" >La vitesse: </font>";
echo "<font size=\"10rem\" >".$row['vitesse']."</font>";
mysql_close($con); //quitter le processus
?>
```

4.7 data_température.php

```php
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$sth = mysql_query("SELECT temperature FROM `meteos` ORDER BY `created_at` DESC LIMIT 1");
//sélectionner la dernière chiffre de la temperature dans la base de donné meteo
$row = mysql_fetch_array($sth);
//enregistrer dans une variable row
echo "<font size=\"10rem\" >La température: </font>";
echo "<font size=\"10rem\" >".$row['temperature']."</font>";
//affricher la donné avec des paramétres de HTML
mysql_close($con); //quitter le processus
?>
```

data.php

```php
<?php
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$m=mysql_query("SELECT id FROM meteos ORDER BY `id` DESC LIMIT 1;");
//sélectionner la dernière ligne dans la base de donné meteo
$n=mysql_fetch_assoc($m);
//enregistrer dans une variable n
$c=$n['id'];
$a=$c-52;
$b=$c-2;
$sth = mysql_query("SELECT created_at FROM meteos ORDER BY `created_at` ASC LIMIT  {$a}, {$b}");
//sélectionner la dernière chiffre du temps dans la base de donné meteo
$rows = array();
$rows['name'] = 'Created_at';
while($r = mysql_fetch_assoc($sth)) {
    $rows['data'][] = $r['created_at'];
}
//réformer dans un groupe de donné
$sth = mysql_query("SELECT vitesse FROM meteos ORDER BY `created_at` ASC LIMIT  {$a}, {$b}");
//sélectionner la dernière chiffre de la vitesse dans la base de donné meteo
$rows1 = array();
$rows1['name'] = 'Vitesse (m/s)';
while($rr = mysql_fetch_array($sth)) {
    $rows1['data'][] = $rr['vitesse'];
}
$sth = mysql_query("SELECT direction FROM meteos ORDER BY `created_at` ASC LIMIT  {$a}, {$b}");
//sélectionner la dernière chiffre de la direction dans la base de donné meteo
```

```php
$rows2 = array();
$rows2['name'] = 'Direction (Degree)';
while($rrr = mysql_fetch_assoc($sth)) {
    $rows2['data'][] = $rrr['direction'];
}
//réformer dans un groupe de donné
$sth = mysql_query("SELECT temperature FROM meteos ORDER BY `created_at` ASC LIMIT  {$a}, {$b}");
$rows3 = array();
$rows3['name'] = 'Temperature (Celsius degree)';
while($rrrr = mysql_fetch_assoc($sth)) {
    $rows3['data'][] = $rrrr['temperature'];
}
    $result = array();
    //réformer dans un groupe de donné
array_push($result,$rows);
array_push($result,$rows1);
array_push($result,$rows2);
array_push($result,$rows3);
//réformer tous ensemble dans un tableau
echo json_encode($result,JSON_NUMERIC_CHECK);
//réformer le tableau sous la forme de Json
mysql_close($con); //quitter le processus
?>
```

style.php

```php
<?php
    header("Content-type: text/css; charset: UTF-8");
$con = mysql_connect("localhost","meteoutbm2016","meteo");    //connecter Mysql via php
if (!$con) {
  die('Could not connect: ' . mysql_error());
}
mysql_select_db("meteo", $con); //sélectionner la base de donné meteo
$sth = mysql_query("SELECT direction FROM `meteos` ORDER BY `created_at` DESC LIMIT 1");
//sélectionner la dernière chiffre de la direction dans la base de donné meteo
$row = mysql_fetch_array($sth);
//enregistrer dans une variable row
mysql_close($con); //quitter le processus
?>
```

```css
.img {
    display: block;
    width: 33%;
    height: auto;
    margin: 0 auto;
}
//la forme d'affichage l'image
# grapha
{
height:5.0rem;
margin: auto;
position: relative;
top: 0; left: 0;
}
//la forme d'affichage le graphe 1
# graphb
{
height:5.0rem;
margin: auto;
position: relative;
top: 0; left: 0;
}
//la forme d'affichage le graphe 2
@import url(http://fonts.googleapis.com/css?family=Dosis:200,400,500,600);
.compass {
  display: block;
  width: 5rem;
  height: 5rem;
  border-radius: 100%;
  box-shadow: 0 0 0.25rem rgba(0, 0, 0, 0.85);
  position: relative;
  font-family: 'Dosis';
  color: #555;
  text-shadow: 1px 1px 1px white;
}
```

```css
.compass .direction p {
  text-align: center;
  margin: 0;
  padding: 0;
  position: absolute;
  top: 50%;
  left: 0;
  width: 100%;
  height: 100%;
  line-height: 80px;
  display: block;
  margin-top: -45px;
  font-size: 0.6rem;
  font-weight: bold;
}
.compass .direction p span {
  display: block;
  line-height: normal;
  margin-top: -24px;
  font-size: 0.4rem;
  text-transform: uppercase;
  font-weight: normal;
}
.compass .arrow {
  width: 100%;
  height: 100%;
  display: block;
  position: absolute;
  top: 0;
}
.compass .arrow:after {
  content: "";
  width: 0;
  height: 0;
  border-left: 5px solid transparent;
  border-right: 5px solid transparent;
  border-bottom: 1.5rem solid red;
  position: absolute;
  top: -6px;
```

```css
  left: 50%;
  margin-left: -5px;
  z-index: 99;
}
.compass .arrow.ne {
  transform: rotate(<?php echo $row['direction'];?>deg);
}
//la forme d'affichage le compass pour la représentation de la direction du vent
```

index.html

```
<!DOCTYPE HTML>
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    <meta name="description" content="meteo" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        //initialisation de HTML
        <meta http-equiv="refresh" content="40000">
        //rafraichir le page après 40 secondes
        <title>Meteo</title>
        <link rel="stylesheet" type="text/css" href="style.php" />
        <script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>
        <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
        <script type="text/javascript">
        var cssEl = document.createElement('style');
        document.documentElement.firstElementChild.appendChild(cssEl);
        function setPxPerRem(){
            var dpr = 1;
            var pxPerRem = document.documentElement.clientWidth * dpr / 10;
            cssEl.innerHTML = 'html{font-size:' + pxPerRem + 'px!important;}';
        }
        setPxPerRem();
    </script> //définir la taille de mot
<div style="text-align:center;background-color:#0000;width:auto;height:1.5rem;overflow:hidden;margin:0 auto;">Meteo</div>
//afficher le titre sur le page
<div style="text-align:center;background-color:#BFBFBF;width:auto;height:1.5rem;overflow:hidden;margin:0 auto;">
<iframe src="datati.php" style="position:relative;top:0.3rem;font-weight:bold;" frameborder="no" height="100rem" width="800rem" ></iframe>
</div>//afficher le temps d'enregistrement de la dernier donné sur le page
<div style="text-align:center;background-color:#B0B0BF;width:auto;height:1.5rem;overflow:hidden;margin:0 auto;">
<iframe src="datat.php" style="position:relative;top:0.3rem;font-weight:bold;" frameborder="no" height="100rem" width="800rem" ></iframe>
</div>//afficher la température d'enregistrement de la dernier donné sur le page
<div style="text-align:center;background-color:#86E2D5;width:auto;height:1.5rem;overflow:hidden;margin:0 auto;">
<iframe src="datav.php" style="position:relative;top:0.3rem;font-weight:bold;" frameborder="no" height="100rem" width="800rem" ></iframe>
<br />
        <div class="compass">
        <div class="direction">
        <p>Direction du Vent<span> </span></p>
        </div>
        <div class="arrow ne"></div>
        </div>//afficher la compass pour indiquer la direction du vent
<div style="text-align:center;background-color:#0000;width:auto;height:7rem;overflow:hidden;margin:0 auto;">
<img src="meteoimage/meteo.png" alt="Camera" style="position:relative;top:1rem;" height="600rem" width="auto"></img>
</div>// afficher le dernier photo capturé par le caméra
<br />
```

Le graphe A a but de montrer la variation de la vitesse du vent et en même temps le graphe B a but de montrer la variation de la température.

```
        $.getJSON("data.php", function(json) {
            optionsa.xAxis.categories = json[0]['data'];
            optionsb.xAxis.categories = json[0]['data'];
            optionsa.series[0] = json[1];
            //options.series[1] = json[2];
            optionsb.series[0] = json[3];
            chart1 = new Highcharts.Chart(optionsa);
            chart2 = new Highcharts.Chart(optionsb);
        });
```

Les codes nous permettent de lire des données dans data.php et puis les affiche.

```
474 <div style="font-size: 0.5rem; text-align:center;background-color:#0000;width:auto;margin:0 auto;"> Vitesse du vent</div>
475 <div id="grapha" style="min-width: 10.0rem; height: 6.0rem; margin: 0 auto"></div>
476 <br />
477 <div style="font-size: 0.5rem; text-align:center;background-color:#0000;width:auto;margin:0 auto;">Temperature</div>
478 <div id="graphb" style="min-width: 10.0rem; height: 6.0rem; margin: 0 auto"></div>
479 //afficher deux tableaux pour indiquer la variation de température et la vitesse du vent
```

4.5. Boitiers

Quatre parties de la station doivent être protégées :

- La batterie et le régulateur de tension
- Le capteur de température
- La caméra (webcam) qui prend les photos
- Le boitier Arduino relié au Dragino Yun et au boitier 3G

Nous décidons donc de créer des boitiers afin d'assurer la protection de ces éléments. La batterie, trop lourde pour être fixée au mât, restera dans une boîte étanche au sol. Les trois autres peuvent être protégés par des boitiers imprimés en 3D.

4.5.1. Boitier de la batterie et du régulateur de tension

Ce boitier reste au sol et doit protéger la batterie et le régulateur de tension des intempéries. Il doit remplir le cahier des charges suivant :

Fonction
• Contenir la batterie et le régulateur de tension
• Protéger son contenu de la pluie
• Avoir des trous pour faire entrer les câbles
• Résister au poids de la batterie
• Etre opaque pour éviter que l'on voie son contenu

Figure 1: Cahier des charges boitier de la batterie et du régulateur de tension

Nous sommes allés à Leroy Merlin chercher une boite respectant ces critères. Le seul contenant vraiment étanche a été un sceau en plastique refermable avec un

couvercle. Ses dimensions sont assez grandes pour accueillir les composants mais il était de couleur transparent.

Nous l'avons donc peint en blanc, après l'avoir poncé pour permettre à la peinture de mieux accrocher.

Nous obtenons au final le résultat suivant :

4.5.2. Boitier de l'Arduino, du Dragino Yun et du boitier 3G

Ce second boitier respecte le cahier des charges si dessous :

Fonction
• Contenir l'Arduino, le Dragino Yun et le boitier 3G
• Protéger l'Arduino, le Dragino Yun et le boitier 3G de la pluie
• Avoir des trous pour faire entrer les câbles
• Pouvoir se fixer au mât
• Etre opaque pour éviter que l'on voie son contenu

Figure 2: Cahier des charges boitier de l'Arduino, du Dragino et du boitier 3G

Nous comptions à l'origine réaliser ce boitier en impression 3D, néanmoins, il a été mis en évidence que notre idée n'était pas adaptée à la fabrication additive vu ses dimensions. Trop de fonctions que réalisait notre concept pouvaient être faites d'une autre manière, plus adaptées ou moins onéreuse. La conception restait très standard, sans profiter des nouvelles possibilités de design rendues possibles par la fabrication additive, telles que les formes courbes ou les objets autres que rectilignes ou circulaires.

Nous nous sommes alors penchés sur un moyen plus modulaire de réaliser ce boitier. C'est finalement avec un assemblage de tuyaux de PVC que nous avons réussi à protéger notre partie commande

Un tuyau PVC, dévissable à ses extrémités, permet de protéger le contenu.

Les câbles sortent par un tuyau coudé en U permettant également l'aération des composants tout en empêchant la pluie de rentrer.

Figure 3: Boitier en PVC contenant l'Arduino, le Dragino et le boitier 3G

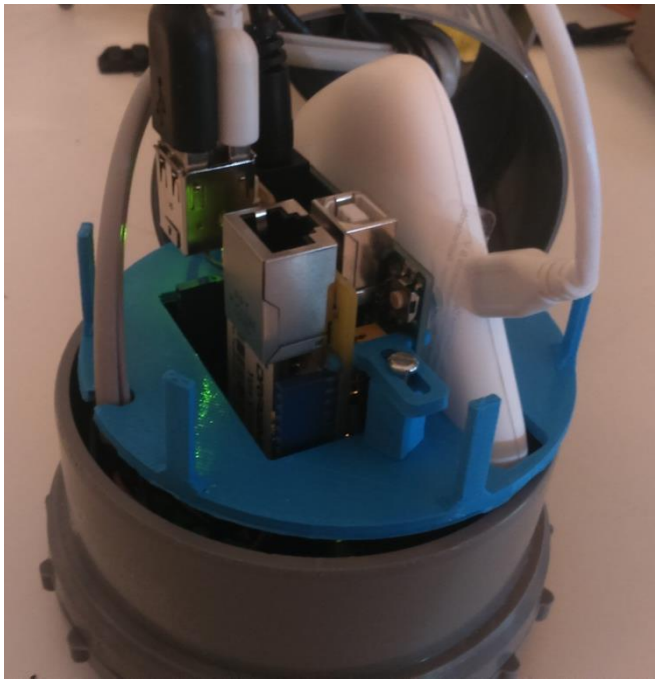


Figure 4: Support de l'Arduino, du Dragino et du boitier 3G

Ces derniers sont maintenus en place à l'intérieur par un support imprimé en 3D (voir figure ci-dessus).

4.5.3. Boitier de la sonde température

Le boitier abritant la sonde de température respecte le cahier des charges si dessous :

Fonction	Critère
• Contenir la sonde de température	
• Protéger la sonde de température de la pluie	
• Ne pas isoler thermiquement la sonde de température de l'extérieur	Température interne égale à celle externe
• Pouvoir se fixer au mât	
• Etre opaque pour éviter que l'on voie son contenu	
• Isoler la sonde de température de la partie commande	
• Etre fabricable en impression 3D	

Figure 5: Cahier des charges boitier de la sonde de température

Dû aux échauffements de la partie commande, il est impossible de placer le capteur de température à proximité de cette dernière. Il est donc décidé, de lui fournir un boitier dédié, de petite dimension, pouvant accueillir ce capteur de température TMP36 d'une longueur de 20mm. Il sera fixé au mat vertical, un tube de 20 mm de diamètre.

Il lui est nécessaire d'être protégé des conditions climatiques mais doit toujours correspondre aux températures extérieures. Le capteur en lui-même est étudié pour ne provoquer un échauffement interne minime.
Nous créons donc un boitier en impression 3D possédant un anneau de fixation au mat, avec une possibilité de serrage par une vis ; et d'un autre côté, une cavité s'ouvrant vers le bas qui contiendra le capteur, et qui est fermée par un capuchon serré.

Nous obtenons le boitier présenté ci-dessous (retournée pour plus de visibilité).

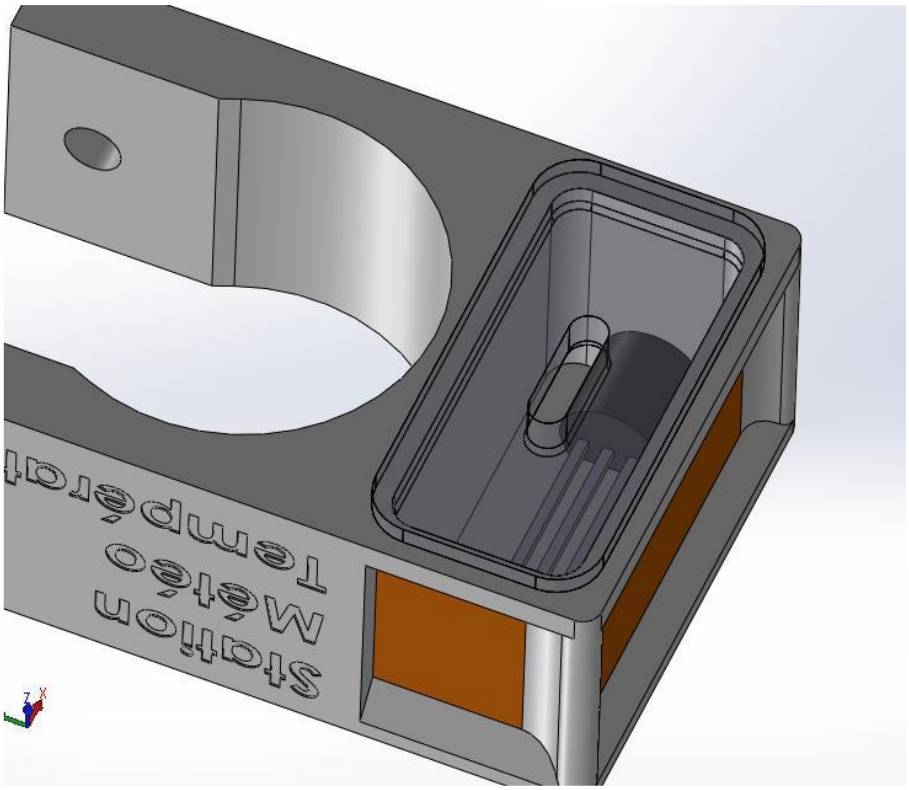


Figure 6: Modèle 3D du boitier de température

Nous remarquons le capteur collé à la surface supérieure avec une ouverture dans le capuchon pour laisser les câbles passer. Les trois surfaces verticales de couleur sont définies pour avoir une épaisseur de 0.5 mm, laissant ainsi les échanges thermiques se réaliser plus facilement. Nous avons confirmé par la suite que la température donnée par le capteur était bien égale à celle externe via un thermomètre.

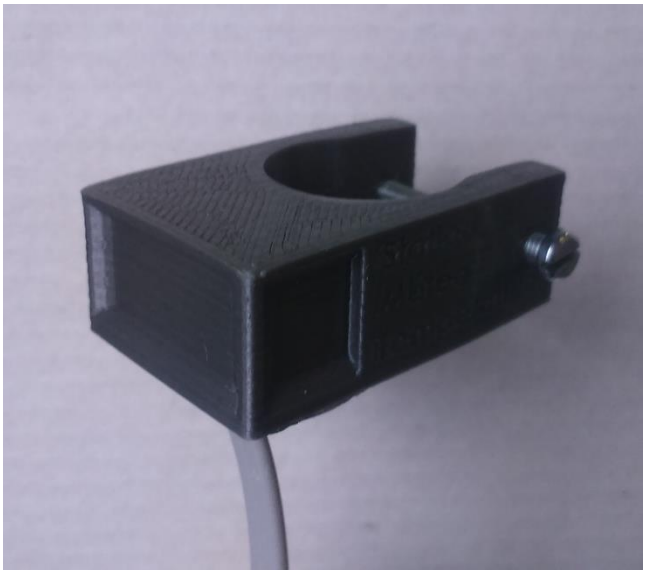


Figure 7: Boitier du capteur de température

4.5.4. Boitier de la Caméra

On commence par créer un nouveau cahier des charges, dédié à La webcam cette fois-ci :

Fonction
• Protéger de la pluie
• Pouvoir se fixer au mât
• Permettre la prise de photos
• Pouvoir attacher la caméra
• Être fabricable en impression 3D

Figure 8: Cahier des Charges du boitier de la Webcam

Plusieurs solutions s'offraient à nous pour rendre ce boitier étanche tout en permettant la prise de photos. Faire une « fenêtre » en pvc transparent était l'une d'entre elles, cependant, il y avait plusieurs inconvénients :

- La fenêtre peut se salir, détériorant la qualité de l'image. Cela reste cependant peu probable si la fenêtre est à l'abri des intempéries.
- La condensation sur la vitre peut créer de la buée et brouiller l'image. La station météo sera en service l'été, ce scénario est donc envisageable. L'enceinte du boitier ne doit donc pas être complètement clos.
- Le plus gros problème peut être la réflexion du soleil sur la surface de l'eau puis sur la vitre ce qui saturerait encore une fois l'image.

Nous avons donc opté pour le choix de laisser le boitier ouvert devant l'objectif et en dessous de la webcam. Ce choix impose donc de « long rebords » pour ne pas que la pluie atteigne la Webcam.

4.5.4.1. Conception du modèle

Une première ébauche sous SolidWorks nous donne la forme générale de la pièce.

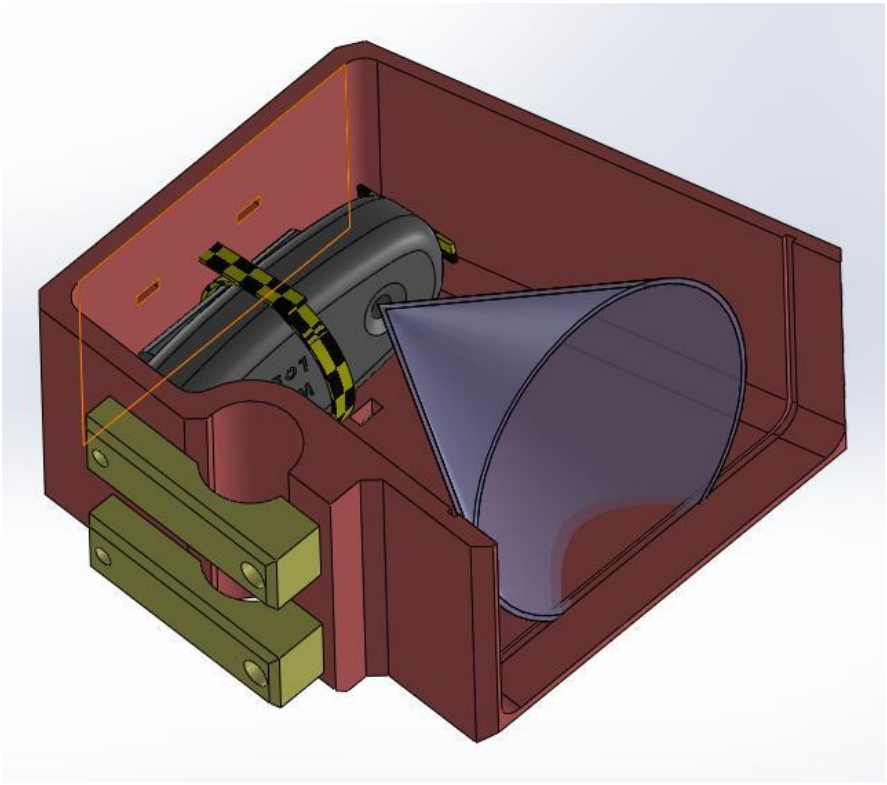


Figure 9: Première ébauche du boitier Camera

Une fois la forme générale validée nous avons refait un modèle surfacique sous Catia. Après quelques ajustements, nous obtenons le modèle suivant :

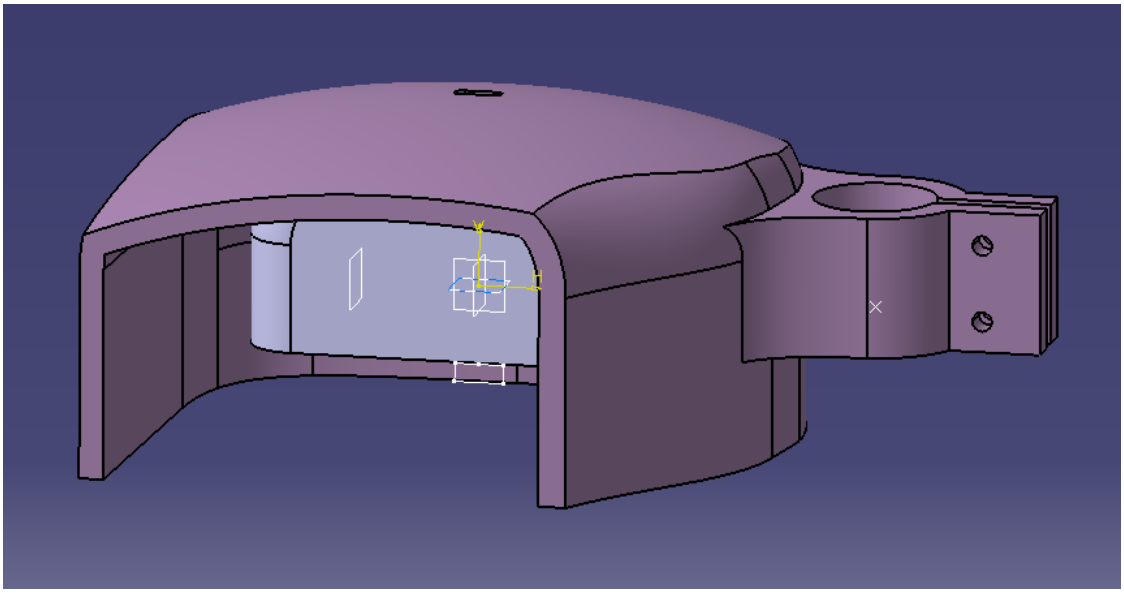


Figure 10: Boitier Webcam surfacique vue de devant

La caméra est protégée de la pluie par la large coque épaisse de 4 mm et son ouverture lui permet de prendre des photos de la surface du lac.

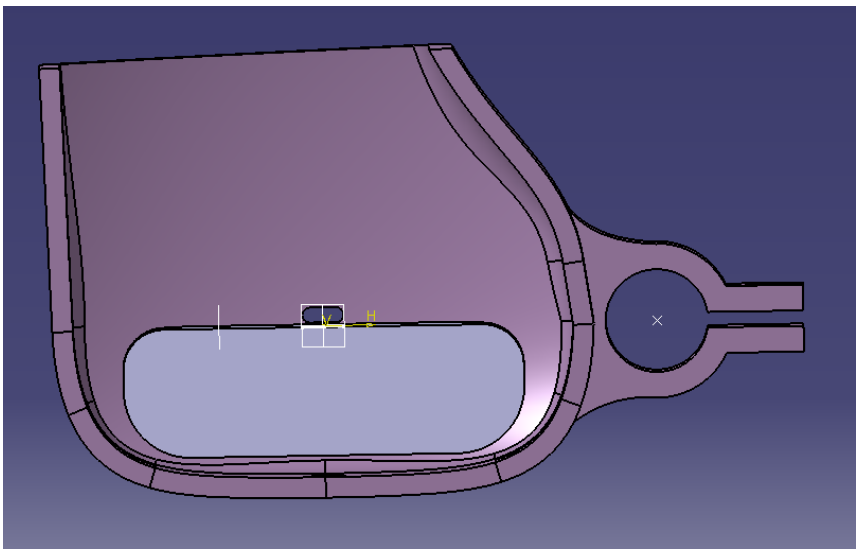


Figure 11: Boitier Webcam surfacique vue de dessous

Le serrage de la coque autour du mât se fait via deux vis qui viennent resserrer l'étau de la fixation autour de l'axe.

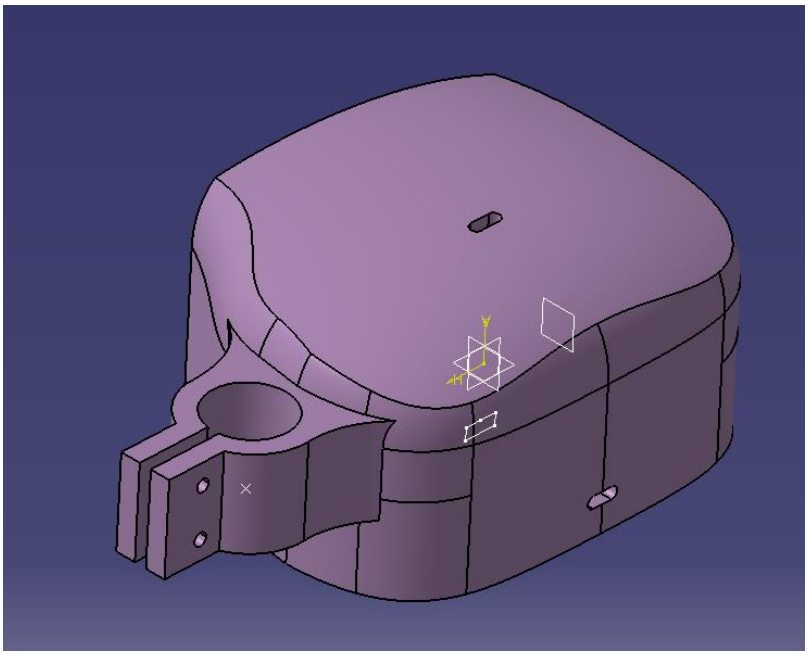


Figure 12: Boitier Webcam surfacique vue du dessus

Deux petits trous dans la coque permettent le passage d'un colson qui tiendra la caméra en place. Ces trous seront rebouchés avec du silicone pour garantir

l'étanchéité de la coque. L'avantage du colson face à une fixation intégrée est que l'on peut plus facilement ajuster l'orientation de la caméra selon les besoins futurs du client.

Ce boitier peut s'imprimer à l'envers sur la face supérieure ce qui évite les supports à l'intérieur. De plus, l'axe du mât est vertical, évitant des déformations. Les trous des vis ont un diamètre de 3,2 mm ce qui est inférieur à 5mm. Ils ne nécessitent donc pas de support non plus. Une fois la face supérieure faite, le reste de la pièce est verticale : plus aucun autre support n'est requis. Il n'y a pas d'angle trop abrupt susceptible de générer des contraintes pendant la fabrication.

Ce modèle remplis donc bien le cahier des charges énoncé plus haut.

4.5.4.2. Analyse du modèle

Avant de lancer l'impression, nous vérifions tout de même la résistance du système au poids de la caméra dans le logiciel Inspire.

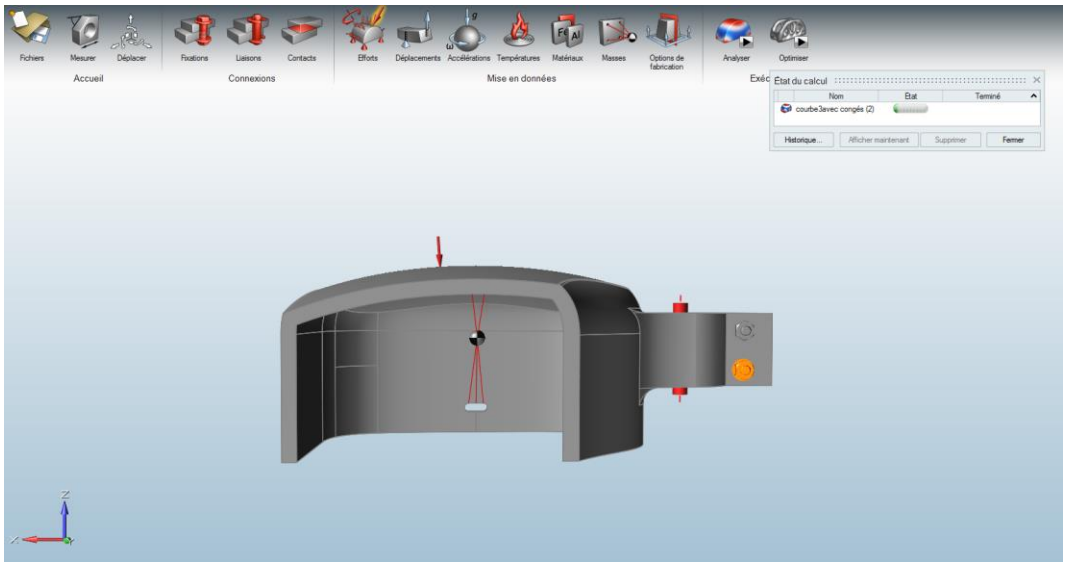


Figure 13: Paramétrage du modèle sous Inspire

On place deux vis-écrous dans la fixation au mât et on simule ce dernier par un support cylindrique. Une masse ponctuelle de 30g (avec marge de sécurité) est placée à la place de la caméra.

Après analyse, nous trouvons que le maintien de la caméra seul génère un déplacement maximal en extrémité de 39µm. Cette valeur est parfaitement satisfaisante.

Ce dernier boitier offrant une surface beaucoup plus large, il est possible que de petits oiseaux s'y posent. Nous effectuons par conséquent une seconde analyse en appliquant une force supplémentaire sur la face supérieure pour simuler le poids éventuel d'un moineau (40g). Cette charge supplémentaire n'induit qu'un déplacement maximum de 56 µm. Une distance encore négligeable à l'échelle du boitier. Nous sommes à 0,5% de la limite élastique et le facteur de sécurité est en tout point largement supérieur à $6.0.10^6$.

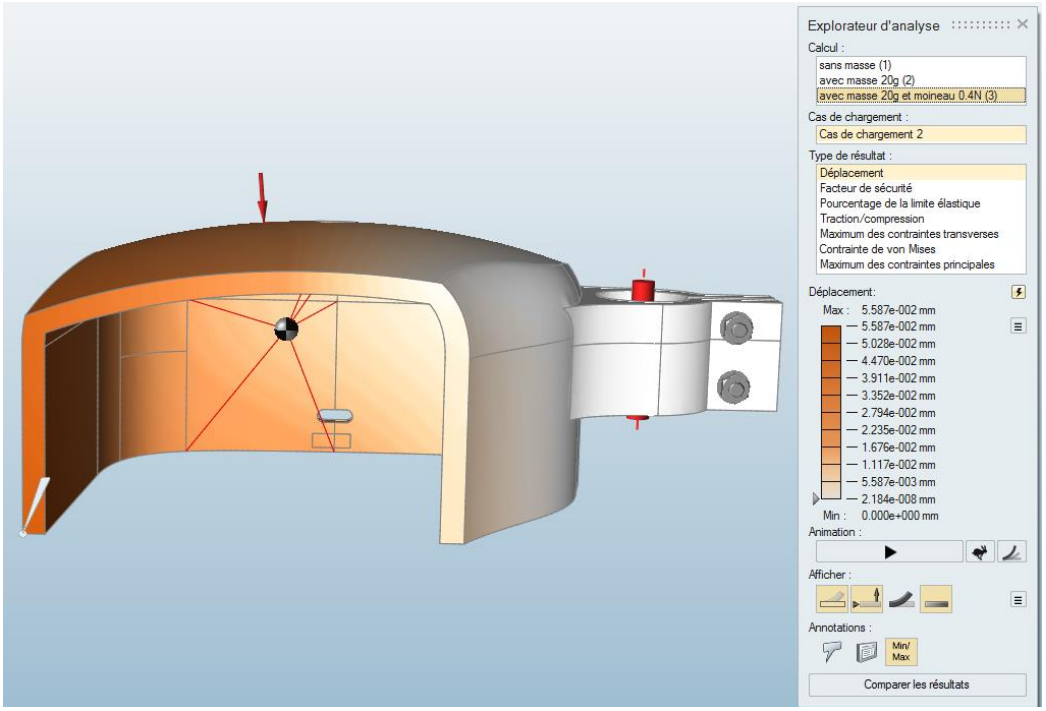


Figure 14: analyse du déplacement du boitier caméra chargé

On remarque donc que l'on résiste très largement à la charge. Il serait donc possible de supprimer de la matière du boitier. Nous laissons néanmoins les dimensions et l'épaisseur telles quelles car ce boitier sera installé à l'extérieur. Il lui faudra résister aux intempéries, à l'érosion et aux éventuels chocs. De plus, nous avons constaté que les pièces imprimées peuvent se déformer sous l'action de la chaleur en été si elles ne sont pas suffisamment épaisses.

Figure 15: Analyse du boitier caméra chargé avec la camera et le moineaux

4.5.4.3. Impression du boitier

On imprime la pièce en PLA sur une machine ULTIMAKER².

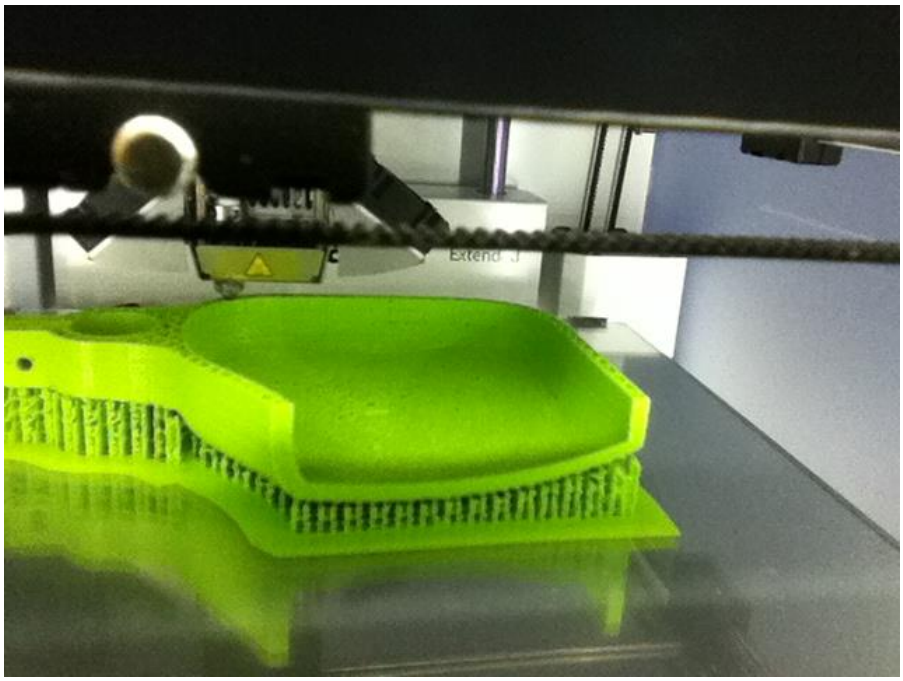


Figure 16: Impression de la pièce

L'impre uts. On remarque en effet plusieurs lacunes entre certaines couches et sur la face supérieure du boitier.

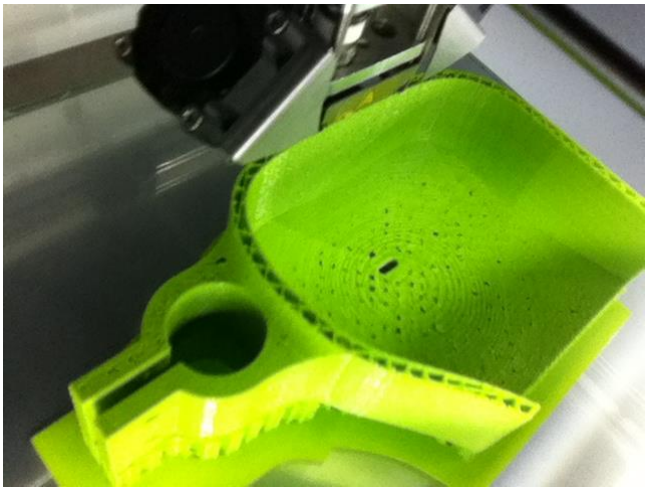


Figure 18: Défauts d'impression

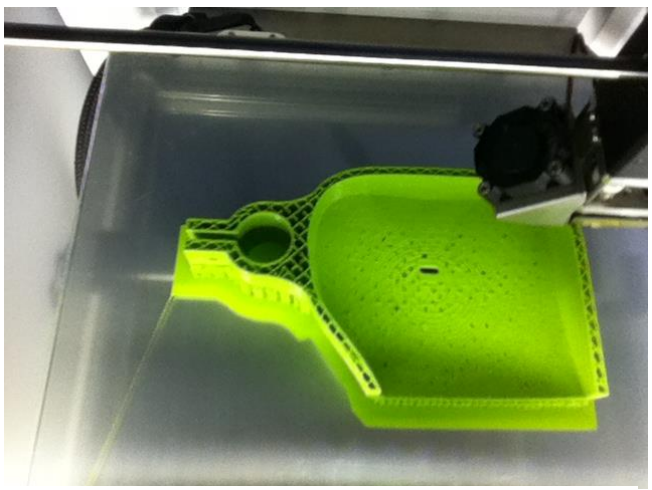


Figure 17: Défauts d'impression 2

Nous pensons que cela est dû aux vibrations de la plaque chauffante. Nous avons en effet constaté que celle-ci vibrait beaucoup et que les fils étaient parfois déposés en ondulant verticalement.

Figure 19: Boitier camera brut vue de dessous

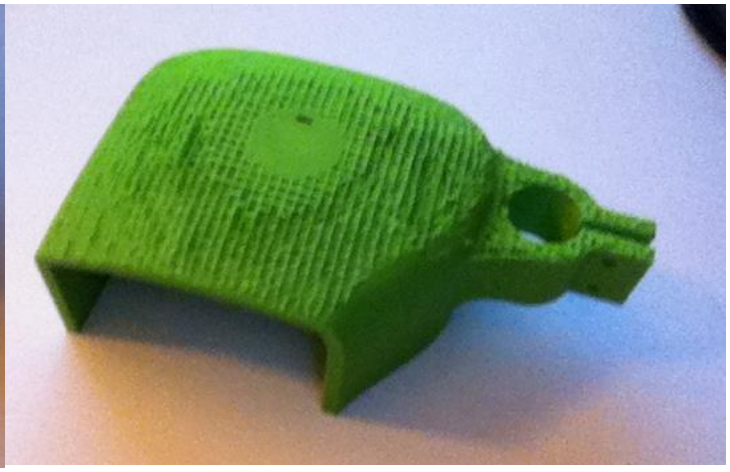


Figure 20: Boitier camera brut vue de dessus

La pièce finale obtenue reste globalement fonctionnelle. Seule l'étanchéité est en défaut. Pour y remédier, nous ponçons et comblons les trous en surface avec du silicone et passons quelques couches de peinture.

5. Produit fini

Nous obtenons au final le produit fini ci-dessous:

Figure 21: Assemblage final

On retrouve tout en haut du mât la girouette et l'anémomètre, puis viens le boitier de la sonde de température puis le boitier de la webcam en dessous.

Ces instruments sont reliés au boitier contenant la partie commande dans le tuyau en PVC. Un des câbles sortant de ce-dernier se connecte à un second câble venant de la batterie dans le sceau.

Cette batterie est rechargée par la tension régulée issue d'un panneau solaire rendant ainsi la station auto-suffisante en énergie.

Le mât se scinde en deux parties. La partie inférieure sera fixée sur place sur le toit d'un bâtiment sur la rive du bassin. La partie supérieure viendra s'emboiter dessus et une goupille maintiendra la position angulaire.

La station pourra ainsi être rapidement installée ou démontée pour l'hiver.

Nous déconseillons le stockage de l'appareil dans un lieu humide ou à températures négatives. Le boitier 3G induit la présence d'une batterie au sein de boitier de commande.

6. Conclusion

Ce projet, qui bien ayant été déjà initié par d'autres, nous a permis de nous éprouver à différentes problématiques, il nous a surtout demandé de les traiter en parallèle.

Bien que ce soit un projet de dimensions limités, il nous a été demandé de maîtriser différents domaines, tels que la conception de boîtiers, mais aussi la programmation sur Arduino et Linux, tout comme la création de pages web avec des affichages graphiques.

Il a donc permit à chacun d'entre nous de mettre en œuvre nos savoirs et compétences et d'en acquérir d'autres dans le seul but de répondre aux différentes problématiques que nous avons rencontré.

7. Annexes

Une documentation plus fournie et plus visuelle ainsi que divers téléchargements sont disponibles en ligne à l'adresse suivante : station-meteo-csm.nhvvs.fr/rapport/documentation/

7.1. Programme détaillé

Le programme est ici en noir et blanc, il est également disponible en couleur dans la documentation en ligne qui accompagne notre rapport

```
//librairies nécessaires
#include <Process.h>

//librairies pour le développement
#include <Console.h>

//définition de define (paramétrage des entrés/sorties)
#define DO_BUZ 7 //pour implantation d'un buzzer (dev)
#define DI_ANEMOMETRE_INT 2
#define AI_GIROUETTE 0
#define AI_TEMPERATURE 1
#define DUREE_LECTURE_CAPTEUR 30

#define HEURE_DEBUT 8
#define HEURE_FIN 20

//variables générales
int direction_analog_in[] =
{786,405,460,84,92,65,184,127,286,243,630,599,945,827,978,702}
;
char path_photo_dragino[] = "/mnt/sda1/meteo.png";
//photo_meteo_2
char path_login_photo_web[] =
"ftp://stationm:utbmcsm2015@station-meteo-
csm.nhvvs.fr/httpdocs/";
char path_donnes_web_csv2[] = "http://station-meteo-
csm.nhvvs.fr/putdata.php";
int date_heure[6]; //contient annee-mois-jour-heure-minute-
seconde
volatile int compteur_vent;
float donnees[3]; //contient température, vitesse du vent et
direction
```

```
int numero_boucle; //numero d'iteration de la boucle
principale
unsigned long tempo = 0;
int derniere_mesure = 0;

//déclarations fonctions
void prendrePhoto();
void envoiPhoto();
void envoiCSV();
void majTime();
void incrementationCompteurVent();
void lectureCapteurs(int duree);
int trouverDirectionProche(int analogiqueGirouette);
String ecrireNombre(int in);
String ecrireTimeComplet();
String ecrireTimeDonneesCSV();
void logC(String txt, int nb);
void cycleMesure();


// SETUP SETUP SETUP SETUP SETUP SETUP SETUP SETUP SETUP
void setup()
{
  Bridge.begin();
  Console.begin();
  //while(!Console);
  logC("Fin Bridge et Console setup", 0);

  logC("Debut setup", 0);
  numero_boucle = 0;
  tempo = 0;

  //pinModes
  pinMode(DO_BUZ, OUTPUT);
  pinMode(DI_ANEMOMETRE_INT, INPUT_PULLUP);
  attachInterrupt(0,incrementationCompteurVent, FALLING);

  tone(DO_BUZ,500,500);
  logC("Fin du setup", 0);
}

// LOOP LOOP LOOP LOOP LOOP LOOP LOOP LOOP LOOP LOOP LOOP
void loop()
{
  logC("Loop Debut numero",numero_boucle);
  majTime(); //mise à jour de l'heure et la date.

  if(constrain(date_heure[3],HEURE_DEBUT,HEURE_FIN) ==
date_heure[3])//si l'on est dans la période d'activation
  {
    //on prend les mesures
```

```
    logC("Prise de mesures", 0);
    cycleMesure();
  }
  else if(date_heure[3] == HEURE_DEBUT-1) //à moins d'une
heure
  {
    //on attend un nombre de minutes jusqu'à l'heure pile
    tempo = 60*(60-date_heure[4]+1);
    tempo = tempo * 1000;
    Console.print("Pause partielle :");
    tone(DO_BUZ,500,1500);
    delay(1500);
    tone(DO_BUZ,500,500);
    delay(500);
    Console.println(tempo);
    delay(tempo);
  }
  else
  {
    //on attend une heure
    tempo = 60*60000;
    Console.print("Pause heure :");
    tone(DO_BUZ,500,1500);
    delay(1500);
    tone(DO_BUZ,500,1500);
    delay(1500);
    Console.println(tempo);
    delay(tempo);
  }


  tone(DO_BUZ,1000,50);
  logC("Loop Fin",0);
}

void cycleMesure()
{
  lectureCapteurs(DUREE_LECTURE_CAPTEUR);
  envoiCSV();
  prendrePhoto();
  envoiPhoto();

  Console.println(ecrireTimeDonneesCSV());
}

//fonctions dragino
void prendrePhoto()
{
  logC("prendrePhoto Process Debut",0);
  Process photo;
  photo.begin("fswebcam");
```

```
  photo.addParameter(path_photo_dragino);
  photo.addParameter("-S 20");
  photo.addParameter("-r 1280x720");
  photo.run();
  while (photo.available()>0) {
    char c = photo.read();
    Console.print(c);
  }
  logC("prendrePhoto Process Fin",0);
}

void envoiPhoto()
{
  logC("envoiPhoto Process Debut",0);
  Process envoi;
  envoi.begin("curl");
  envoi.addParameter("-T");
  envoi.addParameter(path_photo_dragino);
  envoi.addParameter(path_login_photo_web);
  envoi.run();
  while (envoi.available()>0) {
    char c = envoi.read();
    Console.print(c);
  }
  logC("envoiPhoto Process Fin",0);
}


void envoiCSV()
{
  logC("envoiLiDonneesCSV Process Debut",0);
  Process envoi;
  envoi.begin("curl");
  String adresse = path_donnes_web_csv2;
  adresse = String(adresse + "?ligne=");
  //Console.println(adresse);
  adresse += ecrireTimeDonneesCSV();
  Console.println(adresse);
  envoi.addParameter(adresse);
  adresse = "";
  envoi.run();
  logC("envoiLiDonneesCSV Process Fin",0);
}

void majTime() //mise à jours des infos contenues dans
{
  logC("majTime Process Debut",0);
  Process pTime;
  pTime.begin("date");
  pTime.addParameter("+%T-%D");
  pTime.run();
```

```
  while(pTime.available() > 0)
  {
    String getText = pTime.readString();
    date_heure[3] = getText.substring(0,2).toInt(); //heure
    date_heure[4] = getText.substring(3,5).toInt(); //minute
    date_heure[5] = getText.substring(6,8).toInt(); //seconde
    date_heure[1] = getText.substring(9,11).toInt(); //mois
    date_heure[2] = getText.substring(12,14).toInt(); //jour
    date_heure[0] = getText.substring(15,17).toInt(); //annee
  }
  logC("majTime Process Fin",0);
}

//fonctions capteurs
void incrementationCompteurVent()
{
  compteur_vent = compteur_vent +1;
}

void lectureCapteurs(int duree)
{
  logC("lectureCapteurs Debut",0);
  logC("Nombre de lecture des capteurs",duree);
  int boucles =0;
  unsigned long debut_mesure_vent;
  long somme_temperature = 0, somme_girouette = 0;

  //début de la duree de prise de mesures
  debut_mesure_vent = millis();
  compteur_vent = 0;
  for(boucles=0;boucles<duree;boucles++) //si on a encore le
temps de prende une mesure
  {
    logC("Prise de mesure numero",boucles);
    //on prend la valeur de température
    int lecture = 0;
    lecture = analogRead(AI_TEMPERATURE);
    int moy_temp = somme_temperature/boucles;
    if(boucles==0) {moy_temp=derniere_mesure;}
    if(lecture > (moy_temp+10))
    {
      logC("Sur-mesure",(lecture - moy_temp));
      lecture = moy_temp+10;
    }
    else if(lecture < (moy_temp-10))
    {
      logC("Sous-mesure",(lecture - moy_temp));
      lecture = moy_temp-10;
    }

    logC("Capteur : lecture ai_temperature",lecture);
```

```
    //logC("Capteur : calcul ai_temperature",
(lecture*(5000/1024.0)));
    somme_temperature = somme_temperature + lecture;
    //on prend la valeur de girouette
    lecture = analogRead(AI_GIROUETTE);
    logC("Capteur : lecture ai_girouette",lecture);
    somme_girouette = somme_girouette + lecture;

    tone(DO_BUZ,500,5);
    delay(950); //atente de moins d'une seconde pour ne pas
prendre trop de mesures
  }
  //fin de la duree de prise de mesures
  //vitesse du vent
  donnees[1] = (millis() - debut_mesure_vent);
  donnees[1] = (compteur_vent * 2.4 * 1000) / donnees[1];
  logC("Vent, nombre de tics", compteur_vent);
  long difference = (millis() - debut_mesure_vent);
  logC("Vent, duree en ms (approx)", difference);
  logC("Vent, vitesse en km/h", donnees[1]);
  //température
  somme_temperature = somme_temperature / duree;
  derniere_mesure = somme_temperature;
  double temp_temperature = somme_temperature*(5000/1024.0);
  donnees[0] = ( temp_temperature - 500)/10.0;
  logC("Temperature lue (moyenne)",somme_temperature);
  logC("Temperature (etalonnee)", donnees[0]);
  //direction
  donnees[2] = trouverDirectionProche(somme_girouette /
duree)*22.5;
  logC("Direction (degres)", donnees[2]);

  logC("lectureCapteurs Fin",0);
}

int trouverDirectionProche(int analogiqueGirouette)
{
  int p, rtr=-1, difference_min = 1024;
  for(p=0;p<16;p++)
  {
    //on compare la valeur lue avec celle du tableau et si est
est plus petite que
    //celle enregistrée alors c'est la nouvelle direction
    if( abs(analogiqueGirouette -
direction_analog_in[p])<difference_min)
    {
      difference_min = abs(analogiqueGirouette -
direction_analog_in[p]);
      rtr = p;
    }
  }
```

```
  return rtr;
}

//fonction annexes
String ecrireNombre(int in) //écrit un nombre de deux chiffres
ou plus
{
  String rtr = "";
  if(in<10)
  {
    rtr = "0";
  }
  rtr = rtr+in;
  return rtr;
}

String ecrireTimeComplet()
{
  String rtr = "";
  int i;
  for(i=0;i<6;i++)
  {
    rtr = rtr+ecrireNombre(date_heure[i]);
  }
  return rtr;
}

String ecrireTimeDonneesCSV()
{
  String rtr = ecrireTimeComplet();
  int i;
  for(i=0;i<3;i++)
  {
    rtr = rtr+","+donnees[i];
  }
  //rtr = rtr + ";";
  return rtr;
}

void logC(String txt, int nb)
{
  Console.print(millis()/1000.0);
  Console.print(" >=> ");
  Console.print(txt);
  Console.print(" :: ");
  Console.println(nb);
}
```

7.2. Manuel d'usage

7.2.1. Connection au site web

L'adresse est station-meteo-csm.nhvvs.fr

Une connexion par mot de passe est requise, il est initialement défini mais il est toujours possible de le changer en bas de page.

Deux comptes sont créés, le compte pour invités et le compte administrateur. Les deux proposent les même informations mais ce dernier a, en plus, la possibilité de modifier les mots de passe pour les deux comptes.

En cas de perte, ou de soucis, il est possible de le réinitialiser en accédant à l'administration du site et de modifier les deux fichiers correspondants aux mots de passe.

Le graphisme est optimisé pour une lecture sur un écran de téléphone portable.

7.2.2. Installation /Désintallation

Voici les différentes étapes pour procéder à l'installation de la station météo :

- Emboiter le mât sur le socle fixé sur le toit.
- Installer le Panneau solaire sur son socle.
- Brancher le Panneau solaire aux câbles partant du régulateur de tension présent dans le sceau.
- Ouvrir le boitier de l'Arduino par le dessous et appuyer pendant 3 secondes sur le bouton du boitier 3G pour l'allumer. L'allumage se caractérise par l'apparition de lumières vertes. Si le boitier 3G ne s'allume pas, il est alors nécessaire de le charger en alimentant la carte Arduino, pour ensuite appuyer sur le bouton.

- Brancher le câble d'alimentation sortant du boitier contenant l'Arduino (seul câble non relié) à la prise sortant du sceau pour mettre l'Arduino sous tension.
- La station devrait à présent être opérationnelle. Si les données n'arrivent pas, il est possible que le Dragino ait échoué la connexion au boitier 3G. Dans ce cas, ouvrez et refermez la prise étanche pour relancer le lancement.
- Refermer le boitier de la Station Arduino

Concernant la désinstallation, les différentes étapes sont :

- Débrancher les câbles du panneau solaire ainsi que le câble d'alimentation de la station.
- Décrocher le panneau solaire.
- Retourner puis ouvrir le boitier de l'Arduino par le dessous et appuyer longtemps sur le bouton du boitier 3G pour l'éteindre.
- Refermer le boitier de l'Arduino
- Décrocher le mât du socle fixé sur le toit.

Il est à noter que l'abonnement de la carte SIM du boitier 3G est effectif toute l'année. Il est donc possible d'utiliser le boitier 3G comme point d'accès internet durant l'Hiver.

7.3. Connexions et identifiants

Nous avons accédés à divers services internet pour compléter notre projet. Voici ci ci-dessous les différents identifiants et mot de passe pour y parvenir.

7.3.1. Mail : Gmail

Adresse : stationmeteo.csm@gmail.com

Mot de passe : utbmcsm2015

7.3.2. Hébergeur : ProxGroup

Login : stationmeteo.csm@gmail.com

Mot de passe : utbmcsm2015

Tous les identifiants et codes d'accès, nécessaires au fonctionnement interne du site sont accessibles sous l'interface de gestion.

7.3.3. Station Météo

Mot de passe Administrateur : zutop

Mot de passe Invité : allos