RAPPORT FINAL

L’objectif

Notre objectif initial était de concevoir un appareil permettant de recenser une poignée de données atmosphériques telles que la température, l’humidité, la luminosité et la pression. Nous voulions pouvoir accéder à toutes ces données depuis un appareil tiers, ce qui nous a conduit à réaliser une page web sur laquelle elles seraient publiées.

Cela nous permettrait par exemple, en regardant en début de journée les différentes informations comme la température, l’humidité, le vent ou bien la qualité de l’air, si toutes ces conditions sont adaptées à l’activité que nous avons envie de faire. Il peut s’agir de marche, de vélo, de voile…

Un des objectifs premiers était aussi une certaine simplicité d’accès, nous voulions que cela soit rapide d’accès, sans se prendre la tête. En effet, il suffit de se connecter en Wifi et de lancer le site internet. Et en quelques secondes, toutes les informations se mettent à jour !

De plus nous avions une idée assez précise de notre projet, mais au cours de la réalisation de ce dernier d’autre idées nous sont parvenues et nous avons donc décider de rajouter d’autres modules à notre réalisation.

Ainsi, nous avons respecté notre objectif premier et avons apporté quelques idées supplémentaires à notre projet. Le résultat final nous satisfait et correspond bien à ce que nous avions en tête au début de la fabrication de ce MétéoCube !

Sommaire

1. *Explication du projet*
2. *Algorithmie et présentation du fonctionnement général*
3. *Planning initial / Planning final*
4. *Conclusion et perspectives*

***Une image contenant intérieur

Description générée automatiquement***

Boite finale

Vision globale du projet

Notre projet est une station météo. Il faut donc utiliser d’utiliser de nombreux capteurs pour récupérer les données environnementales.

Les différents éléments de notre projet sont :

* Les capteurs dans un premier temps :
  + DHT 11 : température & humidité
  + BMP 180 : pression atmosphérique
  + Capteur de luminosité
  + Un anémomètre
  + Une girouette
  + PM2.5 by Plantower : qualité de l’air
  + Capteur de radioactivité
* Les cartes Arduino :
  + Arduino Mega
  + Arduino Uno R3
* Les autres cartes :
  + ESP 32
  + Modules intégrés à l’anémomètre / girouette
* Une boite réalisée au FabLab (aggloméré)
* Fils
* Colle, scotch, serre-câbles

1° DHT11 : module alimenté en 5V. Permet de mesurer la température et l’humidité grâce à une thermistance et un capteur d’humidité capacitif. Les données sont transmises sur le pin DATA.

Une image contenant équipement électronique, circuit

Description générée automatiquement

2° BMP180 : module alimenté en 5V. Permet de mesurer la pression grâce à une pièzorésistance. Les données sont transmises par le biais de l’interface I2C.

Une image contenant texte, équipement électronique

Description générée automatiquement

3° LDR : module alimenté en 5V. Renvoie une valeur analogique différente selon la luminosité, rendu possible grâce à une photorésistance. Les données sont transmises sur le Pin A0.

Une image contenant équipement électronique, circuit

Description générée automatiquement

4° Anémomètre : système alimenté en 5V. Compte le nombre de tours effectués pour un temps donné grâce à la fonction AttachInterrupt(). Les données sont transmises sur un pin digital.

5° Girouette : système alimenté en 5V. Permet de connaitre la direction du vent grâce à un encodeur renvoyant une valeur analogique précise selon l’orientation de la girouette. Les mesures, avec celles de l’anémomètre, sont transmises via un câble RJ11 et un module RJ11 vers un pin digital de l’Arduino.

6° PM2.5 : module alimenté en 5V. Dénombre le nombre de particules selon leur diamètre. Fonctionne en projetant un laser sur les particules et selon la déviation des faisceaux, en déduit la taille des particules. Les données sont transmises grâce à un objet SoftwareS.



7° Pocket Geiger : module alimenté en 5V. Dès lors qu'un rayon ionisant (rayon gamma, rayon x, …) est détecté, une capacité proche du tube est déchargée. On peut ainsi en déduire le taux de particules radioactives par litre par heure. Les données sont transmises sur deux pins digitaux.

Une image contenant texte

Description générée automatiquement

8° Mega2560 DiyMore : copie de l’Arduino Mega. Fonctionne en 7V. Permet de gérer plusieurs objets SoftwareSerial et dispose d’un grand nombre de pins analogiques et digitaux. A plus de mémoire dynamique qu’une carte Arduino R3. Dans ce projet, elle permet de recenser toutes les données mesurées par les capteurs.

Une image contenant texte, équipement électronique

Description générée automatiquement

9° Arduino Uno R3 : carte alimentée en 5V. Elle récupère les données transmises par le BMP180, l’anémomètre et la girouette. Elle permet de calculer la pression, la vitesse du vent et la direction du vent selon ces données. Le tout est envoyé à la Mega.

Une image contenant texte, équipement électronique, circuit

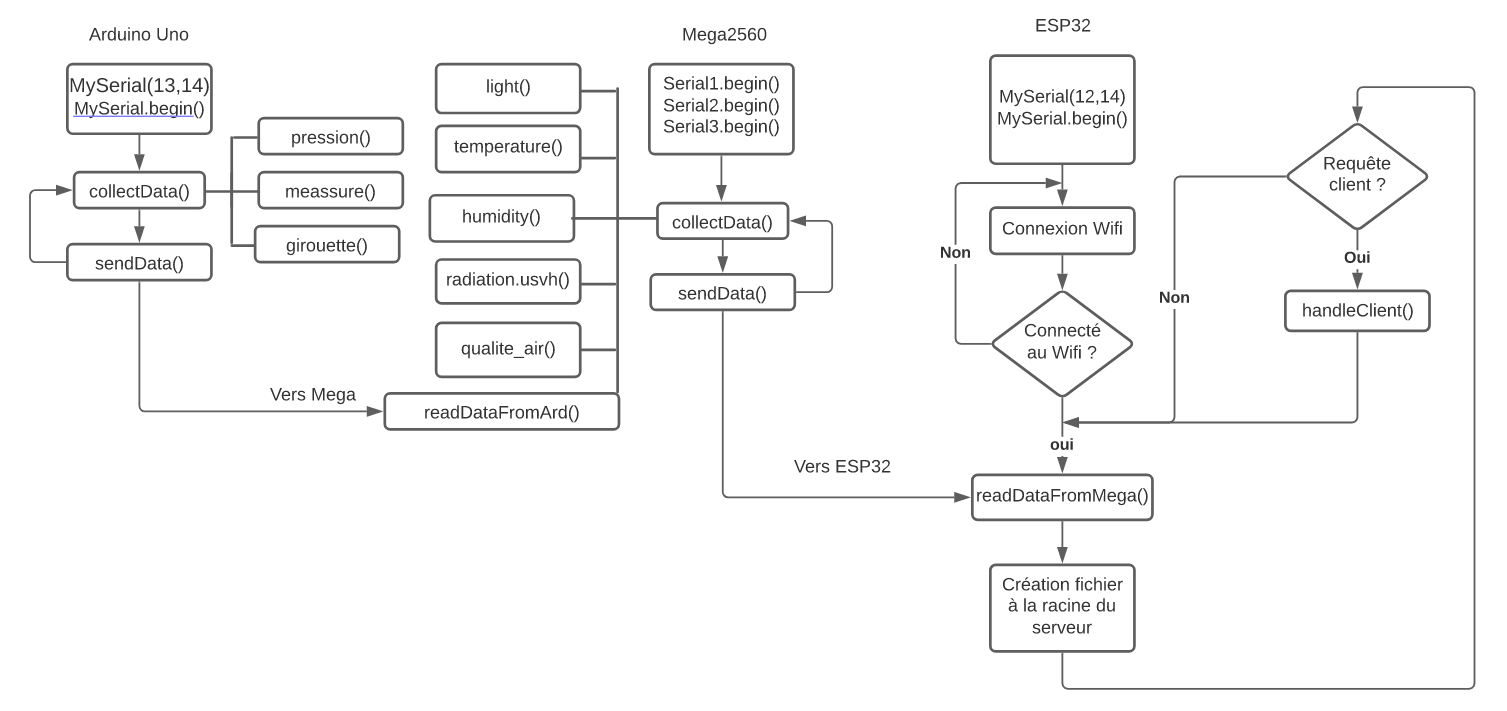
Description générée automatiquement

10° ESP32 : module semblable à l’Arduino R3 mais comporte en plus un module Wifi et Bluetooth. Carte alimentée en 3.3V.

Une image contenant texte, équipement électronique, circuit

Description générée automatiquement

Schéma algorithmique



L’algorithme du projet se décompose en trois sections.

Dans un premier temps, l’Arduino Uno crée un objet SoftWareSerial et le démarre dans le Setup(). Cet objet assurera la communication entre l’Uno et la Mega. Une fois dans le loop(), le programme récolte les données mesurées par les différents capteurs : la pression (en mb) est récupérée par la fonction pression() ; la vitesse du vent (en m/s) est récupérée par la fonction meassure() et la fonction girouette() renvoie l’un des 8 points cardinaux selon l’orientation du vent. Une fois toutes ces données recensées, sendData() les envoie vers l’Arduino Mega par le biais de MySerial().print() etc.

Dans un second temps et en parallèle à l’Uno, la Mega initialise trois objets SoftWareS : un pour la communication vers l’Uno, un pour la communication avec le PMSensor et un pour la transmission de données vers l’ESP32. Dans le loop(), la Mega récupère : la luminosité avec light() ; la température avec temperature() ; l’humidité avec humidity() ; les rayons ionisants avec radiation.uSvh() ; les particules (de diamètre 3, 10 et 50 µm) avec la fonction qualite\_air() et les données transmises par l’Uno avec readDataFromArd(). Toutes ces données sont rassemblées dans une structure de données mesureToSend et sont transmises vers l’ESP32 grâce à la fonction sendData().

Finalement, le programme au bord de l’ESP32 est chargé de crée un objet SoftwareS afin de communiquer avec la Mega. Une fois cela réalisé, l’ESP32 cherche à se connecter au réseau qu’on lui a fournit dans le code. Le programme est gelé jusqu’à ce que la connexion ait été initialisée. Lorsque le module est connecté, celui-ci récupére les données transmises par la Mega par le biais de la fonction readDataFromMega(). Cette dernière permet ainsi à l’ESP32 de démarrer le serveur en créant un fichier HTML à sa racine. De plus, la fonction handleClient() permet de gérer les requêtes client reçues. En tant que client, la seule requête que l’on réalisera sera l’affichage du fichier HTML sur notre navigateur.

Tous ces programmes s’exécutent en boucle et indépendamment les uns des autres. C’est-à-dire que si aucune donnée n’est transmise d’une carte à l’autre, aucune erreur ne sera engendrée. Bien sûr, cela altèrera l’affichage des données sur le fichier HTML.

Une image contenant texte

Description générée automatiquementUne image contenant texte, capture d’écran, moniteur, intérieur

Description générée automatiquement

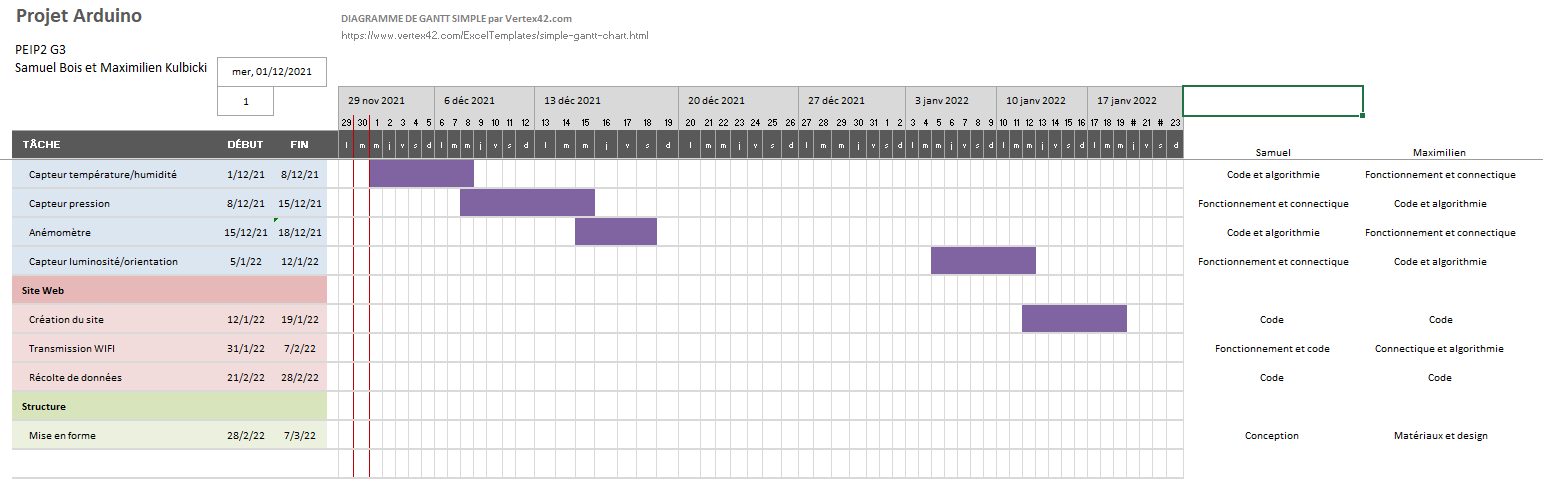
Sortie sur le site

Sortie sur l’arduino

Les plannings initiaux et finaux

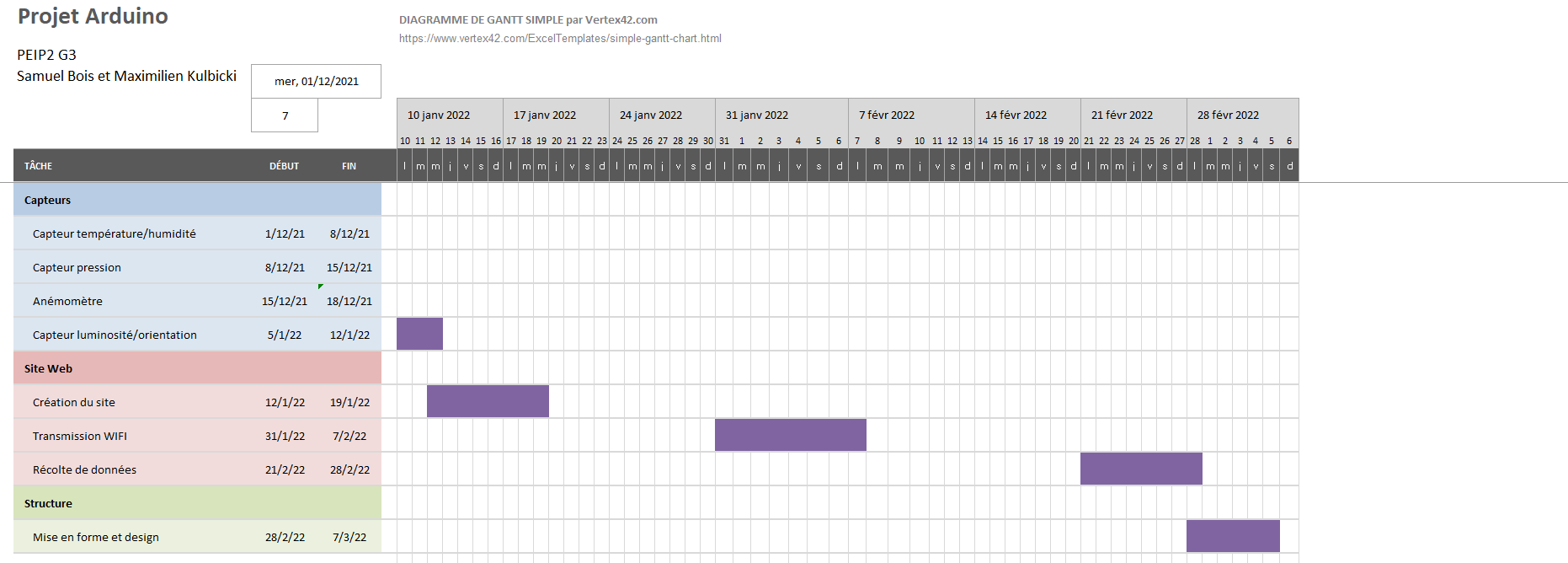
Nous avons essayé de respecter le planning que nous nous étions imposés, dans la mesure du possible. On a été dans un premier temps plus rapide que prévu car nous avions prévu trop de temps sur les tâches rapides comme la mise en marche de certains capteurs, et au contraire, dans la seconde moitié du projet, nous manquions de temps pour faire fonctionner la partie site / ESP32 et la liaison entre les deux. Finalement, nous avons réussi à retomber sur nos pattes, car à la dernière séance nous avons confectionner notre boîte, comme il était prévu sur le planning initial.

Il a été compliqué de tenir un planning, mais surtout il est dur d’estimer le temps que prend des tâches qui sont nouvelles pour nous, en autonomie. Il faut donc toujours se laisser une marge supplémentaire en cas de problème/retard majeur qui pourrait nous faire prendre une/plusieurs semaine(s) de retard.



Planning initial

Planning proche de la réalité



Conclusion & perspectives

Et finalement…

Qu’est-ce qui marche ? notre station météo renvoie 9 données en temps réel sur un site internet de notre choix.

Qu’est-ce qui ne marche pas ? Rien par rapport à l’idée d’origine.

Et quelles améliorations … ?

On peut imaginer différentes améliorations que l’on aurait pu apporter si l’on avait à nouveau 8 séances pour continuer à travailler sur ce projet.

On utiliserait des batteries pour alimenter les différentes cartes. C’est-à-dire une batterie avec plusieurs ports (3) ou bien 3 batteries différentes qui pourraient fournir 5V, 5V et 7V. Cela permettrait de rendre autonome notre projet.

De cette façon un premier pas dans l’autonomie est atteint, on pourrait bien atteindre un deuxième stade en n’ayant pas besoin de recharger nos batteries portables. En installant des panneaux photovoltaïques sur le dessus de la station météo (à voir pour la possibilité avec notre boîte). Cela permettra de rendre complètement autonome notre station.

On rajouterait aussi d’autres capteurs. Nous en avons vu qui pourraient être utile comme un hygromètre que l’on implanterait dans le sol, un magnétomètre et bien d’autres capteurs.

Nous pourrions aussi utiliser les données pour les mettre dans des bases de données, cela permettrait de faire des graphiques, comparer l’évolution des différentes données au cours du temps (que ce soit sur des jours, mais finalement sur plusieurs mois), faire des moyennes dans le but de comprendre l’environnement qui nous entoure.

Bibliographie :

* DHT11 : [[GUIDE] Utiliser un capteur de température et d'humidité DHT11 - Arduino France](about:blank)
* BMP180 : [https://microdigisoft.com/bmp180-pressure-sensor-interfacing-with-arduino/](about:blank)
* Anémomètre et girouette : [https://github.com/Leduodeshok/MeteoStation/blob/rapports-seances/CODES/CODE\_FINAL\_ARD\_EXTERIEUR/code\_AGHT.ino](about:blank)
* ESP32 : [https://www.electronicshub.org/esp32-pinout/](about:blank)

[https://www.programmingelectronics.com/sprintf-arduino/](about:blank)

[https://www.tubefr.com/arduino-esp8266-post-donnees-au-site-web\_2.html#title](about:blank)

* Pocket Geiger : [https://github.com/MonsieurV/ArduinoPocketGeiger](about:blank)
* PMSensor 2.5 Plantower : [http://www.aqmd.gov/docs/default-source/aq-spec/resources-page/plantower-pms5003-manual\_v2-3.pdf](about:blank)
* MEGA2560 DiyMore : [https://www.etechnophiles.com/arduino-mega-pinout-pin-diagram-schematic-and-specifications-in-detail/](about:blank)
* LDR : [https://www.electrotoile.eu/arduino-fabriquer-un-interrupteur-crepusculaire-avec-photoresistance-LDR.php](about:blank)
* MakerCase : [https://fr.makercase.com/#/basicbox](about:blank)
* L’application InkScape