

BERTELOOT Léo, POULEYN Maxime,

DABACH Achraf, GUILBERT Romain

15 Mai 2022

RAPPORT DE PROJET

Centrale de Mesure de la Pollution Intérieure

Sommaire

| | |
|---|-----------|
| Introduction | 3 |
| Cahier des Charges | 4 |
| Dossier de spécifications | 4 |
| Besoins des utilisateurs | 5 |
| Description UML | 6 |
| Diagramme de cas d'utilisation - donné par le client | 6 |
| Diagrammes de séquence | 7 |
| Diagramme de base de données - par Maxime POULEYN | 10 |
| Ressources mises à disposition | 11 |
| Partie de Maxime POULEYN : GPS, Thermomètre et Administration | 12 |
| Devis : Thermomètre et GPS | 13 |
| Vérification du thermomètre et du GPS | 14 |
| Capteur de température | 14 |
| GPS | 16 |
| Installation du Thermomètre | 17 |
| Installation du GPS | 18 |
| Utilisation simultanée du GPS et du Thermomètre | 19 |
| Création de la partie administrateur du site | 20 |
| Partie de Léo BERTELOOT : Liaison Raspberry - Arduino et base de donnée & Création de l'interface Historique | 21 |
| Installation de NodeJS sur Raspberry | 22 |
| Installation des librairies NodeJS | 23 |
| Conception du script NodeJS | 24 |
| Création du système de pagination | 26 |
| Création de l'interface graphique du site | 27 |
| Mise en place de l'HTML | 27 |
| Mise en place du CSS | 28 |
| Conception de l'interface d'accueil | 30 |
| Système d'actualisations des valeurs via AJAX et JQuery | 30 |
| Système de notification via popup (ToastJS) | 31 |
| Partie de Guilbert Romain : Partie utilisateur du site et Capteur d'humidité et de particules | 32 |
| Partie de Dabach Achraf : Liaison Raspberry - Arduino et base de donnée & Capteur de COV et CO2 : | 33 |
| Devis final | 38 |

| | |
|--|-----------|
| Annexe A1 - Documentation client | 40 |
| Annexe A2 - Documentation du matériel utilisé | 41 |
| A2.1 - Thermomètre de Précision | 41 |
| A2.2 - Capteur de température et d'Humidité | 41 |
| A2.3 - Capteur de gaz : eCO2 et VOC | 41 |
| A2.4 - Capteur laser pour micro-particules | 41 |
| A2.5 - Capteur de qualité de l'air | 41 |
| Annexe A3 - Codes programmation | 42 |
| A3.1 - Arduino | 42 |
| A3.2 - Raspberry | 42 |
| A3.2.1 - C++ connexion série | 42 |
| A3.2.2 - Fichier Json | 42 |
| A3.2.3 - NodeJS | 43 |
| A3.2 - Site Web | 44 |
| A3.2.1 - Système de pagination | 44 |
| A3.2.1 - Accueil | 44 |

Introduction

Le projet de Centrale de Mesure de la Pollution Intérieure vise à créer une centrale numérique qui servira à mesurer différentes valeurs permettant de connaître la qualité de l'air :

- la concentration en particules fines ;
- la concentration en composés volatiles COV ;
- la concentration en dioxyde de carbone CO2 ;
- la température et l'humidité;

Les mesures seront faites par des capteurs correspondants, connectés à une carte arduino, et les données récupérées seront stockées dans une base de données, et visibles depuis un site web, le tout hébergé sur une raspberry pi.

Le site sera accessible depuis tout appareil se connectant à la Wifi de la centrale, qui utilisera le wifi intégré à l'arduino.

La centrale sera équipée d'un GPS afin de pouvoir la situer en cas d'utilisation de plusieurs centrales dans un même territoire.

Cahier des Charges

Dossier de spécifications

- Quantifier la pollution intérieure par des mesures objectives.
 - Mesurer la température, en °C, de 0°C à 40°C par pas de 1°C
 - Mesurer l'humidité, en %, de 0% à 100% par pas de 1%
 - Mesurer la concentration en CO₂, en ppm, de 400 ppm à 1479 ppm par pas de 1 ppm
 - Mesurer la concentration en COV, en ppb, de 0 ppb à 2000 ppb par pas de 1 ppb
 - Mesurer la concentration en NO₂, en ppm, de 400 ppm à 1479 ppm par pas de 1 ppm
 - Mesurer les particules fines, en $\mu\text{g}/\text{m}^3$, et les catégoriser : 0.3 μg , 0.5 μg , 1 μg , 2.5 μg , 5 μg , 10 μg
 - Stocker de manière locale les données concernants les mesures et les types de capteurs installés
 - Permettre aux différents types d'utilisateurs de consulter ou de paramétriser la centrale à partir d'un navigateur internet
 - L'utilisateur doit pouvoir consulter les valeurs mesurées et accéder à un historique des mesures
 - L'administrateur hérite des fonctionnalités de l'utilisateur avec la possibilité de paramétriser les capteurs
- Localiser la centrale avec un GPS

¹ Page par Maxime Pouleyn

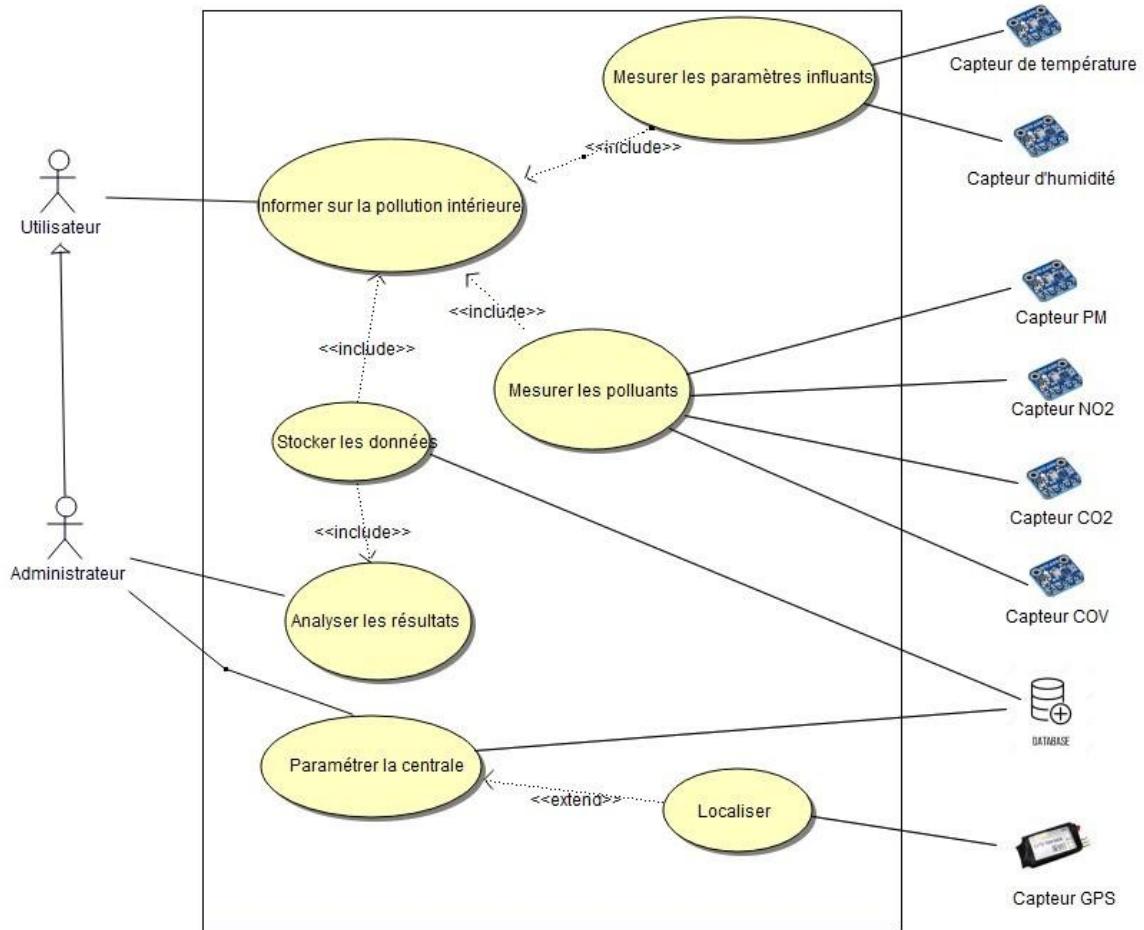
Besoins des utilisateurs

- La centrale doit être transportable
 - La transportabilité n'inclut pas la portabilité, une prise secteur reste nécessaire
 - Un GPS est inclus pour la localisation
 - L'utilisation de plusieurs centrales en même temps doit être envisagée
- L'utilisation doit se faire par un navigateur
 - Un site doit donc être créé et installé sur la centrale
 - Le site doit être accessible depuis d'autres appareils
 - La centrale doit donc avoir un moyen pour que d'autres appareils s'y connectent (sûrement WiFi)
- L'interface du site doit être facile d'utilisation
 - Une personne non-informaticienne doit pouvoir modifier les paramètres
 - Utilisation de boutons, sliders pour les paramètres
 - La lecture des données doit se faire naturellement
 - Code couleur pour la qualité de l'air pour chaque mesures
 - Affichage du nom du capteur au passage de la souris (ou équivalent sur écran tactile)
- La maintenance doit être facile
 - Câblage interne simple, de type plug & play (utilisable dès l'installation)
 - Intérieur pouvant être visible, pour de possibles présentations durant des forums
- L'administration doit pouvoir faire les actions nécessaires
 - Authentification par pseudo, mot de passe et mail valide
 - Activation de la fonction GPS
 - Export de la BDD pour consultation externe
 - Obtention de mesures poussées (pics, moyennes, minimums sur une plage de mesures précise)

² Page par Maxime Pouleyn

Description UML

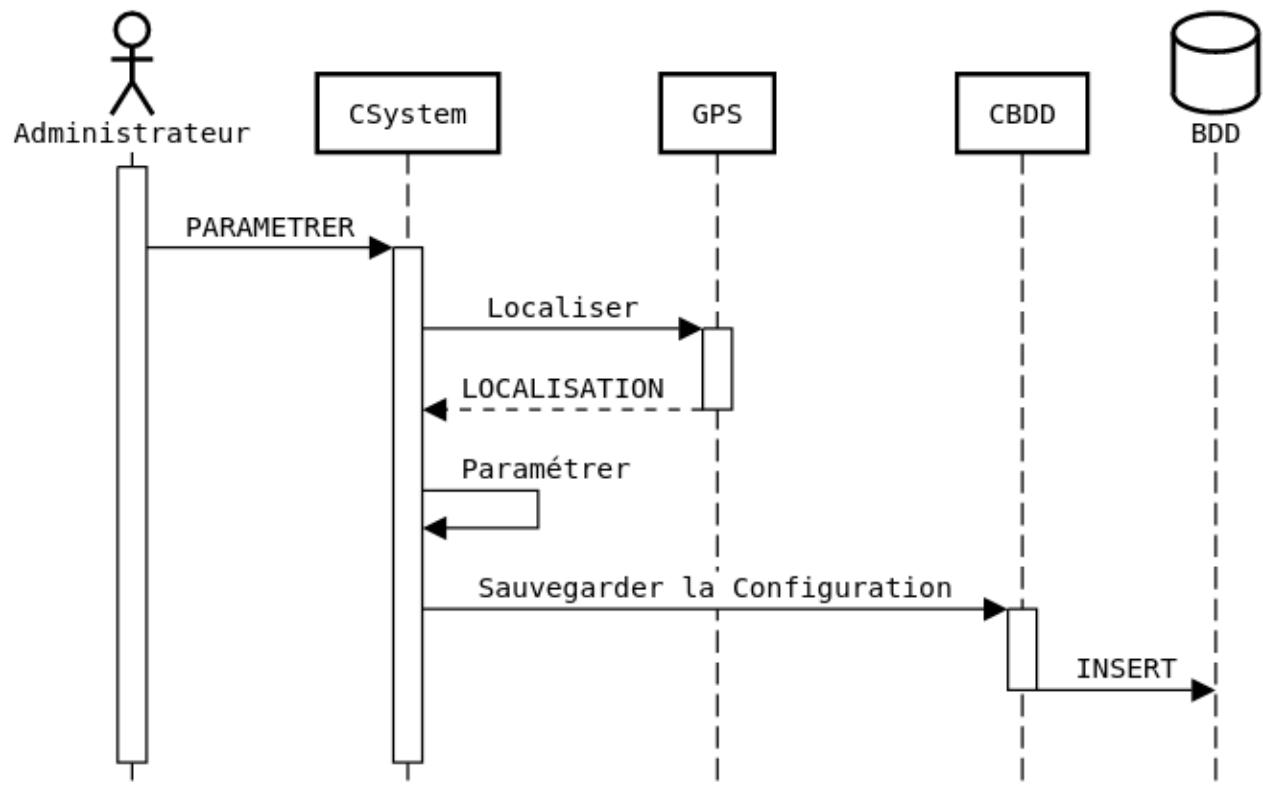
Diagramme de cas d'utilisation - donné par le client



³ Page par Maxime Pouleyn

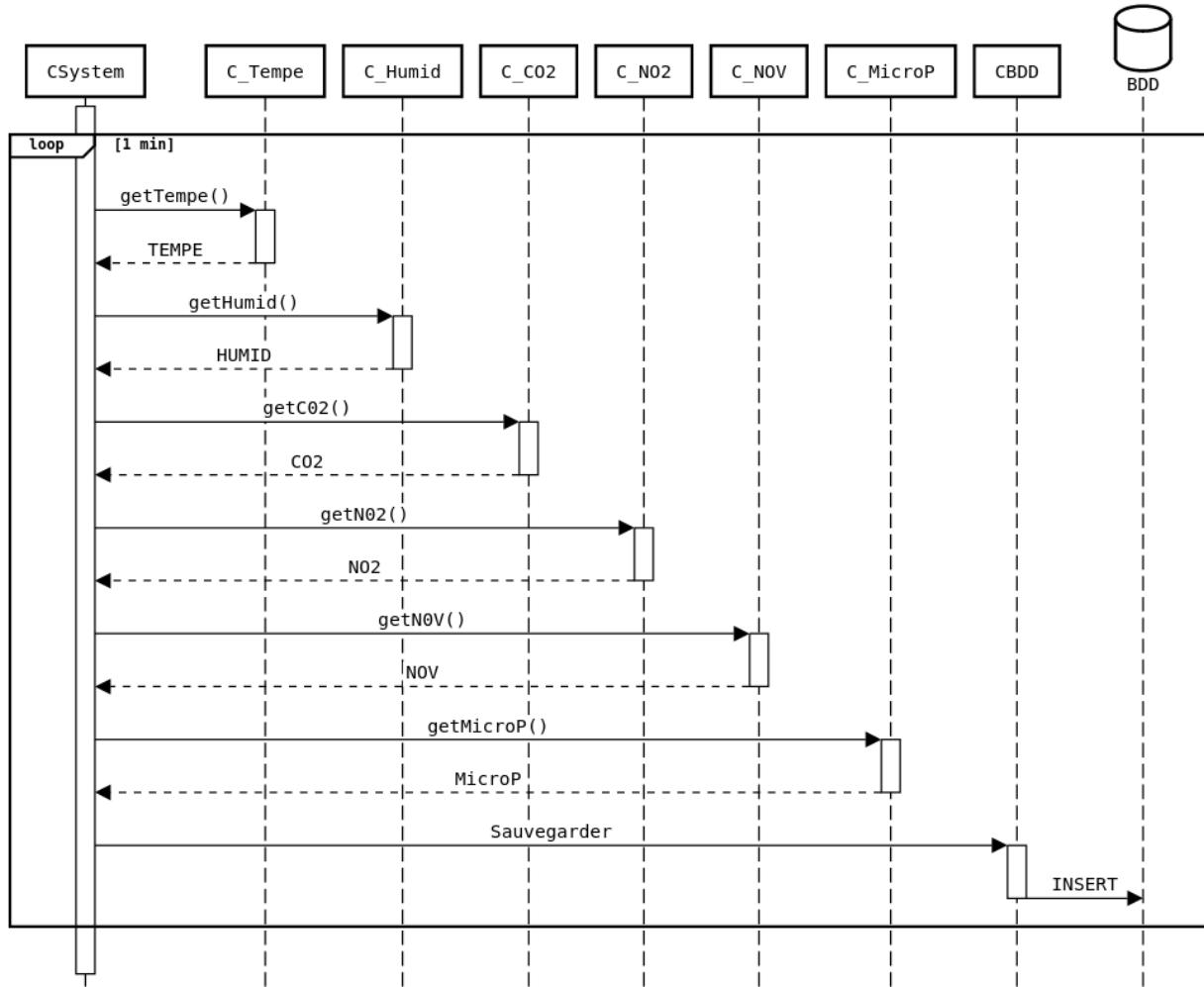
Diagrammes de séquence

Projet Centrale de Pollution - Paramétrter la centrale



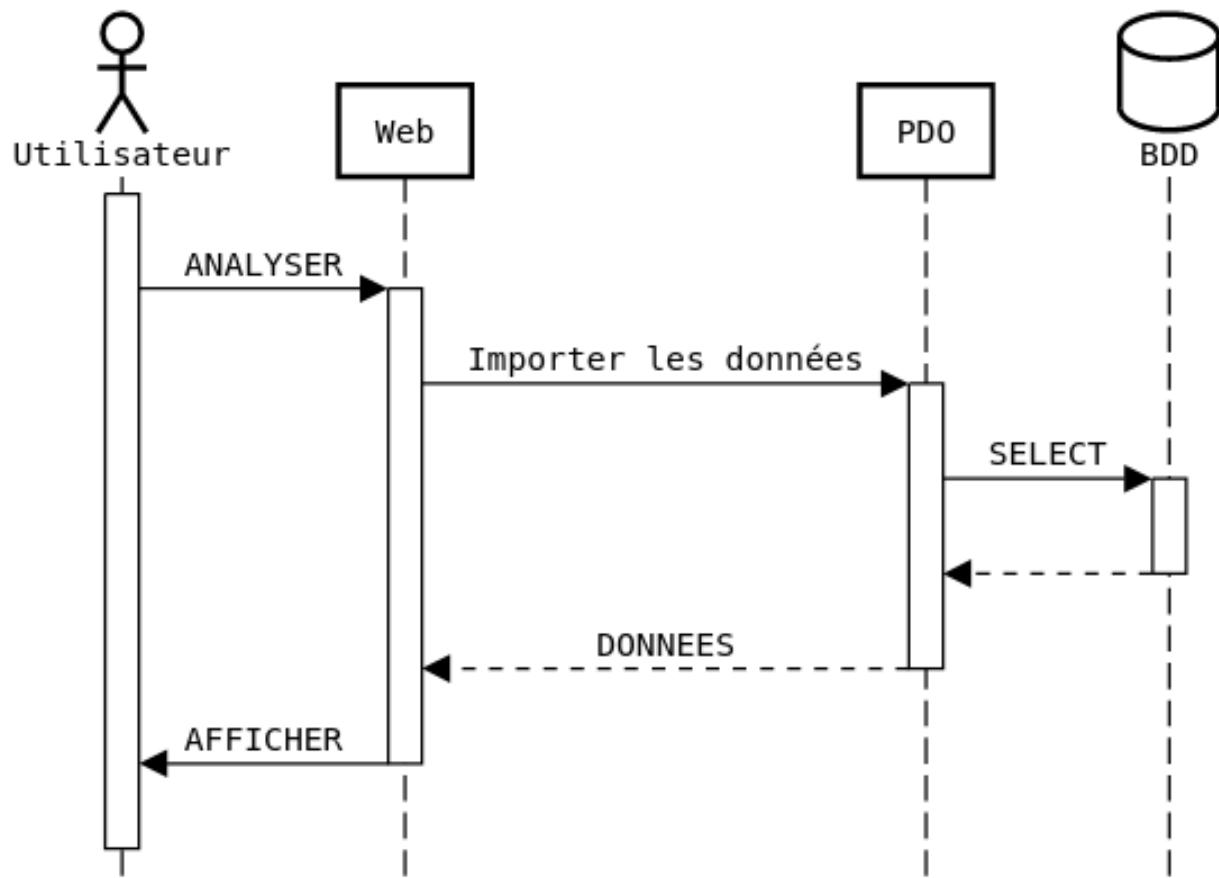
⁴ Page par Maxime Pouleyn

Projet Centrale de Pollution - Informer sur la pollution intérieure



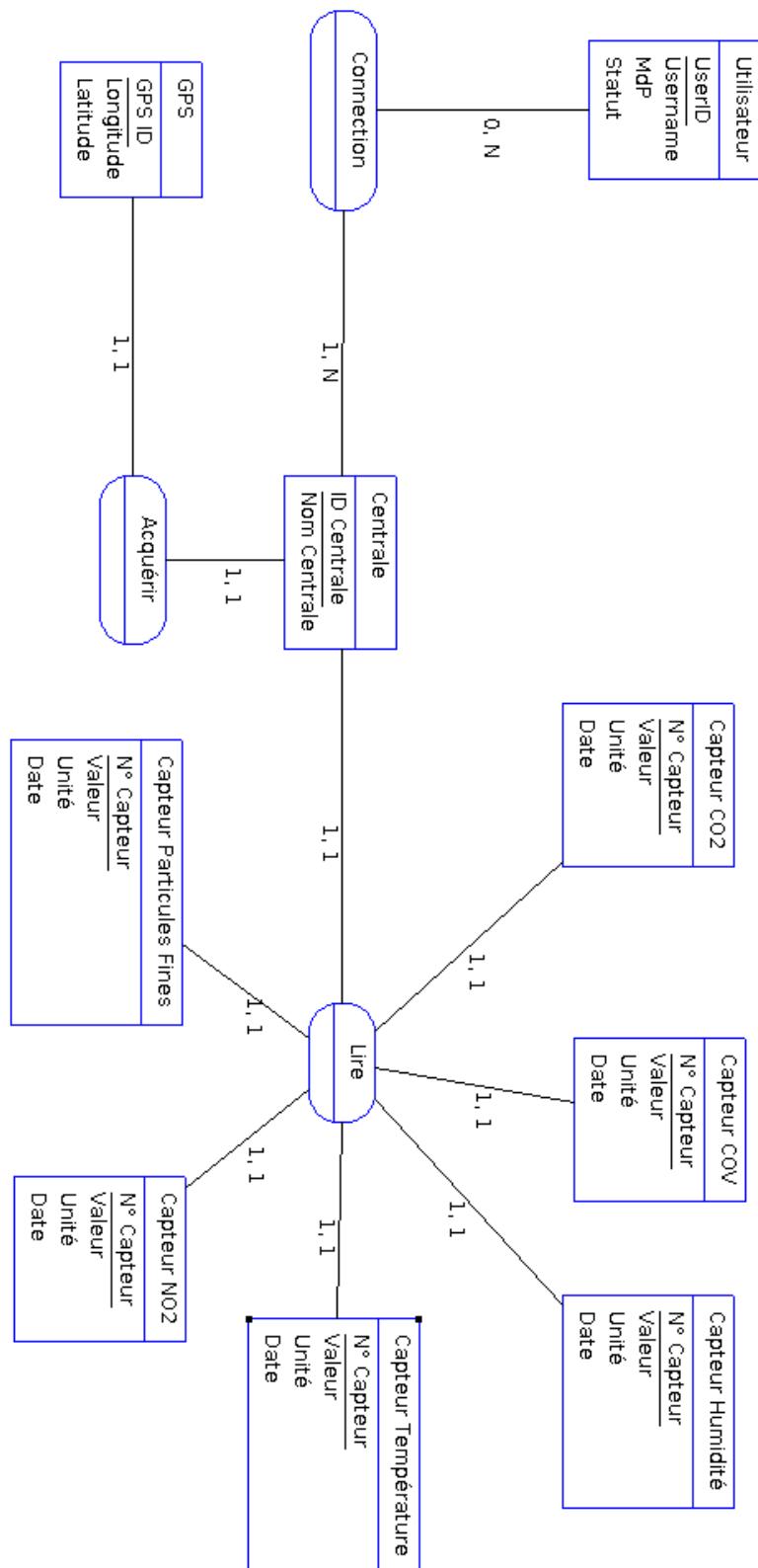
⁵ Page par Maxime Pouleyn

Projet Centrale de Pollution - Analyser les résultats



⁶ Page par Maxime Pouleyn

Diagramme de base de données - par Maxime POULEYN

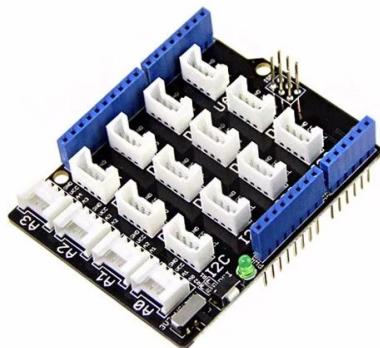


Ressources mises à disposition

Au début étaient mis à notre disposition plusieurs capteurs (tous provenant de la gamme Grove de Seeed studio et compatibles I2C), et un Arduino Uno. Les capteurs sont les suivants (leur documentations sont trouvables dans l'annexe A2) :

- Un thermomètre haute précision
- Un capteur de température et d'humidité
- Un capteur de gaz CO₂eq et COV
- Un capteur à micro-particules
- Un capteur de qualité de l'air (pour gaz dangereux)
- Un GPS SIM28

Au bout de quelques semaines, certaines commandes ont pu être passées et on a obtenu du matériel en plus, comme une Arduino Uno en plus (permettant de travailler sur deux capteurs en même temps avant de les mettre en commun), une raspberry Pi 3B+, et un shield pour Arduino Uno, une plaque se mettant par dessus une carte électronique afin d'y ajouter des fonctionnalités (dans notre cas des ports grove afin de faciliter le branchement).



Shield Grove pour Arduino Uno



Raspberry Pi 3 B+

⁸ Page par Maxime Pouleyn

Partie de Maxime POULEYN : GPS, Thermomètre et Administration

Ma partie consiste, en dehors de la liaison Raspberry-Arduino et de la base de données qui sont communs à tous les membres, en l'installation d'un capteur de température et d'un GPS, en plus de la création de l'interface administrateur pour le site web.

Le capteur de température devra servir à l'estimation du fonctionnement de certains capteurs dont le fonctionnement en dépend, et est aussi une mesure basique qui peut être utile dans certains contextes (exemple : salles devant garder une certaine température).

Le GPS servira à déterminer la position de la centrale de mesure, en particulier lors de l'utilisation de plusieurs d'entre elles dans des entrepôts ou des bâtiments ayant une taille considérable nécessitant plusieurs centrales.

L'interface administrateur sera une partie à part entière du site web de la centrale. Elle servira à pouvoir gérer les utilisateurs (suppression et modification de comptes), et à gérer les capteurs de la centrale (modification du pas). Un utilisateur de type administrateur doit pouvoir continuer d'utiliser les parties non-administrateur du site comme un utilisateur normal.

Les objectifs sont les suivants :

- Pouvoir mesurer la température ambiante;
- Pouvoir situer la centrale (utile en cas d'utilisation de plusieurs centrales sur un même site);
- Avoir une interface administrateur sur le site permettant la gestion d'utilisateurs et des capteurs.

Devis : Thermomètre et GPS

Ceci est une estimation du prix du matériel que j'utilise. Les prix peuvent varier de ceux du devis pour plusieurs raisons :

- Fournisseur différent
- Rupture de stock (le MCP9808 était, au 13/05/2022, hors stock sur le site de Seeed studio)
- Le module utilise des composants en rupture de stock
- Prix de la livraison variable

| Description | Quantité | Prix unitaire HT | Prix total HT |
|---|----------|------------------|---------------|
| MCP 9808 Capteur de température, de Seeed Studio | 1 | 5,00 € | 5,00 € |
| GPS Grove Module GPS, antenne non-inclue, de Seeed Studio | 1 | 25,00 € | 25,00 € |
| Antenne GPS Antenne pour le module GPS, de Seeed Studio | 1 | 5,00 € | 5,00 € |

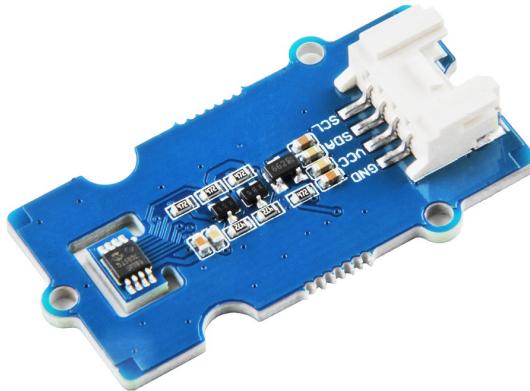
| | |
|------------------|----------------|
| Total HT | 35,00 € |
| TVA (20,00 %) | 7,00 € |
| Total TTC | 42,00 € |

Vérification du thermomètre et du GPS

Comme expliqué précédemment dans le rapport, on avait à notre disposition dès le départ plusieurs capteurs à notre disposition, mis à disposition en attendant le passage de potentielles commandes de matériel. Il nous a donc fallu dans un premier temps déterminer lesquels nous seraient utiles et lesquels seraient inutiles, en plus de considérer les demandes du cahier des charges.

Dans mon cas, j'ai dans un premier temps vérifié que tous les capteurs fonctionnaient un par un, incluant ceux de mes camarades, avant de me concentrer sur ceux qui avaient la possibilité de m'être utiles dans mes tâches.

Capteur de température



Pour le capteur de température j'avais deux choix déjà présents :

- Le capteur Grove - I²C High Accuracy Temperature Sensor MCP9808, un capteur se voulant précis et n'utilisant un code de base ne comportant qu'une dizaine de lignes (librairie excluse).
- Le capteur Grove - Temperature & Humidity Sensor (que je vais abréger en T&H), un capteur hybride température et humidité, mais avec une précision amoindrie et un code bien plus lourd.

Ainsi le choix se portait entre la précision ou la simplicité. J'ai donc voulu vérifier si l'un des capteurs est inadapté à notre projet :

| | Cahier des Charges | MCP9808 | TH02 |
|--------------------|--------------------|-----------------------|------------|
| Portée des mesures | 0°C - +40°C | -40°C - +125°C | 0°C - 70°C |
| Pas de mesure | ±1°C | ±0,25°C, ±0,5°C, ±1°C | |

Comme on peut le voir, les deux capteurs sont adaptés aux caractéristiques voulues. J'ai donc discuté avec mes camarades sur le chemin à prendre, et on s'est mis d'accord sur le MCP9808, plus de précision en dehors des limites demandées peut être utile, spécialement dans une région comme Dunkerque où la température peut descendre en dessous de 0°C en hiver.

On peut aussi regarder les consommations électriques afin de justifier le choix :

| | Intensité | Tension | Energie |
|---------|-----------|---------|---------|
| MCP9808 | 200 µA | 5 V | 1 mWh |
| T&H | 320 µA | 5 V | 1,6 mWh |

Le T&H, même s'il permet de n'utiliser qu'un capteur pour deux mesures, a plusieurs défauts. Il est plus lourd niveau programmation, et vu le nombre de capteurs que notre arduino devra supporter mieux vaut limiter le plus possible la taille du code de chaque capteur. Aussi on a séparé les tâches avant de regarder les capteurs disponibles de sorte que les mesures de la température et de l'humidité soient faites par deux personnes différentes.

En tant que tel l'utilisation d'un capteur fusionnant les deux pourrait ralentir l'avancée du projet plus qu'elle ne l'avantagerait, nous menant au choix du MCP9808.

GPS



Pour le GPS il ne nous était présenté qu'un choix de base : le GPS Grove Sim28 avec interface I²C. Et même après avoir regardé les autres offres de GPS, il était clair que le Grove Sim28 avait un avantage : l'interface I²C, facilitant le câblage.

De plus, il a deux caractéristiques que l'on recherche :

- une programmation simple et légère;
- une consommation électrique basse.

Ainsi il n'est pas requis de rechercher un autre modèle, celui-ci étant suffisant.

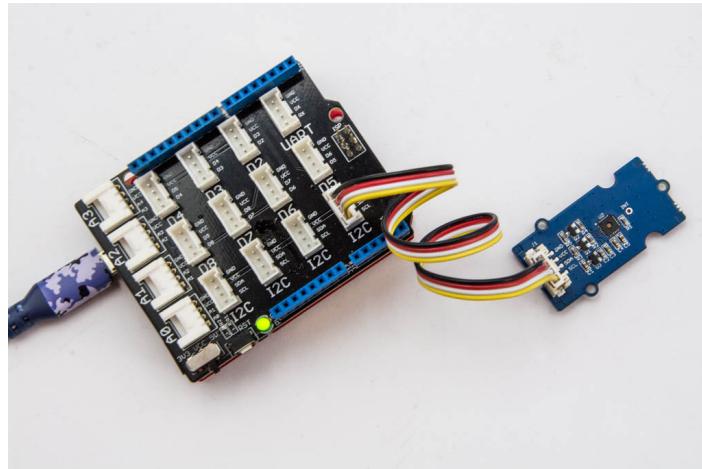
Il est intéressant de noter qu'il existe une version Air 530 du GPS Grove, présentée comme une version plus adaptée aux milieux urbains ou aux situations où peu de satellites GPS sont disponibles.

Mais d'après certains tests que l'on a réalisé avec le GPS Grove Sim28, ce dernier fonctionne assez bien en intérieur en zone urbaine pour qu'on le garde au lieu de changer, de plus l'Air 530 a une plus grande consommation électrique :

| | Intensité | Tension | Energie |
|-----------------------|-----------|---------|---------|
| Sim28 - Acquisition | 24 mA | 5 V | 120 mWh |
| Sim28 - Suivi | 19 mA | 5 V | 95 mWh |
| Air 530 - Acquisition | 42,6 mA | 5 V | 213 mWh |
| Air 530 - Suivi | 36,7 mA | 5 V | 183 mWh |

Installation du Thermomètre

Comme expliqué plus tôt, le thermomètre utilise une interface I²C, rendant le câblage simple : on branche le câble sur un des ports I²C du Shield Grove sur l'Arduino.



La partie programmation est presque aussi simple. Un exemple fonctionnel est donné avec la librairie du capteur une démo de base, sur laquelle je vais baser mon code :

```
1 #include "Seeed_MCP9808.h"                                //include de la librairie
2 MCP9808 sensor;                                         //Déclaration du capteur
3 void setup() {                                          
4     Serial.begin(115200);                                 //En 115200 bauds
5     if (sensor.init()) {                                 //Message d'erreur si l'initialisation échoue
6         Serial.println("sensor init failed!!"); 
7     }                                                       //Message si l'initialisation est réussie
8     Serial.println("sensor init!!"); 
9 }
10 void loop() {                                         
11     float temp = 0;                                     //Température stockée en float
12     sensor.get_temp(&temp);                            //Récupère la température dans la variable temp
13     Serial.print("temperature value is: ");
14     Serial.println(temp);
15     delay(1000);|
16 }
```

Lorsque l'on mettra ensemble les différents capteurs, la manière la plus simple de le faire sera de simplement fusionner les parties Setup et Loop de chaque programme individuel, en faisant quelques adaptations (par exemple, avec deux capteurs I²C il faut faire attention à ne pas avoir deux fois les mêmes adresses I²C).

Installation du GPS

Le GPS a lui aussi une interface I²C, mais contrairement au thermomètre ce n'est qu'en apparence, le câble devant être branché sur un port Digital (ici on a choisi D2 par défaut, celui-ci étant celui donné pour les exemples) :

Au niveau de la programmation, le code donné en exemple sur le Seeed wiki est un programme utilisé pour lier le GPS à un logiciel Windows, ce qui est totalement opposé à ce qu'on cherche : un programme simple, léger et qui nous permet d'utiliser Raspberry, qui est une distro Linux. Afin d'atteindre notre objectif on réalise un autre programme :

- On récupère la sortie du GPS, qui est sous la forme d'un string contenant une trame
- On vérifie si la trame est une trame GPGGA (trame contenant les coordonnées GPS)
- On sépare la trame GPGGA en plusieurs morceaux : un float pour la longitude, un char pour N ou S : Nord/Sud, idem pour la latitude et Est/Ouest;
- On envoie les coordonnées ainsi récupérées à la Raspberry

On remarque que le programme ainsi créé est très long, avec plusieurs variables, on peut donc le raccourcir pour juste récupérer la trame GPGGA puis l'envoyer à la Raspberry, qui en extrait elle-même les coordonnées :

Utilisation simultanée du GPS et du Thermomètre

Afin de pouvoir utiliser les deux appareils ensembles, il a fallu fusionner les codes utilisés par chacun d'entre eux, surtout dans la partie loop :

```
30 void loop() {
31     //Partie GPS
32     if (SoftSerial.available())
33     {
34         while(SoftSerial.available())
35         {
36             buffer[count++]=SoftSerial.read();
37             if(count == 64)break;
38         }
39         if(isGPGGA()==true)
40         {
41             writeNS();
42             writeEW();
43         }
44         clearBufferArray();
45         count = 0;
46     }
47     //Fin GPS, Thermomètre ensuite
48     temperature();
49     //Fin thermomètre, attendre 1 seconde et re-loop
50     delay(1000);
51 }
```

Après tests, il a été conclu qu'une modification du code, ou l'achat d'une Arduino plus performante, est à prévoir : l'ajout d'un capteur en plus par la suite a suffit pour faire atteindre la limite de stockage de l'Arduino.

Le croquis utilise 10030 octets (31%) de l'espace de stockage de programmes. Le maximum est de 32256 octets.
Les variables globales utilisent 1907 octets (93%) de mémoire dynamique, ce qui laisse 141 octets pour les variables locales. Le maximum est de 2048 octets.
La mémoire disponible faible, des problèmes de stabilité pourraient survenir.

Code d'erreur de la compilation avec l'ajout du SGP30 (capteur de COV et eCO2)

Une possibilité serait d'optimiser l'utilisation des variables : moins de chaînes de caractères à afficher durant le fonctionnement (ce qui n'est utile qu'en phase de développement), faire gérer la décomposition de la trame GPGGA par la Raspberry, ou bien optimiser les librairies (option la moins favorable, augmentant considérablement le temps nécessaire au développement).

Création de la partie administrateur du site

Pour créer la partie administrateur du site, je me suis basé sur le site créé par Léo, mais remanié pour fonctionner sous PHP et une base de données en SQL.

Ainsi l'interface graphique et les parties créées sous node-js sont reproduites à l'identique, et les parties utilisant AJAX et le JQuery restent sous ses systèmes là.

La partie administrateur consiste en deux pages : une page permettant de créer/modifier/supprimer des comptes utilisateur, et une autre pas encore créée au moment de l'écriture de ces lignes qui permettra de gérer les capteurs.

The screenshot shows a web application interface for managing users. On the left is a sidebar with a logo, the name 'Zoroxus', and navigation links: Accueil, Historique, Capteurs, and Gestion utilisateurs (which is highlighted with a blue background). The main content area has a title 'Gestions utilisateurs'. Below it is a table with columns: ID, Nom d'utilisateur, Adresse E-mail, Role, and Actions. There are two rows of data:

| ID | Nom d'utilisateur | Adresse E-mail | Role | Actions |
|----|-------------------|-------------------------|-------|---|
| 32 | admin | leo.berteloot@gmail.com | admin | <button>Destituer</button> <button>Supprimer</button> |
| 39 | Zoroxus | maxime.zorox@gmail.com | admin | <button>Destituer</button> <button>Supprimer</button> |

An error message is displayed in a red box at the top right: 'Erreur lors de la récupération des données' (Error during data retrieval).

Page de gestion des utilisateurs du site

Partie de Léo BERTELOOT : Liaison Raspberry - Arduino et base de donnée & Création de l'interface Historique

Ma partie consiste à effectuer la liaison entre le Raspberry et l'Arduino par le biais d'un script NodeJS ainsi que le développement de l'interface par défaut du site internet permettant la visualisation des valeurs.

Le script NodeJS servira principalement à communiquer entre le Raspberry et l'Arduino les valeurs de chaque capteur afin de les insérer dans la base de données.

L'interface par défaut du site contiendra quant à elle, toute l'interface graphique du site, ainsi que sa page d'accueil, sur cette dernière, nous serons en mesure d'y voir un diagramme ainsi que les valeurs de chaque capteur actualisés toutes les 2 minutes. L'interface possédera également un système de notification indiquant par le biais d'une pop-up rouge, lorsqu'un capteur possède une valeur anormale.

Les objectifs seront donc:

- Installation de NodeJS sur Raspberry
- Création du script NodeJS;
- Communication entre le Raspberry et l'Arduino;
- Création de l'interface graphique du site
 - Mise en place de l'HTML
 - Mise en place du CSS
- Création du système de pagination;
- Création de la page d'accueil:
 - Système d'actualisation des valeurs via AJAX et JQuery
 - Système de notification via pop-up

Installation de NodeJS sur Raspberry

L'installation de NodeJS sur Raspberry s'effectue comme une installation classique de NodeJS sur d'autres noyau Linux.

Afin d'installer NodeJS, il est préférable de mettre à jour tous les paquets du système d'exploitation Linux, en utilisant la commande suivante:

```
sudo apt-get update && sudo apt-get upgrade
```

Pour installer NodeJS, il est nécessaire d'avoir le paquet **cURL**, un paquet permettant de récupérer le contenu d'une ressource accessible depuis un réseau informatique.

```
curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -
```

Une fois cela effectué, nous pouvons procéder à l'installation de nodeJS par le biais de la commande apt-get install à lancer **obligatoirement** en super utilisateur (sudo)

```
sudo apt-get install -y nodejs
```

A la fin de l'installation, il est conseillé de vérifier la version de NodeJS ainsi que de NPM, une librairie permettant l'installation de librairie open-source de NodeJS

```
node --version  
npm --version
```

Ainsi, l'installation de nodeJS est désormais terminée. Cependant, l'installation des paquets doit être effectuée.

Installation des librairies NodeJS

Le script NodeJS utilise divers librairies tel que:

1. MongoDB
 - a. MongoDB permet la synchronisation, l'insertion et la mise à jour de données au sein d'une base de données. MongoDB fonctionne sous divers langages tels que NodeJS, PHP et bien d'autres.
2. Colors
 - a. Colors permet la coloration dans la console de NodeJS, cela permet, en outre, d'y afficher en couleur, les erreurs ainsi que les messages importants ou non. Ici, les messages d'erreurs y seront indiqué en rouge, les messages importants en jaunes, les messages de succès en vert.
3. FS
 - a. FS signifie "File System", cela permet de naviguer depuis un script NodeJS dans les fichiers du système d'exploitation afin de permettre la lecture, l'écriture sur les fichiers autorisés par l'utilisateur authentifié.
4. Node-cron
 - a. Node-cron permet la programmation de tâches à exécuter à des moments ou des intervalles spécifiques.
5. Console-stamp
 - a. Console Stamp permet la mise en place de l'horodatage des messages dans la console, cela permet un historique dans la console afin de savoir à quelle heure une erreur ou une information à été indiquer. Cela permet donc à un technicien d'obtenir l'heure d'un incident ou d'une modification.

Toutes ces librairies sont utilisées constamment dans le script NodeJS, le script nodeJS à été optimisé au maximum afin de garantir une fluidité de fonctionnement et une lisibilité du code.

Afin d'installer ces différentes librairies, la commande suivante doit être effectué:

```
npm install mongodb colors fs node-cron console-stamp
```

⁹ Page par Léo Berteloot

Conception du script NodeJS

Le script NodeJS fonctionne de telle sorte à récupérer les valeurs inscrites dans un fichier JSON par l'Arduino sur le Raspberry et de les insérer chaque minute, dans la base de données MongoDB.

Le script lis un fichier nommée “data.json” grâce à la librairie FS.

```
fs.readFile('data.json', (err, data) => {
  if (err) throw err;
  rpiData = JSON.parse(data);
});
```

Les données récupérées grâce à la librairie FS sont ensuite incluses dans une variable rpiData: le fichier étant à la base, un fichier au format JSON, nous transformons via la méthode “parse” les données dans un tableau, dit “Array”.

Dans ce script NodeJS, l'utilisation de la librairie Node-cron est très importante, elle permet la lecture des valeurs tout les minutes et l'insertion dans le base de donnée par le biais de MongoDB des dernières valeurs.

```
[19:13:59.323] [LOG]  [PROJETBTS] Starting script..
[19:13:59.325] [LOG]  [PROJETBTS] MongoDB Initialized and connected to db.
[19:13:59.334] [LOG]  [PROJETBTS] Table 'capteurs' already exists. Initialization..
[19:14:01.356] [LOG]  [PROJETBTS] Receiving data from Arduino
[19:14:01.357] [LOG]  [PROJETBTS] Data pushed into the database
```

¹⁰ Page par Léo Berteloot

A chaque démarrage du script NodeJS, le script NodeJS vérifie par le biais de MongoDB l'existence de la base de donnée, si cette dernière existe, le script continue son exécution, dans le cas contraire, le script créer la base de donnée et s'arrête afin que l'administrateur relancer le script sans problème.

```
dbo.createCollection("capteurs", function(err, res) {
  if (err) throw err;
  console.log("[PROJETBTS]".cyan + " Table 'capteurs' created ! Please re-launch this app.".green);
  db.close();
});
```

```
[19:11:16.421] [LOG]  [PROJETBTS] Starting script..
[19:11:16.423] [LOG]  [PROJETBTS] MongoDB Initialized and connected to db.
[19:11:16.433] [LOG]  [PROJETBTS] Table 'capteurs' didn't exists. Creating Table...
[19:11:16.567] [LOG]  [PROJETBTS] Table 'capteurs' created ! Please re-launch this app.
```

¹¹ Page par Léo Berteloot

Création du système de pagination

Afin de rendre le site plus facilement accessible, la mise en place d'un système de pagination est nécessaire. Le système de pagination permet dans un premier temps de ranger et de trier les pages des fichiers systèmes grâce à un dossier "pages".

Le système de pagination mis en place permet l'écriture d'url ci-dessous:

<https://central-de-pollution.fr/?p=accueil>

"?p" permet d'indiquer que nous recherchons une page, suivi du nom de la page. Lorsque cette requête est indiqué dans la page, le système de pagination recherche dans le dossier le fichier correspondant au nom de la page demander, si cette dernière n'existe pas ou plus, le système de pagination redirigera sur une page 404: Page introuvable.

Dans le cas contraire, le système de pagination inclura la page demander dans le fichier index.php

Le fichier index.php contient **uniquement** le système de pagination, lorsqu'aucune page est inscrite dans l'URL, le fichier index.php inclut directement la page accueil.

Le code de programmation du système de pagination est disponible en Annexe [A3.2.1](#)

¹² Page par Léo Berteloot

Création de l'interface graphique du site

Mise en place de l'HTML

```
<header class="navbar navbar-dark sticky-top bg-dark flex-mdnowrap p-0 shadow">
  <a class="navbar-brand col-md-3 col-lg-2 me-0 px-3" href="#">Centrale de Pollution</a>
  <button class="navbar-toggler position-absolute d-md-none collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#sidebarMenu" aria-controls="sidebarMenu" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
</header>

<div class="container-fluid">
  <div class="row">
    <nav id="sidebarMenu" class="col-md-3 col-lg-2 p-3 mt-5 d-md-block text-white bg-dark sidebar collapse">

      <div class="dropdown">
        <a href="#" class="d-flex align-items-center text-white text-decoration-none dropdown-toggle" id="dropdownUser1" data-bs-toggle="dropdown" aria-expanded="false">
          
          <strong>admin</strong>
        </a>
        <ul class="dropdown-menu dropdown-menu-dark text-small shadow" aria-labelledby="dropdownUser1" style="">
          <li><a class="dropdown-item" href="#">Mes Paramètres</a></li>
          <li><a class="dropdown-item" href="#">Mon Profil</a></li>
          <li>
            <hr class="dropdown-divider">
          </li>
          <li><a class="dropdown-item" href="logout.php">Déconnexion</a></li>
        </ul>
      </div>
      <hr>

      <ul class="nav nav-pills flex-column mb-auto">
        <li class="nav-item">
          <a href="?p=accueil" class="active nav-link text-white">
            Accueil
          </a>
        </li>
        <li>
          <a href="?p=historique" class="nav-link text-white">
            Historique
          </a>
        </li>
      </ul>
    </nav>
  </div>
</div>
```

La mise en place de l'HTML est très importante, cela permet la mise en page principale. Ici, la création de la barre latérale. Dans cette barre latérale, sont indiquées les différentes pages accessibles par les utilisateurs.

¹³ Page par Léo Berteloot

Mise en place du CSS.

Le CSS permet la mise en forme complète du site, basé sur Bootstrap 5, l'interface se voit être responsive (adapté au téléphone, tablettes..).

Ici, la page de connexion, cela permet de s'authentifier pour accéder à la totalité du site, les administrateurs et utilisateurs normaux se connectent sur la même page. Cela permet d'alléger le nombre de pages du site sans alourdir le nombre de lignes de programmation.

The screenshot shows a login form titled "Centrale de Pollution". The form is labeled "Connexion". It contains two input fields: "Adresse E-mail" and "Mot de passe". Below the inputs is a checkbox labeled "Se souvenir de moi". A blue button labeled "Connexion" is at the bottom left, and a link "Je ne possède pas de compte" is at the bottom right. At the very bottom of the form, there is a copyright notice: "© 2021-2022".

¹⁴ Page par Léo Berteloot

```

/*
 * Sidebar
 */

.sidebar {
    position: fixed;
    top: 0;
    /* rtl:raw:
    right: 0;
    */
    bottom: 0;
    /* rtl:remove */
    left: 0;
    z-index: 100;
    /* Behind the navbar */
    padding: 48px 0 0;
    /* Height of navbar */
    box-shadow: inset -1px 0 0 rgba(0, 0, 0, .1);
}

@media (max-width: 767.98px) {
    .sidebar {
        top: 5rem;
    }
}

.sidebar-sticky {
    position: relative;
    top: 0;
    height: calc(100vh - 48px);
    padding-top: .5rem;
    overflow-x: hidden;
    overflow-y: auto;
    /* Scrollable contents if viewport is shorter than content. */
}

.sidebar .nav-link {
    font-weight: 500;
    color: #333;
}

.sidebar .nav-link .feather {
    margin-right: 4px;
    color: #727272;
}

.sidebar .nav-link.active {
    color: #fff;
}

```

Diverses modifications ont été apportées via un fichier nommée “style.css” afin de rendre l’interface plus conviviale et intuitive.

Par exemple, la barre latérale de navigation à été modifiée afin de la rendre “fixe” et beaucoup plus adaptée pour les téléphones.

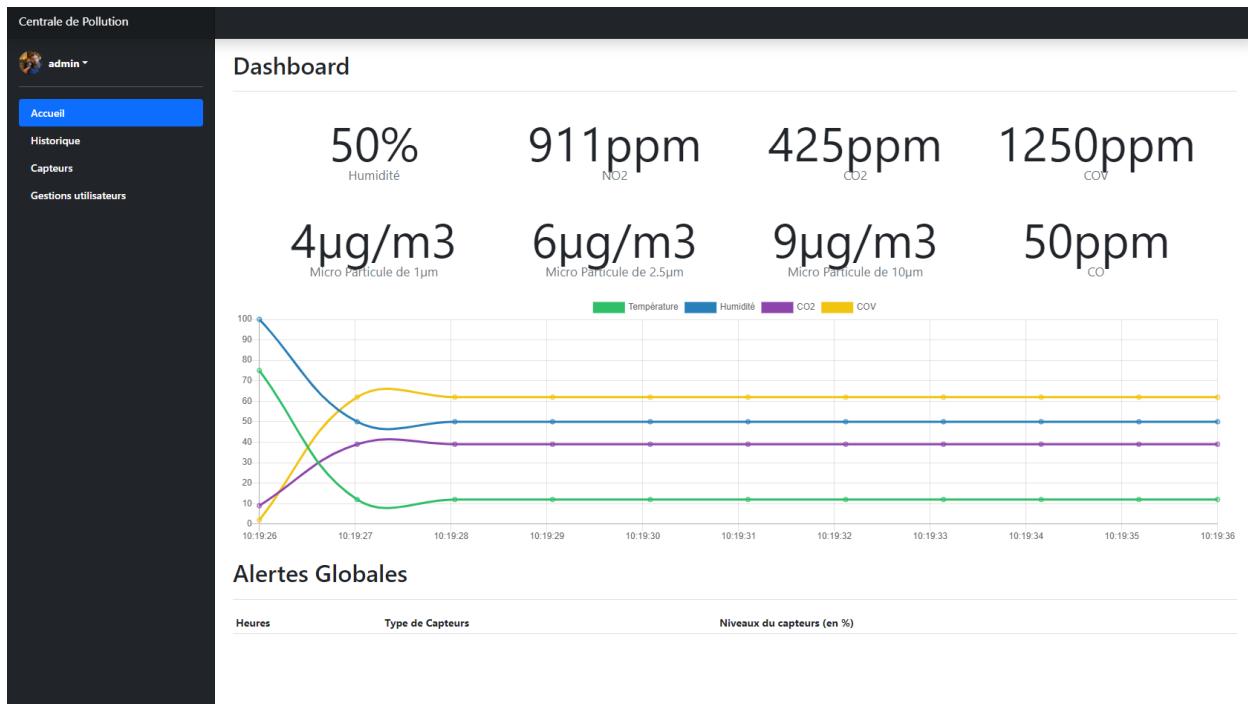
Les modifications apportées par le biais du fichier “style.css” passe **obligatoirement** au-dessus du fichier Bootstrap 5, dans le cas contraire, le fichier “style.css” ne fonctionnerait pas.

15

¹⁵ Page par Léo Berteloot

Conception de l'interface d'accueil

Système d'actualisations des valeurs via AJAX et JQuery



Le système des valeurs ainsi que du diagramme fonctionne par le biais de AJAX et JQuery, toutes les minutes, une requête ajax et effectuer afin de récupérer la valeur la plus récente dans la base de donnée.

Les dernières valeurs sont affichées dans le diagramme utilisant ChartJS, en plus des valeurs actuelles, afficher quant à elles, en haut sous forme de widget.

Lors de chaque récupération des valeurs depuis la requête AJAX, une vérification des limites des capteurs est nécessaire, cela permet, en outre, d'afficher des alertes globales en bas de la page d'accueil mais également de pouvoir afficher une notification sur le site.

¹⁶ Page par Léo Berteloot

Système de notification via popup (ToastJS)

Le système de notification utilise la librairie Javascript ToastJS. Cela permet d'afficher une petite notification en haut à droite de la page lorsqu'une valeur atteint la limite.

Ce système est opérationnel sur chacune des pages du site afin de permettre l'information en temps réel.

```
<script>
var checkAnomalie = function(){
    $.ajax({
        url: 'includes/receiveLastData.php',
        success: function(data){
            var LastData = $.parseJSON(data);

            if(LastData.humidite == 100 || LastData.NO2 >= 1479 || LastData.CO2 >= 1079 || LastData.CO >= 2008 || LastData.CO >= 700){
                toastr["error"]("Anomalie détecter sur un capteur !");
            }
        });
}
checkAnomalie();
setInterval(checkAnomalie, 12500);

var getNewData = function(){
    $.ajax({
        url: 'includes/getLastData.php',
        success: function(data){
            if(data == false){
                toastr["error"]("Erreur lors de la récupération des données");
            }
        });
}
getNewData();
setInterval(getNewData, 1000);
```

¹⁷ Page par Léo Berteloot

Partie de Guilbert Romain : Partie utilisateur du site et Capteur d'humidité et de particules

Ma partie consisté à récupérer les données des capteur de particule fine

C'est donné sont utilisés pour la base de données qui seront visibles par l'utilisateur sur le site

Pour ceci j'ai eu en ma disposition 2 capteur et le logiciel Arduino

La partie utilisateur elle sera utilisée pour que celui ci aille voir les données après s'être créé un compte

¹⁸ écrit par Romain Guilbert

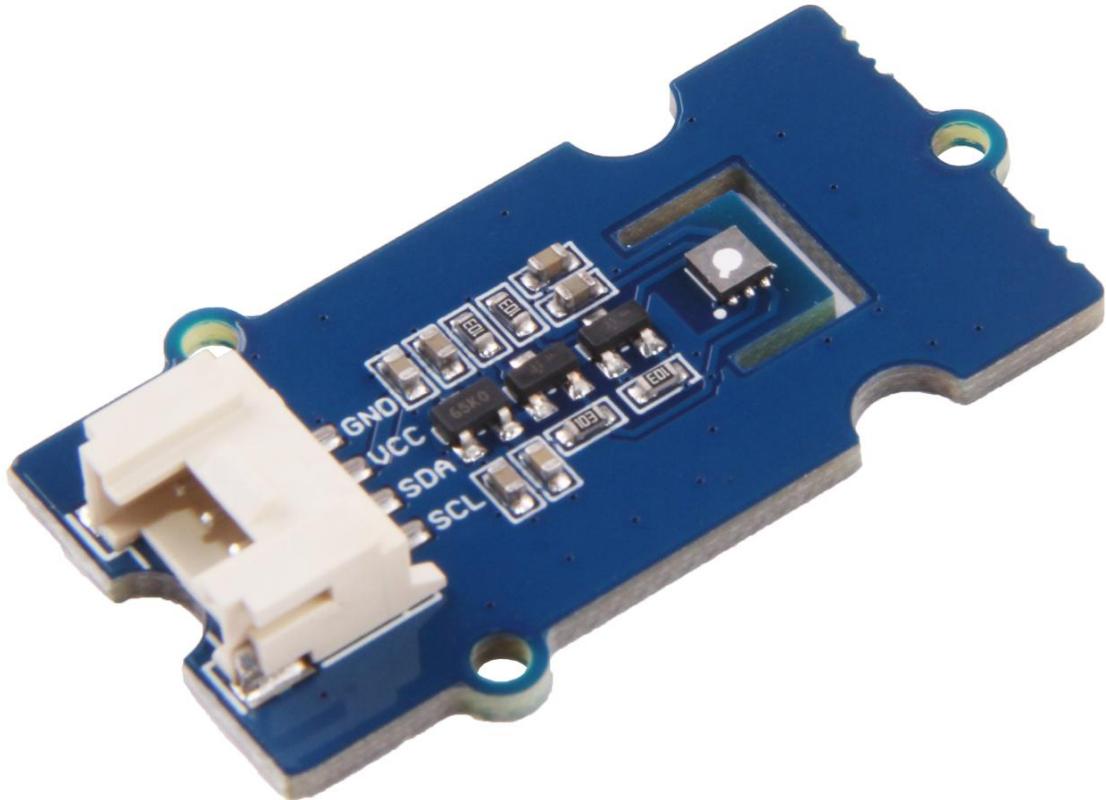
Partie de Dabach Achraf : Liaison Raspberry - Arduino et base de donnée & Capteur de COV et CO2 :

Parmis les différentes tâches de travail qu'il y avait à réaliser, celles qui m'ont été attribué étaient de m'occuper de la partie liaison arduino raspberry, de la base de données ainsi que du capteur COV (Composés Organiques Volatiles) et CO2 (Dioxyde de carbone).

Les Composés Organiques Volatiles regroupent plusieurs milliers de composés (hydrocarbures, solvants, ...) aux caractéristiques très variables. Ils ont un impact direct sur la santé (certains sont toxiques ou cancérogènes). Ce sont des gaz et des vapeurs qui contiennent du carbone, comme les vapeurs d'essence et des solvants. Ils interviennent dans le processus de formation d'ozone dans la basse atmosphère et participent donc au réchauffement de la planète. Les Composés Organiques Volatils (COV) entrent dans la composition des carburants mais aussi de nombreux produits courants: peintures, encres, colles, détachants, cosmétiques, solvants...pour des usages ménagers, professionnels ou industriels (pour ces raisons, leur présence dans l'air intérieur peut aussi être importante). Les effets de COV sont très variables selon la nature du polluant envisagé. Ils vont d'une certaine gêne olfactive à des effets mutagènes et cancérogènes, en passant par des irritations diverses et une diminution de la capacité respiratoire.

Le dioxyde de carbone, également appelé gaz carbonique, de formule moléculaire CO_2 , est un gaz incolore. Il se compose de deux atomes d'oxygène et d'un atome de carbone. L'impact du CO_2 sur la santé et l'efficacité est réel, tout particulièrement au sein des bâtiments. D'où l'importance de surveiller la qualité de l'air intérieur. Il faut dire que l'impact du CO_2 sur la santé n'est pas un sujet majeur, notamment car cette molécule n'est pas chimiquement毒ique. Pour preuve, les êtres vivants en émettent naturellement lors de la respiration. Pourtant, les cas d'intoxication au dioxyde de carbone ne sont pas inexistant.

Pour les Composés Organiques Volatiles (COV) et le Dioxyde de carbone (CO2) un capteur s'occupe de mesurer les deux différentes valeurs. C'est le Grove VOC and eCO2 Gas Sensor(SGP30) :



Voici les différentes caractéristiques des capteurs sur un tableau :

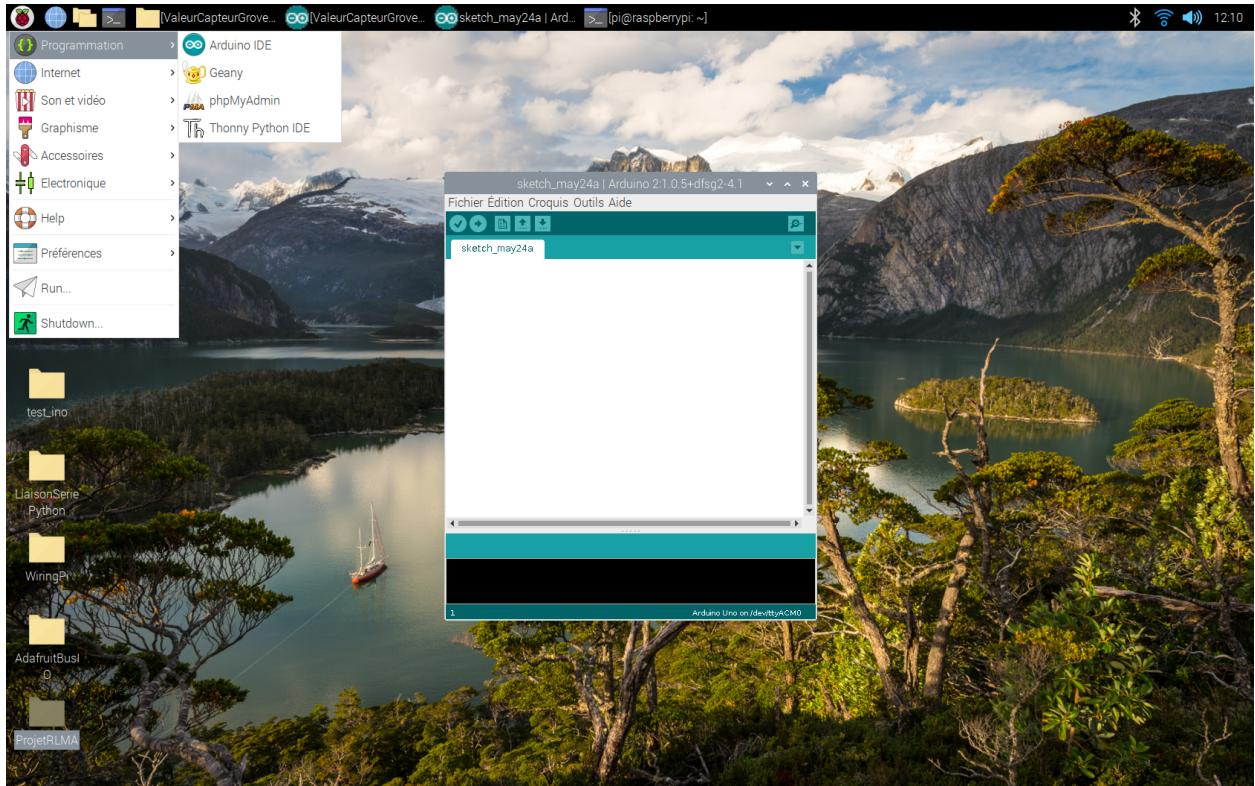
| Paramètre | Signal | Valeur |
|-----------------------|--------|---------------------|
| Tension d'utilisation | VOC | 0 ppb à 60000 ppb |
| Plage de sortie | CO2 | 400 ppm à 60000 ppm |

Voici le coût ainsi que le matériel totale pour mesurer le cov et le eco2 :

| Description | Quantité | Prix |
|---------------------------|----------|--------|
| Raspberry Pi 3B+ | 1 | 75€ |
| Arduino uno | 1 | 20€ |
| Capteur SGP 30 (COV eCO2) | 1 | 17.50€ |
| Câbles | 1 | 3.20€ |
| | Total | 115.7€ |

Dans un premier temps j'ai branché ma raspberry a un écran pour pouvoir y accéder et télécharger arduino.





Ensuite l'étape suivante était de pouvoir connecter un arduino au raspberry et de créer un code pour récupérer les valeurs d'un capteur. Etant donné qu'au début du projet mon Capteur SGP 30 (COV eCO2) n'était pas encore disponible, j'ai donc choisis un capteur utile au détecteur de fuite de gaz à la maison ou bien en usine en me disant que plus tard il pourrait nous servir dans notre projet.

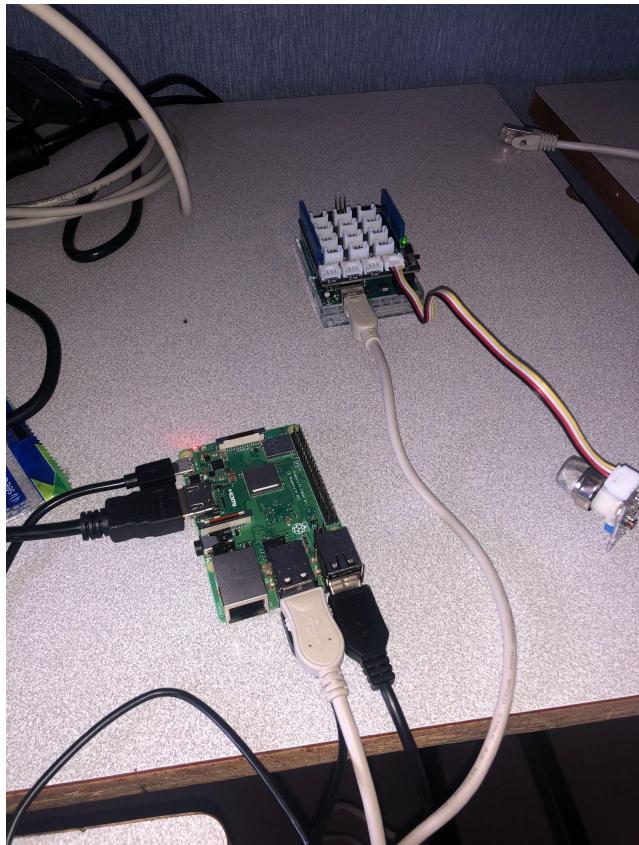
Voici le capteur :

Grove - Gas Sensor(MQ5)



Le module Grove - Capteur de gaz (MQ5) grâce à sa haute sensibilité et à son temps de réponse rapide, les mesures peuvent être prises le plus rapidement possible. La sensibilité du capteur peut être réglée à l'aide du potentiomètre.

Voici le montage et la connexion du capteur l'arduino ainsi que l'arduino connecter au raspberry:



Ceci est le code permettant de récupérer les valeurs du capteur sur la raspberry

```
ValeurCapteurGroveGasMQ5_COV
void setup () {
    Serial.begin(9600);
}

void loop()
{
    float sensor_volt;
    float sensorvalue;
    const int Moncapteur = 0;

    sensorValue = analogRead(MonCapteur);
    sensor_volt = sensorValue/1024*5.0;

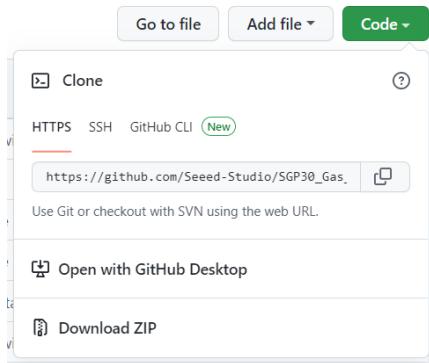
    Serial.print("cov=");
    Serial.print(sensorValue);
    Serial.print(".");
    // Serial.print(" sensor_volt = ");
    // Serial.print(sensor_volt);
    //Serial.println(" V");
    delay (3000);
}
```

Une fois les valeurs bien recuperer la suite logique est de pouvoir lire les valeurs sur le port série pour pouvoir les stockées dans un fichier json sur la raspberry qui servira au site web.

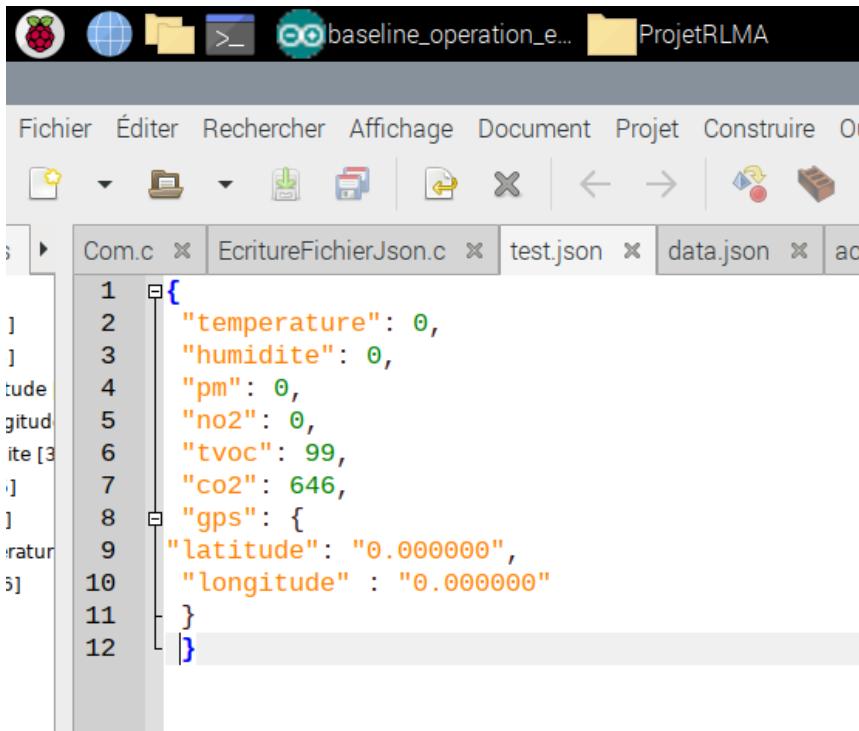
Pour cela j'ai donc créé un script en C++ qui va lire les valeurs arrivant sur le port série et qui les écrit dans un fichier json (Voir annexes **A3.2.1 - C++ connexion série; A3.2.2 - Fichier Json**).

Après avoir corrigé les bugs pouvant exister et avoir reçu mon Capteur SGP 30 (COV eCO2) je me met à rechercher les différentes bibliothèques arduino pour pouvoir le faire fonctionner et reproduire le même schéma d'action qu'avec le Capteur de gaz Grove - (MQ5).

Bibliothèques nécessaire :



```
pi@raspberrypi:~/Desktop/ProjetRLMA $ ./a.out
*tvoc = 0          co2 = 400
tvoc = 21         co2 = 400
tvoc = 32         co2 = 400
tvoc = 59         co2 = 425
tvoc = 64         co2 = 522
tvoc = 99         co2 = 646
^C
pi@raspberrypi:~/Desktop/ProjetRLMA $
```



The screenshot shows a terminal window with the following details:

- File navigation icons: Raspberry Pi, globe, folder, back, forward.
- Project name: baseline_operation_e... ProjetRLMA
- Menu bar: Fichier, Éditer, Rechercher, Affichage, Document, Projet, Construire, Outils.
- Toolbar: New, Open, Save, Print, Close, Undo, Redo, Copy, Paste, Find, Replace.
- Tab bar: Com.c, EcritureFichierJson.c, test.json, data.json, ac.
- Code editor content (test.json):

```
1 {  
2     "temperature": 0,  
3     "humidite": 0,  
4     "pm": 0,  
5     "no2": 0,  
6     "tvoc": 99,  
7     "co2": 646,  
8     "gps": {  
9         "latitude": "0.000000",  
10        "longitude": "0.000000"  
11    }  
12 }
```

Les valeurs sont bien récupérées et transmises au fichier json comme voulu.

Devis final

On peut établir un devis rassemblant tout le matériel à commander :

| Description | Quantité | Prix unitaire HT | Prix total HT |
|---|----------|------------------|---------------|
| Raspberry Pi 3B+ | 1 | 75,00 € | 75,00 € |
| Arduino Uno | 1 | 20,00 € | 20,00 € |
| MCP 9808 Capteur de température | 1 | 5,00 € | 5,00 € |
| GPS Grove | 1 | 25,00 € | 25,00 € |
| Antenne GPS | 1 | 5,00 € | 5,00 € |
| Capteur humidité | 1 | 12,70 € | 12,70 € |
| Capteur Laser PM2.5 HM3301 Capteur de micro-particules | 1 | 32,90 € | 32,90 € |
| Capteur SGP30 Capteur COV et eCO2 | 1 | 17,50 € | 17,50 € |
| Câbles Câbles Grove de 20cm, 5 par pack | 1 | 3,20 € | 3,20 € |

Total HT

196,30 €

Annexe A1 - Documentation client

Annexe A2 - Documentation du matériel utilisé

A2.1 - Thermomètre de Précision

Grove - I2C High Accuracy Temperature Sensor (MCP9808)

https://wiki.seeedstudio.com/Grove-I2C_High_Accuracy_Temperature_Sensor-MCP9808/

A2.2 - Capteur de température et d'Humidité

Grove - Temperature & Humidity Sensor (High-Accuracy & Mini) v1.0

https://wiki.seeedstudio.com/Grove-TemptureAndHumidity_Sensor-High-Accuracy_AndMini-v1.0/

A2.3 - Capteur de gaz : eCO2 et VOC

Grove - VOC and eCO2 Gas Sensor (SGP30)

https://wiki.seeedstudio.com/Grove-VOC_and_eCO2_Gas_Sensor-SGP30/

A2.4 - Capteur laser pour micro-particules

Grove - Laser PM 2.5 Sensor (HM3301)

https://wiki.seeedstudio.com/Grove-Laser_PM2.5_Sensor-HM3301/

A2.5 - Capteur de qualité de l'air

Grove - Air Quality Sensor v1.3

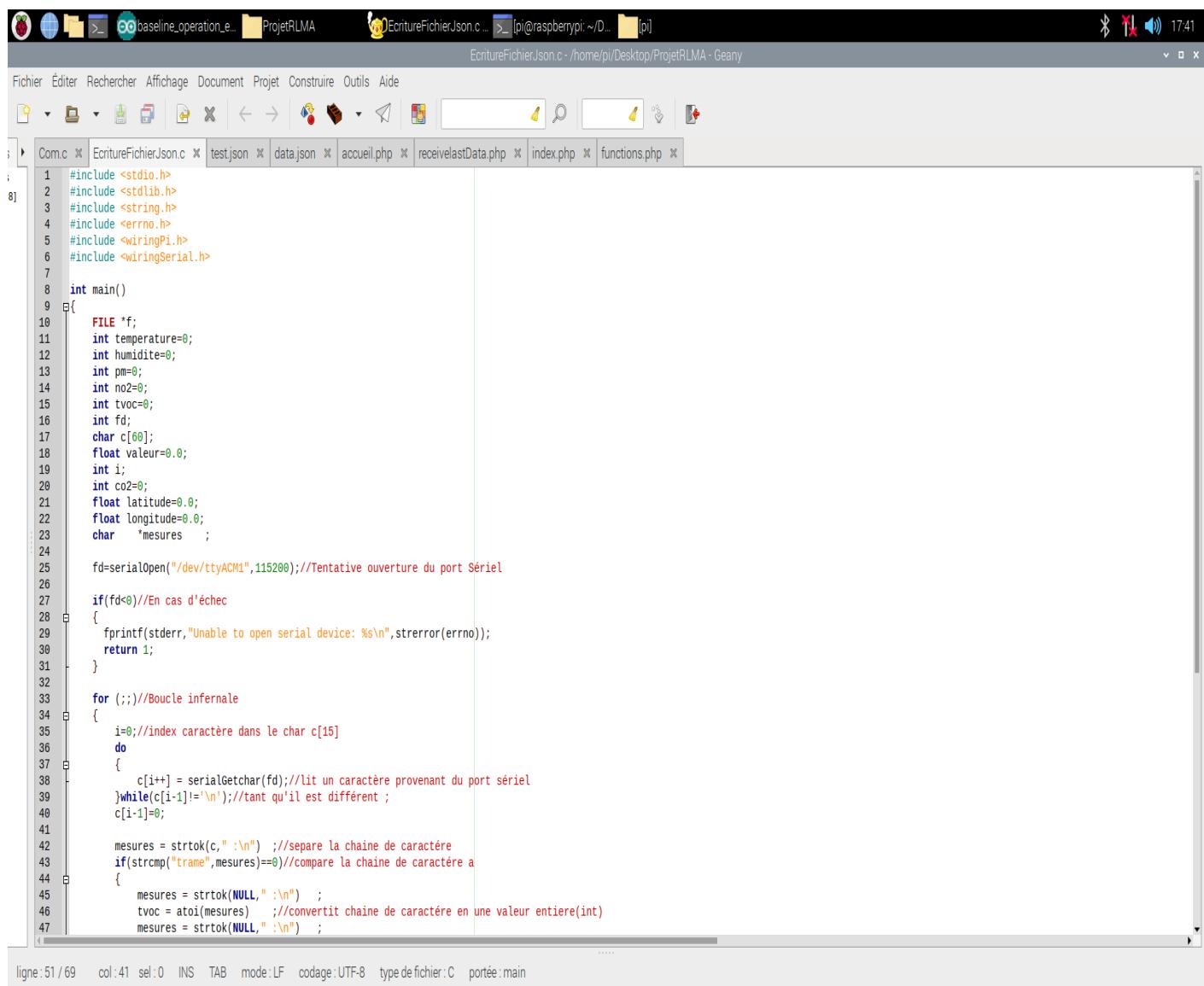
https://wiki.seeedstudio.com/Grove-Air_Quality_Sensor_v1.3/

Annexe A3 - Codes programmation

A3.1 - Arduino

A3.2 - Raspberry

A3.2.1 - C++ connexion série



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

int main()
{
    FILE *f;
    int temperature=0;
    int humidite=0;
    int pm=0;
    int no=0;
    int tvoc=0;
    int fd;
    char c[60];
    float valeur=0.0;
    int i;
    int co2=0;
    float latitude=0.0;
    float longitude=0.0;
    char *mesures = NULL;

    fd=serialOpen("/dev/ttyACM1",115200); //Tentative ouverture du port Série

    if(fd<0)//En cas d'échec
    {
        fprintf(stderr,"Unable to open serial device: %s\n",strerror(errno));
        return 1;
    }

    for ();//Boucle infernale
    {
        i=0;//Index caractère dans le char c[15]
        do
        {
            c[i++]= serialGetchar(fd); //lit un caractère provenant du port série
        }while(c[i-1]!='\n');//tant qu'il est différent ;
        c[i-1]=0;

        mesures = strtok(c, "\r\n"); //separe la chaîne de caractère
        if(strcmp("trame",mesures)==0)//compare la chaîne de caractère à
        {
            mesures = strtok(NULL, "\r\n");
            tvoc = atoi(mesures) ;//convertit chaîne de caractère en une valeur entière(int)
            mesures = strtok(NULL, "\r\n");
        }
    }
}
```

ligne: 51 / 69 col: 41 sel: 0 INS TAB mode:LF codage:UTF-8 type de fichier:C portée:main

```

1 // Ecrire dans un fichier json
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <errno.h>
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

The screenshot shows the Geany IDE interface with multiple tabs open. The active tab is 'EcritureFichierJson.c'. The code in the editor is a C program that opens a serial port, reads data from it, and writes it to a JSON file named 'test.json'. The code includes error handling for the serial port opening and reading, and it uses strtok and atoi functions to parse the incoming data into integers.

A3.2.2 - Fichier Json

```

1 {
2     "temperature": 0,
3     "humidite": 0,
4     "pm": 0,
5     "no2": 0,
6     "cov": 144.000000,
7     "co2": 0,
8     "gps": {
9         "latitude": "0.000000",
10        "longitude": "0.000000"
11    }
12 }

```

The screenshot shows the Geany IDE interface with multiple tabs open. The active tab is 'test.json'. The content of the file is a JSON object with several properties: temperature, humidite, pm, no2, cov, co2, and gps. The gps property is an object containing latitude and longitude values. The JSON is displayed with syntax highlighting.

A3.2.3 - NodeJS

```
/* Initialisation des Modules Node */
var MongoClient = require("mongodb").MongoClient;
const colors = require('colors');
const random = require('random');
const fs = require('fs');
const cron = require('node-cron');
require('console-stamp')(console, 'HH:MM:ss.l');
var url = "mongodb://127.0.0.1:27017/";
/* Connexion à la BDD de MongoDB */
MongoClient.connect(url, (err, db) => {
    if(err) throw err;
    var dbo = db.db("projetsbts");
    console.clear();
    if(!err){ console.log("[PROJETBTS]".cyan+" Starting script..".magenta); }
    console.log("[PROJETBTS]".cyan+" MongoDB Initialized and connected to db.".green);
    dbo.listCollections({name: "capteurs"}).next((err, exists) => {
        if(err) console.log(err);
        if (exists) {
            console.log("[PROJETBTS]".cyan+" Table 'capteurs' already exists. Initialization..".yellow);
            cron.schedule('1 * * * *', () => {
                fs.readFile('rpi.json', (err, data) => {
                    if(err) throw err;
                    rpiData = JSON.parse(data);
                });
                var ping = random.int(10, 100);
                console.log("[PROJETBTS]".cyan+" Receiving data from".green+".magenta+" `${ping}ms` ".yellow);
                var d = new Date().toISOString().replace(/T/, ' ').replace(/\..+/, '');
                var dataToPush = { date:d, temperature: rpiData.temperature, humidite: rpiData.humidite, pm: rpiData.pm };
                dbo.collection("capteurs").insertOne(dataToPush, function(err, res) {
                    if (err) throw err;
                    console.log("[PROJETBTS]".cyan+" Data pushed into the ".yellow+"database".magenta);
                });
            });
        }else{
            console.log("[PROJETBTS]".cyan+" Table 'capteurs' didn't exists. Creating Table...".brightRed);
            dbo.createCollection("capteurs", function(err, res) [
                if (err) throw err;
                console.log("[PROJETBTS]".cyan+" Table 'capteurs' created ! Please re-launch this app.".green);
                db.close();
            ]);
        }
    });
});
```

A3.2 - Site Web

A3.2.1 - Système de pagination

```
<div class="row">
    <?php require_once('includes/sidebar.php'); ?>

    <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4">
        <?php
            if(!empty($_GET['p'])){
                if(file_exists("pages/".$_GET['p'].'.php')){
                    include('pages/'.$_GET['p'].'.php');
                }else{
                    include('pages/404.php');
                }
            }else{
                include('pages/accueil.php');
            }

        ?>
    </main>
</div>
```

A3.2.1 - Accueil