

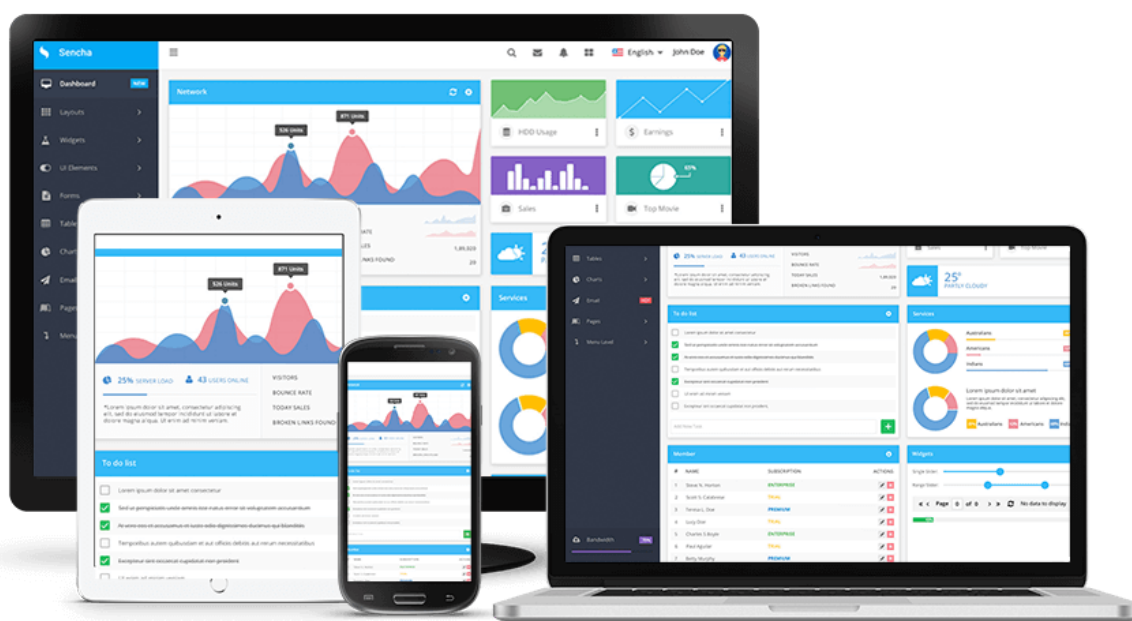
Dossier technique du projet- Partie personnelle

Etudiant 2 : Application Web – Visualiser l'évolution de chaque mesure avec choix de période

Acquérir la mesure de l'intensité lumineuse

Etude et configuration réseau des différents matériels

Mise en place d'une boucle de courant 4-20 mA

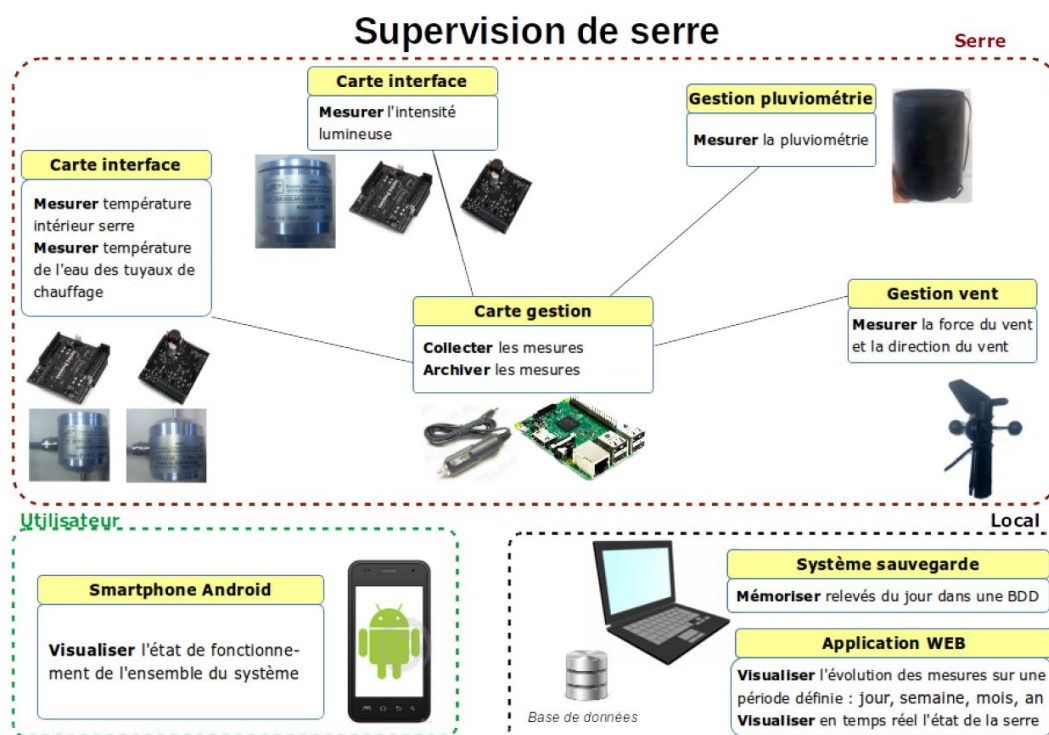


SOMMAIRE

1. SITUATION DANS LE PROJET	2
1.1) SYNOPTIQUE DE LA REALISATION	2
1.2) RAPPEL DES TACHES DE L'ETUDIANT	3
1.3) CONTRAINTES DE REALISATION	4
1.4) PROBLEME MATERIEL	4
2. CONCEPTION ET MISE EN ŒUVRE	5
2.1) FONCTIONNEMENT DU SOLARIMETRE	5
2.2) FONCTIONNEMENT DE LA BOUCLE 4-20 MA.	5
2.3) REALISATION DU DIAGRAMME DE CLASSE	6
3. ETUDE ET CONFIGURATION RESEAU	7
3.1) ARDUINO A LA RASPBERRY	7
3.2) APPLICATIONS A LA BASE DE DONNEES	8
4. RECUPERATION ET ENVOIE DES DONNEES	9
4.1) CHOIX DU SHIELD ARDUINO POUR LA BOUCLE DE COURANT	9
4.2) MISE EN PLACE DE LA BOUCLE 4-20 MA	10
4.3) TEST DU SOLARIMETRE	10
4.4) DETECTION DU PROBLEME	12
4.5) RECUPERER LES DONNEES D'UN CAPTEUR SUR LA BOUCLE 4-20 MA	13
5. MISE EN PLACE DE L'APPLICATION WEB	15
5.1) CONCEPTION DE LA CHARTE GRAPHIQUE	15
5.2) ARCHITECTURE DE L'APPLICATION	17
5.3) CONNEXION A LA BASE DE DONNEES	18
5.4) GESTION DE LA PERIODE	19
5.5) AFFICHAGE D'UN GRAPHIQUE	20
5.5.1) <i>Requêtes pour récupérer les données du graphique</i>	20
5.5.2) <i>Insérer des données au graphique</i>	22
5.6) CREATION D'UNE PAGE POUR AJOUTER UN CAPTEUR	22
5.7) MODIFICATION QUE J'AIMERAI APPORTER	23
6. TESTS UNITAIRES	24
6.1) TEST UNITAIRE DE LA METHODE GET_TEMPERATURE_AIR()	24
7. CONCLUSION	26
8. ANNEXES	27
ANNEXE 1 – DIAGRAMME DE GANTT	27
ANNEXE 2 – CONCEPTION DE LA BASE DE DONNEES	28
ANNEXE 3 – MANUEL UTILISATEUR DE LA PAGE EVOLUTION_MESURE.PHP	29

1. Situation dans le projet

1.1) Synoptique de la réalisation



Au sein du projet, j'ai eu pour tâche de mettre en place le capteur de l'intensité lumineuse pour .. et la création de l'application Web, comprenant le design du site ainsi que la page pour visualiser l'évolution des mesures sur une période définie.

1.2) Rappel des tâches de l'étudiant

Dans ce projet de supervision d'une serre, j'avais pour but de mettre en place un capteur pour l'intensité lumineuse, qui a pour but d'automatiser des stores intérieurs lors de la deuxième année. En effet, les stores assureront un ombrage adéquat pour éviter une forte oscillation des températures. Je devais aussi étudier et configurer le réseau des différents matériels pour pouvoir faire dialoguer tous les systèmes entre eux, pour ensuite me consacrer à l'application Web. Celle-ci permettra à l'utilisateur de voir dans une période initialement choisie les variations des différentes données réalisées par les capteurs.

Dans un premier temps, je me suis concentré sur le capteur, pour savoir comment il communiquait, comment le mettre en place, ainsi que sur la boucle de courant 4-20mA (boucle qui permet de transmettre un signal analogique sur une grande distance sans modifier ou perdre ce signal). Pour assurer cette boucle de courant, nous devons choisir avec l'étudiant

Je me suis ensuite intéressé à la connexion entre la carte Arduino, à laquelle nous avons ajouté un Shield pour qu'elle soit rattachée à la boucle de courant, et la Raspberry Pi 3, qui est la carte de gestion. J'ai donc utilisé la librairie fournie par le Shield pour pouvoir utiliser correctement la boucle de courant et pouvoir communiquer avec la carte de gestion.

Et ensuite, nous avons réalisé avec Samuel, l'étudiant 3 l'application Web. Nous avons choisi de partir sur un design proche du site qu'ils utilisent actuellement, pour faciliter la prise en main du superviseur.

Ainsi, la réalisation de ces tâches a été effectuée en trois grandes étapes (spécifications, analyse, conception) qui suivent le diagramme de Gantt présent en annexe 1.

1.3) Contraintes de réalisation

Dans un premier temps, nous avons une **contrainte financière**. Nous avons donc un budget alloué de 100 euros permettant l'achat d'une carte adaptateur 4-20mA (Shield pour Arduino). Donc nous avons eu à choisir un shield permettant de lire plusieurs canaux, car nous avons initialement 3 capteurs à intégrer à la boucle de courant.

Ensuite, la **contrainte de développement** nous a fait réaliser l'application Web sous le patron Modèle-Vue-Contrôleur dans l'environnement de développement NetBeans sous Windows. Le framework Symfony aurait aussi pu être utilisé, cependant, nous avons choisi de ne pas l'utiliser car les requêtes que nous utilisons restent assez basique.

Et pour terminer, nous avons plusieurs **contraintes de qualité**. La première est une contrainte d'évolutivité forte, ainsi, lorsque l'utilisateur voudra ajouter un capteur, ou une mesure, le travail à réaliser de son côté doit-être minime, voir automatique. Une documentation complète sur le système doit être fournie au client, pour qu'une fois le projet terminé, une autre équipe que l'équipe d'étudiant puisse donner suite à ce projet.

Ce projet va être réaliser sur deux années :



- La première année, où nous sommes, nous nous occupons de la supervision de la serre, qui comprend la récupération des données des capteurs, l'enregistrement des données dans une base de données, l'application Web ainsi que l'application Android.
- La deuxième année se penchera sur l'automatisation de la régulation des différentes données tels que la température, ou encore l'hydrométrie.

1.4) Problème matériel

Lors d'une phase de test, je me suis aperçu que le solarimètre en notre possession ne fonctionnait plus, un solarimètre en notre possession n'étant plus en vente, et notre budget alloué pas assez élevé, je me suis occupé du capteur mesurant la température intérieure. Cependant, vu que mes recherches étaient portées sur le solarimètre, l'ensemble du dossier technique comportera mon analyse à son propos.

2. Conception et mise en œuvre

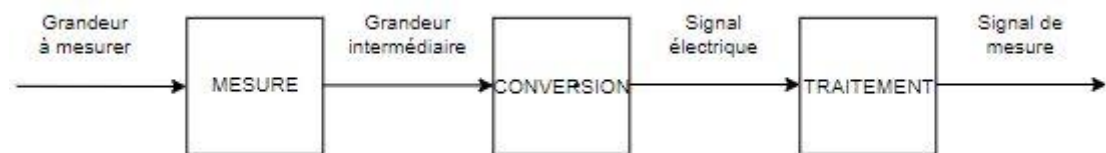
2.1) Fonctionnement du solarimètre

Le solarimètre est un capteur industriel, il doit donc être alimenté pour pouvoir fonctionner. Grâce à ses deux câbles, bleu et blanc, respectivement le plus et le moins. Les mesures du solarimètre vont de 0 à 1000 W/m².

Le fonctionnement du solarimètre assure plusieurs fonctions :

- Il dimensionne la grandeur à mesurer, car un capteur est avant tout un appareil de mesure.
- Il convertit la mesure en un signal analogique.
- Il émet un signal standard de la grandeur à mesurer.

Le principe du capteur peut ainsi être représenté par le schéma fonctionnel suivant :



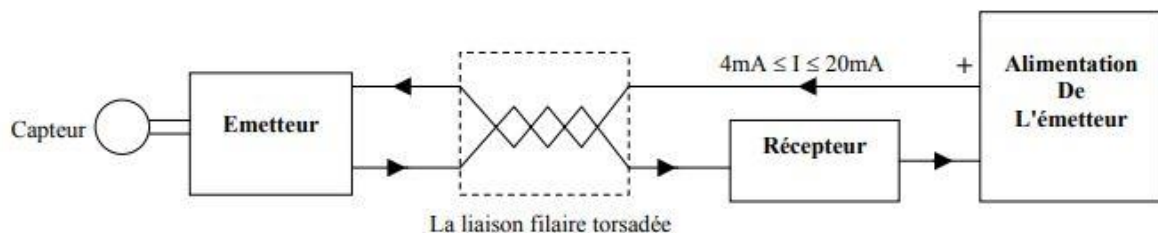
2.2) Fonctionnement de la boucle 4-20 mA.

Inventée vers 1930, par un ingénieur du groupe ESSO aux Etats-Unis, ce procédé est destiné à transmettre un signal analogique à quelques dizaines ou centaines de mètres. Il repose sur le constat que le long d'un câble, aussi long soit-il, le courant continu qui le traverse est constant.

L'idée est de réaliser un dispositif, capteur et circuit associé, dont la consommation en mA sera proportionnelle à la tension que l'on devrait mesurer aux bornes du capteur et de faire en sorte que celle-ci se situe dans la plage 4-20mA, ces limites correspondant alors aux limites d'utilisation du capteur, c'est-à-dire que si un capteur a pour plage de données 0 à 50°C, la valeur 0°C sera interprétée par un courant électrique de 4 mA.

On aurait pu choisir 0-20mA mais ceci peut être problématique en cas de dérive qui décale le courant vers les valeurs négatives, la plage 0-4mA constitue donc une marge de sécurité. De plus, le fait de retenir la plage 4-20mA permet de détecter un défaut dans la boucle si le courant devient nul.

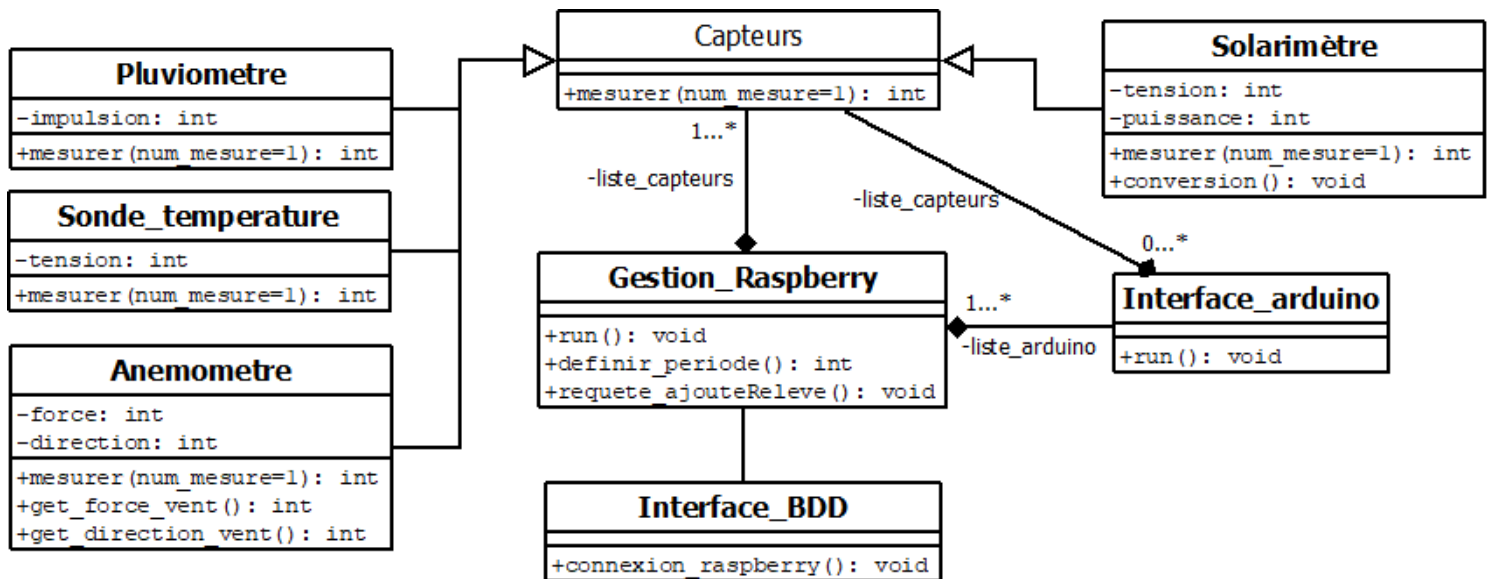
Le principe de la boucle de courant peut être représenté par le schéma fonctionnel suivant :



Pour réaliser cette boucle de courant, il faut 4 éléments principaux minimum :

- Le capteur va mesurer des grandeurs physiques et délivrer une tension de faible amplitude.
- L'émetteur convertit la valeur mesurée par le capteur en un courant compris dans l'intervalle 4-20mA.
- L'alimentation de l'émetteur est une alimentation externe de 24V pour pouvoir
- Le récepteur sera ici une carte de gestion, qui enregistrera les données dans une base de données.

2.3) Réalisation du diagramme de classe



Ce diagramme de classe a été fait en groupe. En mettant en commun au fur et à mesure du projet, le diagramme de classe a connu quelques changements. Le diagramme ci-dessus est la version finale. Les classes présentes dans ce diagramme de classe sont les classes de la carte de Gestion, et donc compose le programme Python que Steven a réalisé.

Dans ce diagramme de classe, je ne serai lié qu'à quelques classes.

La classe Capteurs :

C'est la classe mère de la classe **Solarimètre**. C'est une classe abstraite car elle contient la méthode virtuelle pure *mesurer()*. Ce qui signifie qu'elle est déclarée mais non définie dans la classe **Capteurs**. Cette méthode doit donc être définie dans les classes filles de la classe **Capteurs**.

La classe Solarimètre :

C'est une classe fille de la classe **Capteurs** car elle hérite de celle-ci. La classe **Solarimètre** contient ainsi la méthode *mesurer()*, ainsi, dans cette méthode, je devais étalonner les valeurs du solarimètre pour ensuite envoyer une valeur réelle et non une tension dans la base de données.

3. Etude et configuration réseau

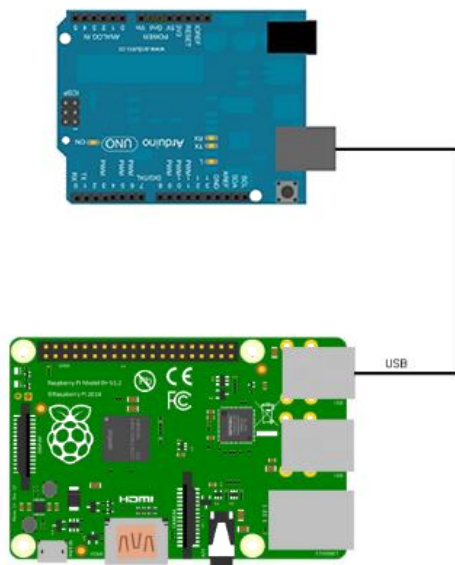
3.1) Arduino à la Raspberry

Afin de pouvoir dialoguer directement de la Arduino à la Raspberry, il a été nécessaire de réfléchir au moyen utiliser pour pouvoir utiliser la connexion entre les deux cartes. C'est alors que plusieurs choix étaient possibles.

Des connexions sans fils, et des connexions filaires.

Dans un premier temps, étudions les connexions sans fils qui n'ont pas été retenues. En effet, pour le Bluetooth le choix n'était pas possible, car il aura fallu rajouter un Shield à la Arduino, qui elle est possède déjà un. Ainsi, ce choix a été le premier à être rejeté. Nous avons aussi le Wi-Fi qui aurait été une bonne alternative au Bluetooth, cependant, celui-ci n'a pas été choisi car ne sachant pas où allaient se trouver les capteurs par rapport à la carte de Gestion, il était impossible de savoir si la plage du Wi-Fi allait pouvoir recouvrir l'ensemble du système.

Ensuite, les connexions filaires, deux choix étaient possibles. Nous avons le choix entre le bus de série TX/RX et le port USB. Pour bien savoir comment s'est porté ma décision, il me fallait plus de connaissance sur le bus TX/RX. Le bus TX/RX est en réalité comme le port USB, cependant, il ne nécessite pas d'encapsulation, qui fera perdre du temps à un transfert de données. Or, pour notre projet, le temps d'envoi des données ne nécessite pas un transfert de données élevé, environ 2 données toutes les 30 minutes. C'est pourquoi j'ai choisi de faire communiquer la Arduino à la Raspberry en USB.



Liaison Arduino à Raspberry

3.2) Applications à la base de données

Pour que les applications se connectent à la base de données, il a été décidé en groupe d'héberger la base de données sur un serveur distant, pour deux raisons :

- Nous voulions que l'application Androïd soit accessible de n'importe quel endroit. En effet, même lorsque le superviseur n'est pas sur son poste de travail, il pourra regarder de chez lui l'état, fonctionnels ou non, des capteurs.
- L'application Web qui fonctionne en local pourra être utilisée sur plusieurs PC, par exemple, si l'ordinateur de supervision ne fonctionne plus, il sera possible de le changer et de ne rien modifier à la structure de la base de données, ainsi qu'à l'application en elle-même.

4. Récupération et envoi des données

La partie concernant la boucle de courant 4-20 mA a été faite avec Samuel, l'étudiant 3.

4.1) Choix du Shield Arduino pour la boucle de courant

Pour le choix du Shield Arduino, nous étai imposé un budget de 100€ maximum, c'est ce qui a fait ressortir deux adaptateurs :

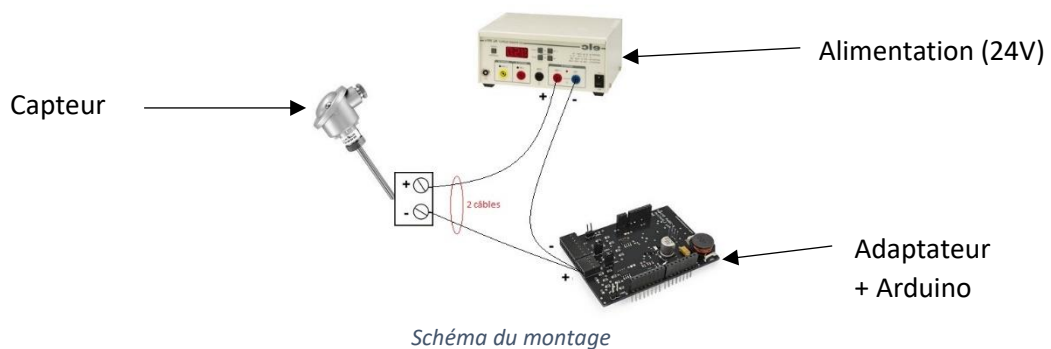
Adaptateur 4-20 mA 1132_0	Adaptateur 4-20 mA Current Loop Sensor Board
	
Nombre de canaux : 1	Nombre de canaux : 4
Livré avec câble de raccordement.	Livré avec câbles en paire torsadée.
Température de service : -40°C à +85°C	Température de service : 0°C à +85°C
Dimensions : 46 x 30 x 18 mm.	Dimensions : 73 ; 5 x 51 x 13 mm.
Module prêt à l'emploi.	Module prêt à l'emploi.
Prix : 34€50 + 5€90 pour la livraison	Prix : 78€00

L'adaptateur numéro 1 possède 1 canal, donc il aura fallu en prendre 3, avec les frais de livraison, la contrainte budgétaire aurait été dépassée (113.40€). De plus, pour faire fonctionner cet adaptateur, il aurait fallu rajouter une carte d'interface 8/8/8 qui aurait rajouté au prix initial environ une centaine d'euros.

Notre choix s'est ainsi porté sur la carte numéro 2, dont le prix est moins élevé, de plus, elle possède 4 canaux. Ayant 3 capteurs sur la boucle de courant 4-20mA, ce choix était le mieux adapté.

4.2) Mise en place de la boucle 4-20 mA

Pour mettre en place la boucle de courant, nous devons dans un premier temps attendre le délai de livraison de l'adaptateur 4-20 mA Current Loop Sensor Board. Une fois l'adaptateur reçu, nous l'avons placé sur la carte Arduino Mega 2560 comme précisé sur la documentation où l'adaptateur a été acheté. Nous avons reproduit la boucle de courant en suivant le schéma fonctionnel de celle-ci.



Pour ensuite tester la boucle, nous avons inséré un programme dans la carte Arduino pour récupérer le voltage. Pour ce faire, nous avons utilisé la bibliothèque fournie avec l'adaptateur.

```
float currentLoop::readVoltage(uint8_t channel)
{
    return (readChannel(channel) * 50. / 1000);
}
```

Il nous suffisait ainsi de modifier le channel (1,2,3 ou 4) avec celui qui correspondait, et on s'est aperçu que les valeurs variaient en fonction des modifications effectuées aux capteurs. Malheureusement, ces valeurs variaient même lorsque qu'elles ne le devaient pas, nous nous sommes ensuite aperçus que le jumper de l'adaptateur n'était pas enclenché, ce qui provoquait un mauvais fonctionnement de la carte. Une fois le problème réglé, tout fonctionnait.

4.3) Test du solarimètre

Pour tester le solarimètre, nous avons utiliser un Superviseur ARIA. Le superviseur ARIA est un prédécesseur à notre projet. Il s'agissait d'un automate permettant de superviser des données, cependant, son logiciel n'était disponible que sur Windows XP/95/98/2000.

Voici un aperçu du logiciel :

Superviseur ARIA v05.03-7

Exploitation | Général arrosage | Climat | Arrosage | Bureau | EDF

Tableaux à Imprimer

Serre 1

T°C air mesurée en °C	T°C air désirée en °C	DF	Hygro mesurée en %	Déficit en g/kg	Assai- nisse- ment	T°C HT mesurée en °C	Position aération	Position abris en %	Ombre vent en %	Obscur thermiq. en %	Lampes
20,0	21,0	0	52	07,0	0	24,0	00	00	050	000	0

Chauffage Serre 1

T°C BT mesurée en °C	T°C BT calculée en °C	Pompe BT	T°C HT mesurée en °C	T°C HT calculée en °C	Pompe HT	Aerotherme	T°C VT mesurée en °C	T°C VT calculée en °C	Pompe VT
55	45	1	78	78	1	0	50	48	0

Serre 2

T°C air mesurée en °C	T°C air désirée en °C	Hygro mesurée en %	Déficit en g/kg	Assai- nisse- ment	Mode FOG	T°C HT mesurée en °C	Position aération	Position abris en %	T°C HT mesurée en °C	T°C HT calculée en °C	Pompe HT	Aerotherme
19,0	18,0	53	06,4	0	0	20,0	00	00	38	25	1	0

Serre 3

T°C air mesurée en °C	T°C air désirée en °C	Hygro mesurée en %	Hygro des en %	Niveau lumière en kLux	T°C eau mesurée en °C	T°C eau calculée en °C	Etat éclairage	Etat chauffage	Etat groupe	Etat dernier coupure	Heure du EDF
23,8	24,0	70	65	52,0	+16,7	+18,0	1111	0	0	0	06:00

On peut ainsi remarquer que les valeurs présente sont leurs valeurs reliées à l'automate, ici, il s'agit d'une image prise sur Internet, d'où le nombre de capteur présent.

Pour mettre en place ce superviseur, il faut brancher les capteurs sur l'automate, qui gère lui-même la boucle de courant 4-20 mA. Voici donc les branchements effectués pour la mise en place du superviseur. Et ensuite le relier à un ordinateur possédant le système d'exploitation XP.

Câblage du superviseur ARIA

Centrale pour
Automate M16



Automate M16

Capteurs

Nous avons ensuite testé de mettre le solarimètre dans la boucle de courant 4-20 mA, pour essayer de voir une variation au voltage de celui-ci, or, avec différent éclairage, les valeurs ne variaient point.

C'est pourquoi j'ai décidé de trouver d'où venait la panne dans le solarimètre.

4.4) Détection du problème

Pour détecter le problème, je suis tout d'abord parti du principe que l'automate fonctionnait, et que le capteur lui aussi, car il avait été testé l'année précédente.

J'ai donc utilisé un tableur pour réaliser une phase de test :

Procédure de test à 0 minutes			Procédure de test à 5 minutes		
Capteurs	T_Intérieure	T_Tuyaux	Capteurs	T_Intérieure	T_Eau
Valeurs	23,6	20	Valeurs	33,7	28
Calculée	27	50	Calculée	29	50
Puissance	4	6	Puissance	4	6

Solarimètre	
Sans lumière	0
Lumière ext	1
Flash 10 cm	4
Flash 0 cm	73

On constate ainsi que les deux sondes températures fonctionnent bien, cependant, les valeurs du solarimètre sont erronées.

Le capteur une fois brancher à l'automate, j'ai utilisé un Voltmètre pour voir si le capteur fonctionnait toujours. C'est à ce moment là que je me suis aperçu qu'il ne fonctionnait pas.

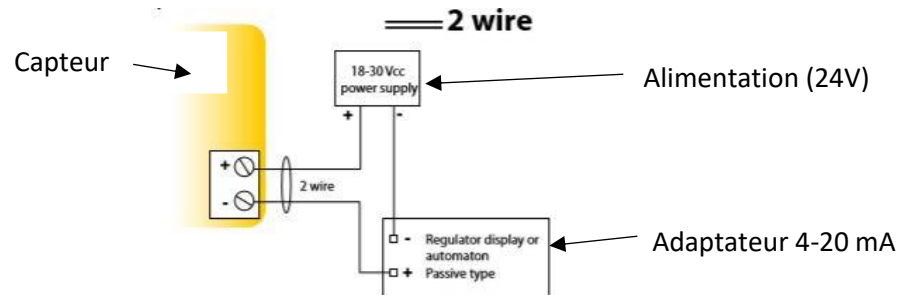
Pour aller plus loin, j'ai décidé de démonter le capteur pour voir le problème. Avec l'accord de mes professeurs référents, j'ai coupé les fils qui reliaient le capteur en lui-même à l'oscillateur.



Une fois les fils du capteur relié à la boucle de courant, je me suis aperçu que le courant variait sur le voltmètre. Le professeur de physique et moi-même avons donc conclu que l'élément qui ne fonctionnait plus était l'oscillateur.

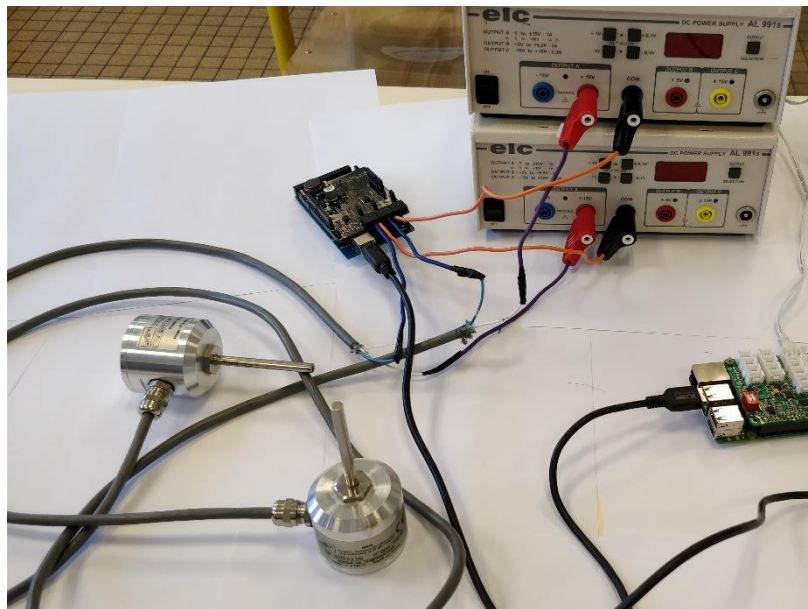
4.5) Récupérer les données d'un capteur sur la boucle 4-20 mA

Pour récupérer les données d'un capteur, il faut d'abord le brancher à l'alimentation externe et à l'adaptateur 4-20 mA. Le branchement change en fonction du nombre de câbles sur le capteur, ici, on a 2 câbles.



Ce qui nous donne, une fois monter :

Câblage de la boucle 4-20mA



Pour récupérer l'ensemble des données, nous avons utilisé la librairie fournie par le vendeur, et donc avons monté le programme en utilisant celle-ci.

Cette librairie nous permet de retourner 4 valeurs :

- `readChannel` (lecture d'un canal)
- `readVoltage` (lecture du voltage)
- `readCurrent` (lecture du voltage d'un canal)
- `isConnected` (savoir si un canal est connecté)

Ainsi, pour récupérer les valeurs du capteur de température de l'air, qui se trouve sur le canal 3, et qui fonctionne de 0°C à 45°C, on procède de cette façon :


```
if (sensorBoard.isConnected(CHANNEL3))
```

Cette ligne sert à savoir si un capteur est bien relié au canal 3

```
current = sensorBoard.readCurrent(CHANNEL3);
```

*Cette ligne sert à affecter la valeur du voltage au **float** current.*

```
valeur = ((current-4)*45)/16;
```

*Cette ligne sert à affecter la valeur de la température au **float** valeur.*

-4 correspond à la boucle 4-20mA, qui correspond à une boucle 0-16mA, qui facilite les calculs. Le 16 au maximum.

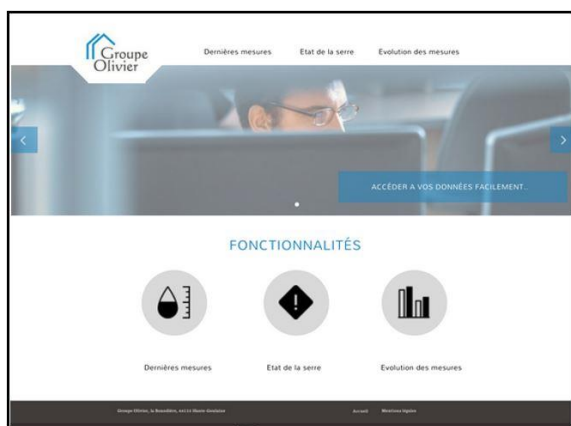
45 correspond à la valeur maximale que le capteur peut capter.

5. Mise en place de l'application Web

5.1) Conception de la charte graphique

Pour la conception de la charte graphique de l'application Web, nous nous sommes inspirés du site Web qu'ils utilisent actuellement. Ainsi, avec le site Canva, nous avons préparé les pages du site à l'avance pour savoir comment réaliser le CSS du site, ainsi que le HTML.

Canva est un site pour créer facilement des designs, ici, il nous sert à présenter nos pages Web, ainsi, nous avons réalisé :



Cette page est la page d'accueil du site, nous pouvons grâce à la barre de navigation naviguer sur les différentes pages.

Nous pouvons aussi y accéder en cliquant sur les icones que nous pouvons voir ici sous le titre « Fonctionnalités ».

Cette page est la page qui concerne la partie application Web de Samuel, l'étudiant 3. On y voit une serre en fond d'écran, où les dernières données des capteurs seront affichées dynamiquement.



Cette page est la page qui concerne ma partie de l'application Web. On pourra ainsi voir les données d'un capteur sur une période définie.

Ainsi on voit deux calendriers qui serviront à choisir la période, ainsi que la courbe.

Il a été décidé plus tard dans le projet de créer une page pour ajouter des capteurs directement via le site Internet, cependant, nous n'avons pas réalisé le design de cette page Web avec Canva.

Nous n'avons que l'IHM de la page en tant que telle, sans le pied de page, et la barre de navigation.

Ajouter un capteur

Nom

Type de relevé

1 : mm
2 : °C
3 : w/m²

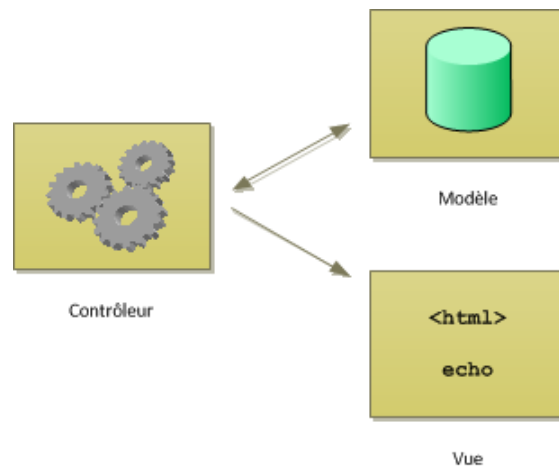
Autre

5.2) Architecture de l'application

Pour l'architecture de l'application, il a été demandé de suivre le modèle Modèle-Vue-Contrôleur. Ce pattern sert à bien organiser son code. Ainsi, il permet de diviser son code en trois grandes parties :

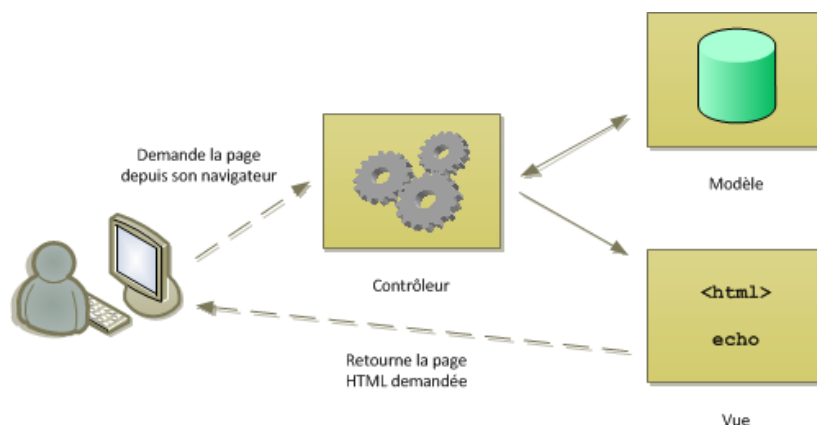
- **Modèle** : Cette partie gère les données du site. C'est entre autres elle qui va chercher les informations brutes dans la base de données, donc ce sont des requêtes SQL.
- **Vue** : Cette partie se concentre sur l'affichage. Elle se contente de récupérer les variables, et comporte essentiellement du code PHP, et HTML. Dans notre projet, il s'agira d'un script.
- **Contrôleur** : Cette partie gère la logique du code qui prend des décisions. C'est lui qui fait le lien entre le modèle et la vue. Donc il va demander les données au modèles les données, les analyser et ensuite envoyer le texte à afficher à la vue.

Echange d'informations entre les éléments



Concrètement, le visiteur demandera la page au contrôleur et c'est la vue qui lui sera retournée.

La requête du client au contrôleur qui affiche la vue



Dans le projet, nous aurons ainsi trois pages dans ma partie :

- [Evolution_mesure.php](#) (le contrôleur)
- [Graphique_mesure.js](#) (la vue)
- [Recuperer_donnees.php](#) (le modèle)

5.3) Connexion à la base de données

La connexion à la base de données s'effectue dans une page située dans un dossier inc. Il est nommé connect.php. Notre base de données étant sur un serveur distant, il doit être renseigné l'adresse de celui-ci, et beaucoup d'autres données. Ce qui nous donne :

```
$db = null;
$db_engine = 'mysql';
$host = '92.222.92.147';
$charset = 'utf8';

$db_user = 'projetbts';
$db_password = 'Nantes44';
$db_base = 'supervision_serre';
```

On indique le moteur de la base de données

Adresse de la base de données

Information concernant la table, le mot de passe et la base de données

Après avoir renseigné tous ce qu'il nous faut comme informations pour la connexion, on essaye de se connecter à la base de données :

```
try{

    $dsn = "mysql:host=$host;dbname=$db_base;charset=$charset";
    $options = [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_OBJ,
        PDO::ATTR_EMULATE_PREPARES => true,
    ];
    $bdd = new PDO($dsn, $db_user, $db_password, $options);
}
```

\$dsn prend en paramètre les informations

PDO = PHP Data Objects

On crée la connexion avec les paramètres

Ensuite, si la connexion échoue, l'utilisateur doit être averti par un message d'erreur :

```
catch (PDOException $e){
    print(json_encode(array('outcome' => false, 'message' => 'Impossible de se connecter !')));
}
```

Ici, si la connexion à la base de données n'est pas possible, il y aura un message d'erreur qui s'affichera sur les pages possédant une connexion à la base de données.

5.4) Gestion de la période

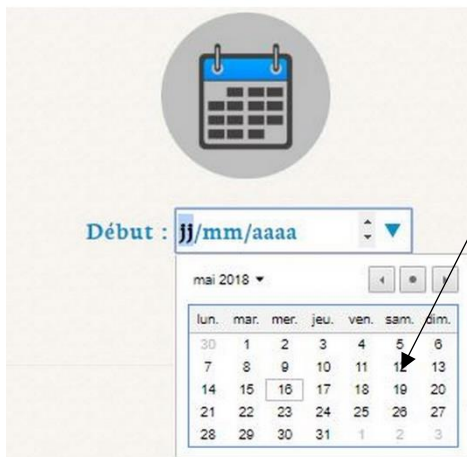
Pour afficher le graphique, dans un premier temps, l'utilisateur doit choisir une période, ainsi dans la page `evolution_mesure.php`, deux `input` de type `'date'` sont présent sur la page.



```
<input type="date" name="date_1" style="text-align: center">
```

L'élément `<input>` de type `date` permet de créer un champ pour la saisie d'une date. Elle est composée d'une année, d'un mois et d'un jour.

Ainsi, après avoir effectué un clic sur l'input, on aura un affichage beaucoup plus aisé pour choisir les dates :



En cliquant sur une date, la date s'auto inscrit dans l'input.

Le `name='date_1'` sera ensuite utiliser pour le code PHP, c'est ce qui va pouvoir modifier les deux valeurs rentrées dans les input.

Pour ensuite envoyer les informations, on utilisera un bouton de validation, et un formulaire.

```
<form method="get">
<button type="submit" value="submit" class="btn btn-primary">Valider</button>
```


Ici le formulaire a pour méthode get, c'est-à-dire que les valeurs des deux dates seront envoyées directement dans l'URL. Nous aurons ainsi :

```
evolution_mesure.php?date_1=2018-05-03&date_2=2018-05-18
```

Bien évidemment les dates auparavant choisies doivent être le 3 Mai 2018 et le 18 Mai 2018.

Il faut ensuite pouvoir gérer les différentes erreurs, comme le fait de choisir une date de fin plus récente que la date de début. Pour montrer cette erreur au superviseur, j'ai choisi d'afficher un message d'erreur directement sur la page.

```
$date_debut = $_GET['date_1'];
$date_fin = $_GET['date_2'];
if ($date_fin < $date_debut) {
    echo '<h1 style="color: red;">Les dates sont erronées.</h1>';
}
```

Le script ici récupère les valeurs dans l'URL, et si la date de fin est plus récente que la date du début, le superviseur sera ainsi informé par un message que les dates qu'il a sélectionnées sont erronées. Le graphique ne sera pas affiché sur le message d'erreur apparaît.

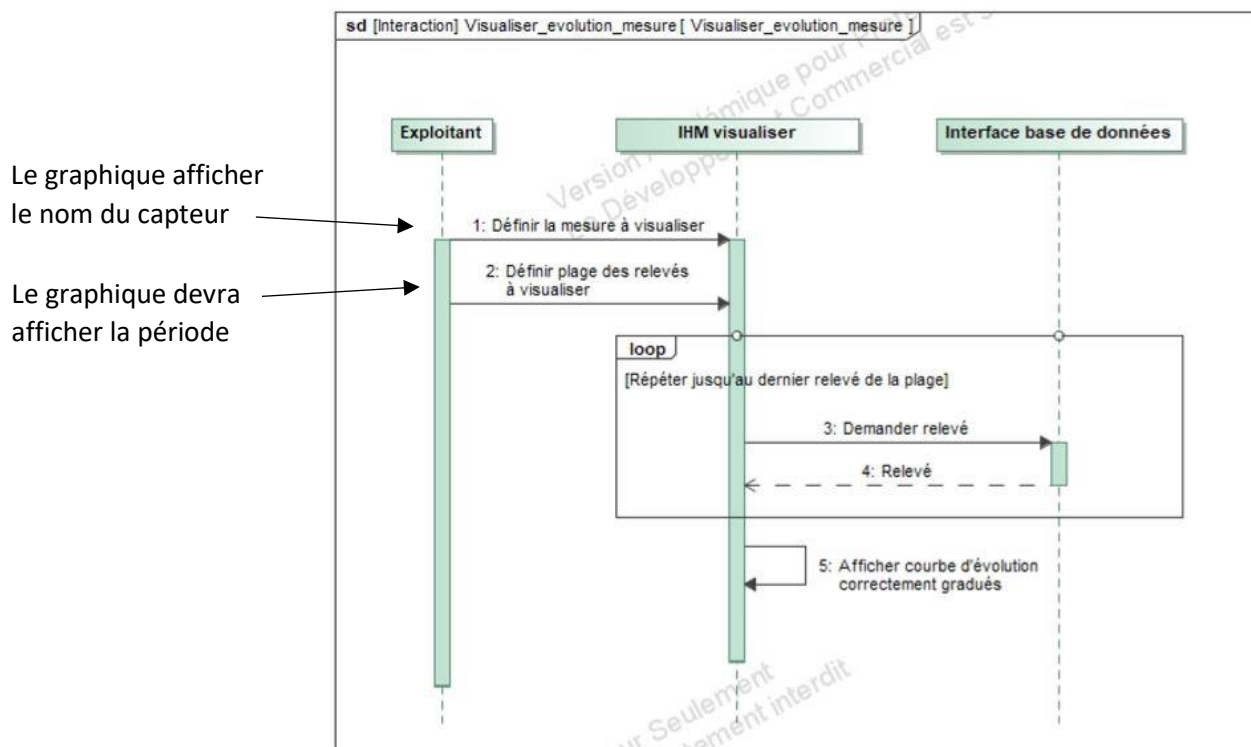
5.5) Affichage d'un graphique

5.5.1) Requêtes pour récupérer les données du graphique

La partie affichage d'un graphique correspond à la Vue du modèle MVC. Quant au PHP pour récupérer les données, il s'agit du modèle.

Pour afficher le graphique, il faut pouvoir récupérer 3 valeurs distinctes. Le nom du capteur, la période choisie par le superviseur ainsi que les valeurs du capteur capturés durant la période.

On peut ainsi repérer les requêtes à faire une à une sur le diagramme de séquence du système.



Pour récupérer le nom du capteur, nous utiliserons une variable :

- `$nom_capteur = $_GET['choix_capteur'];`

La valeur du `$nom_capteur` prendra en paramètre le choix d'un select qui comportera tous les noms des capteurs.

Pour récupérer les dates, nous utiliserons simplement les variables :

- `$date_debut = $_GET['date_1'];`
- `$date_fin = $_GET['date_2'];`

La méthode GET est utilisée ici car les valeurs des dates sont placées dans l'URL de la page Internet.

Pour récupérer les valeurs de chaque capteur dans l'intervalle donné, il faudra utiliser les deux paramètres ci-dessus :

- ```
function getIDofSensor($dbh)
{
 $request = $dbh->prepare('SELECT id FROM materiel WHERE nom = $nom_capteur');
 return $request->execute() ? $request->fetchAll() : null;
}
$sensor = getIDofSensor($bdd);
```

Cette fonction permet de récupérer l'ID grâce au nom du capteur. Et on affecte cet ID à la variable `$sensor`.

- ```
function getFullOfValues($dba)
{
    $request = $dba->prepare('SELECT valeur FROM releve WHERE id_materiel = $sensor'
    AND WHERE valeur BETWEEN $date_debut AND $date_fin);
    return $request->execute() ? $request->fetchAll() : null;
```

Cette fonction quant à elle permet de récupérer toutes les valeurs correspondantes à l'ID du capteur choisi, se situant entre la période choisie.

5.5.2) Insérer des données au graphique

Pour pouvoir insérer les valeurs PHP dans le tableau, il faut encoder les tableaux du script en JSON pour que Javascript puisse les lire. Ainsi, on utilise la fonction `json_encode`.

Ainsi, pour reprendre l'exemple du tableau de valeur ci-dessus, on écrit :

```
$series = [  
  'name' => [$sensors],  
  'type' => ['line'],  
  'data' => [$releves]  
];
```

Name, type et data sont les paramètres pour définir le graphique en javascript. On leur injecte ainsi les valeurs reçues par le code PHP. C'est alors ici que la fonction va servir, car le Javascript ici ne peut lire que des données extraites au format JSON.

```
json_encode($series);
```

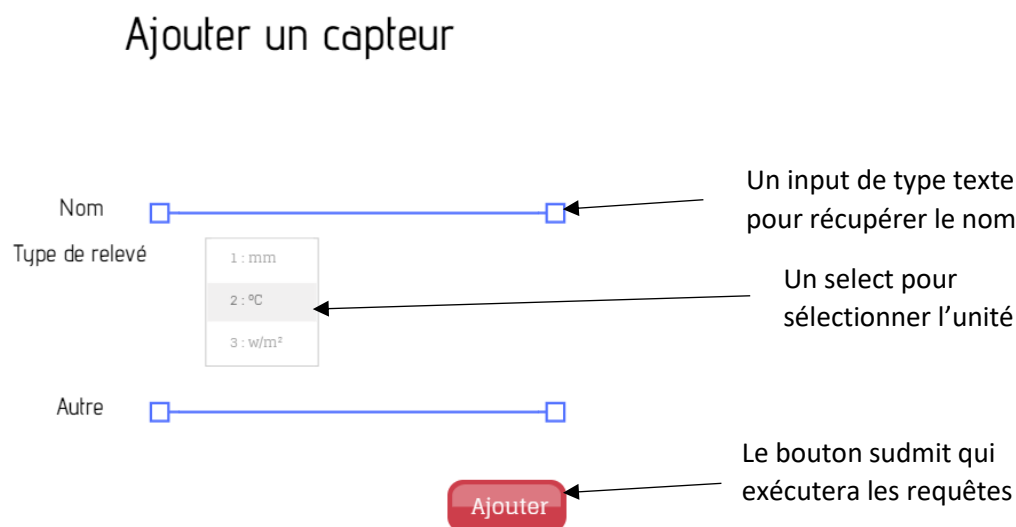
5.6) Création d'une page pour ajouter un capteur

Pour pallier la contrainte d'évolutivité, il a été décidé avec Samuel d'ajouter un page pour créer un capteur, et l'ajouter directement à la base de données. Ainsi, grâce au site Web, le superviseur pourra ajouter le capteur avec les données qu'il souhaite, ainsi, via cette page, il pourra ainsi ajouter :

- Le nom du capteur
- Son abréviation
- L'unité du matériel
- Le type du matériel

Ainsi, la page ressemblera à :

Ajouter un capteur



The form consists of the following elements:

- Nom**: A text input field for the sensor name.
- Type de relevé**: A select menu with three options: "1 : mm", "2 : °C", and "3 : w/m²".
- Autre**: Another text input field.
- Ajouter**: A red submit button.

Annotations with arrows:

- Un input de type texte pour récupérer le nom (points to the 'Nom' input field).
- Un select pour sélectionner l'unité (points to the 'Type de relevé' select menu).
- Le bouton submit qui exécutera les requêtes (points to the 'Ajouter' button).

Lors du clic sur le bouton Ajouter, de nombreux INSERT se feront dans la base de données. Par exemple, pour ajouter le nom du capteur, on aura :

```
$nom_capteur = $_POST['nom_capteur'];
INSERT INTO materiel (nom) VALUES ($nom_capteur)
```

Je n'ai pas fait la requête préparée encore, car ce n'est qu'une ébauche de travail que nous allons réaliser.

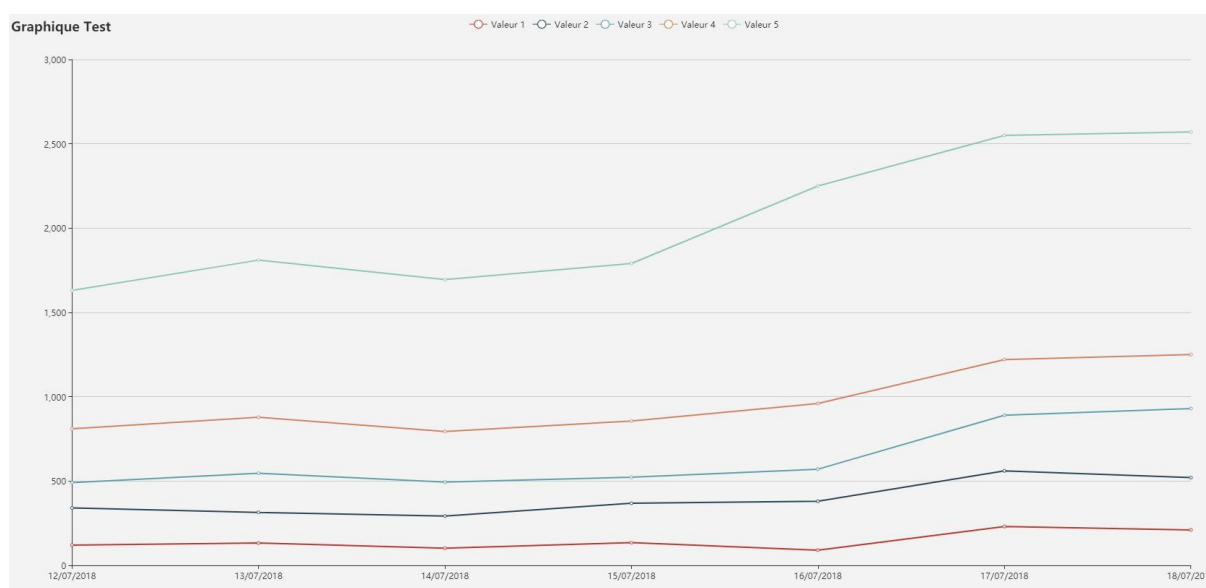
5.7) Modification que j'aimerais apporter

Pour avoir une visualisation plus élaborée dans le site, j'aimerais supprimer le select qui permet de choisir un capteur.

Ainsi, j'afficherai le graphique avec tous les capteurs dans la période adéquat.

Le script qui me permet d'afficher le graphique peut masquer des données lors d'un clic dessus.

Par exemple, si l'utilisateur choisi une période du 12 Juillet 2018 au 18 Juillet 2018, il aura :



Maintenant, si l'on appuie sur la valeur 1 et la valeur 2, ces valeurs seront désactivées, comme-ci présent :



Ainsi, le graphique nous affichera que les valeurs des capteurs 3,4 et 5.

6. Tests unitaires

6.1) Test unitaire de la méthode `get_temperature_air()`

La méthode `get_temperature_air()` est présente dans le programme Arduino, cette méthode sert à convertir le voltage en température. Elle comprend ainsi la valeur du voltage au canal où se trouve le capteur de température, et un produit en croix.

Le voltage récupéré par l'adaptateur 4-20mA peut varier de 4mA à 20mA. Ainsi, le produit en croix permet de réaliser l'échantillonnage des valeurs. 4mA correspondra ainsi à la valeur minimale du capteur de température de l'air, soit 0°C, et 20mA correspondra à la valeur maximale, soit 45°C.

Le test unitaire servira à vérifier que les valeurs se situent bien entre 0 et 45°C.

Pour faire ce test unitaire, j'ai utilisé une librairie postée sur Github pour les tests unitaires sur Arduino. La librairie se nomme `ArduinoUnit` et a été créée par Matthew Murdoch.

Pour inclure la librairie au programme, je l'inclus au code :

```
#line 2 "sketch.ino"
#include <ArduinoUnit.h>
#include <currentLoop.h>
```

Je code ensuite mon test unitaire :

```
test(temperature_air_max)
{
    float valeur=get_temperature_air;
    int max_degre=45;
    assertLessOrEqual(valeur,max_degre);
}
```

Valeur prend la valeur du capteur

On initialise la température max à 45°C

On regarde sur la valeur est inférieure à 45°C

On effectue le même test pour la température minimale.

```
test(temperature_air_min)
{
    float valeur=get_temperature_air;
    int min_degre=0;
    assertMoreOrEqual(valeur,min_degre);
}
```

Il faut ensuite définir le débit de la communication série à 9600 Bauds :

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

Ensuite, il faut juste lancer en boucle le test unitaire.

```
void loop()
{
    Test::run();
}
```

Elément testé :	Méthode get_temperature_air()		
Objectif du test :	Vérifier que les données retournées par get_temperature_air() sont comprises dans l'intervalle de données du capteur.		
Nom du testeur :	Willy RINEAU	Date :	Lundi 05 Mai 2018
Procédure de test			
Description du vecteur de test :	Résultat attendu :	Résultat obtenu :	Validation (O/N)
Tester la valeur maximale que le capteur puisse acquérir.	Avoir une température inférieure à 45°C, qui est le maximum que peut collecter le capteur de température.	Test temperature_air_max passed.	O
Procédure de test			
Description du vecteur de test :	Résultat attendu :	Résultat obtenu :	Validation (O/N)
Tester la valeur minimale que le capteur puisse acquérir.	Avoir une température supérieure à 0°C, qui est le minimum que peut collecter le capteur de température.	Test temperature_air_min passed.	O
La procédure de test s'effectue jusqu'à la fin.	La procédure termine son exécution sans rencontrer d'erreurs.	Test summary : 2 passed, 0 failed, and 0 skipped , out of 2 test(s).	O
Conclusion du test :	Les résultats attendus sont validés. Les données reçues sont bien comprises entre 0 et 45°C.		

7. Conclusion

Pour apporter une conclusion sur les tâches que j'ai eu à réaliser au sein de ce projet, je dirais que ce fut une expérience et un projet très enrichissant autant sur le point de vue technique que personnel. En effet, j'avais déjà des connaissances dans la création de site Web, avec le langage PHP, SQL et les langages de balises comme HTML et CSS, mais ce projet a pu renforcer mes compétences dans ce domaine, et dans un autre domaine qui est l'embarqué avec notamment la carte Raspberry et la carte Arduino. Les outils logiciels et matériels que j'ai eu à ma disposition m'étaient familiers et ont contribué au bon déroulement de mes tâches.

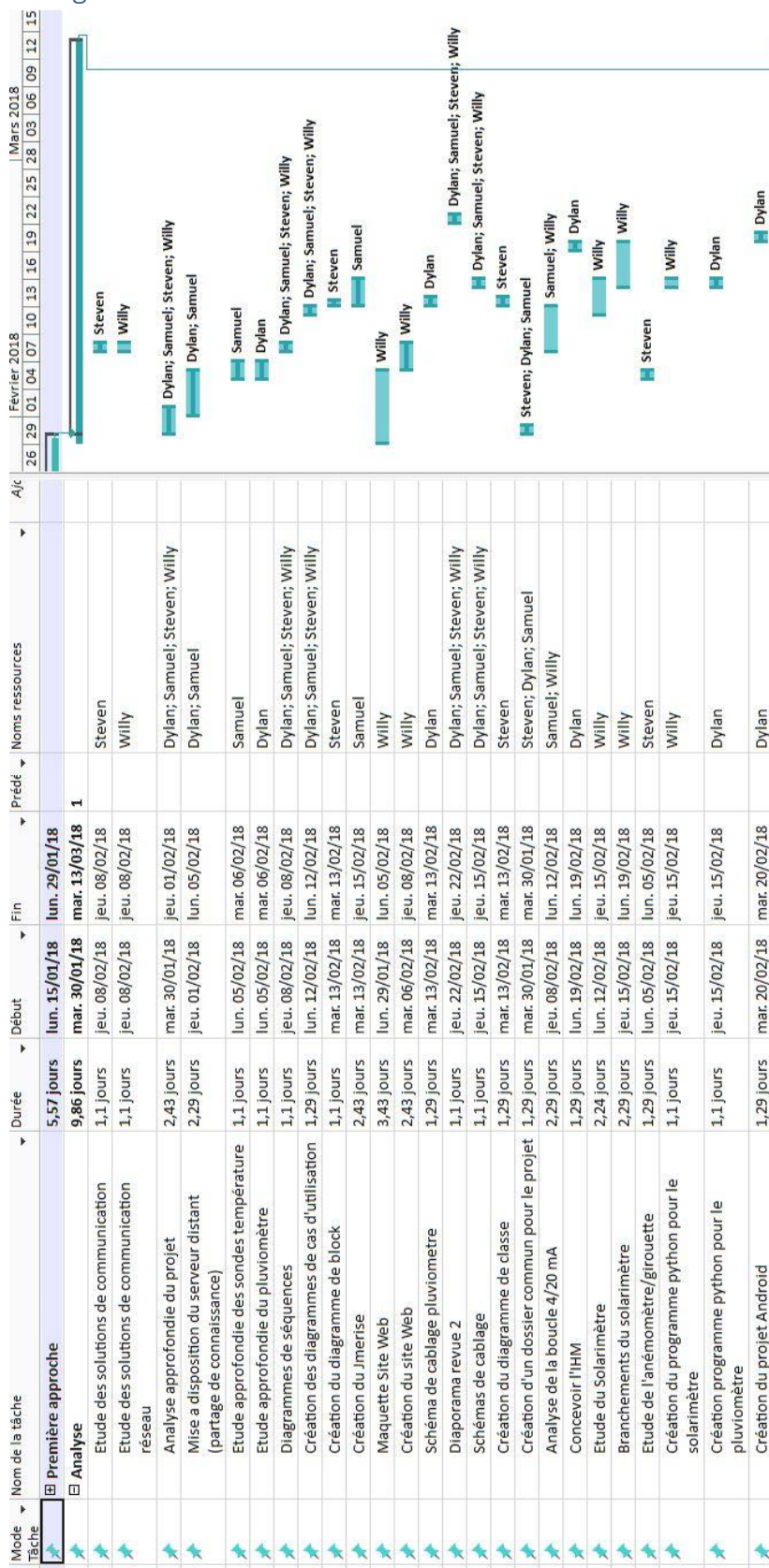
Durant ce projet, j'ai découvert le travail en équipe, et donc former des liens avec l'ensemble du groupe, plus particulièrement avec Samuel, qui travaillait avec moi pour la mise en place de la boucle 4-20mA, et pour le site Web. Nous nous sommes ainsi partagé les tâches, et donc nous avons dû faire preuve de rigueur quant à la disposition des fichiers. Ce travail en équipe fut une expérience enrichissante, au point de vue personnel, social et professionnel.

Du point de vue professionnel, la mise en place de la boucle 4-20mA est un plus, car il s'agit d'un standard de l'industrie, et la création de site Web est toujours essentiel aux entreprises.

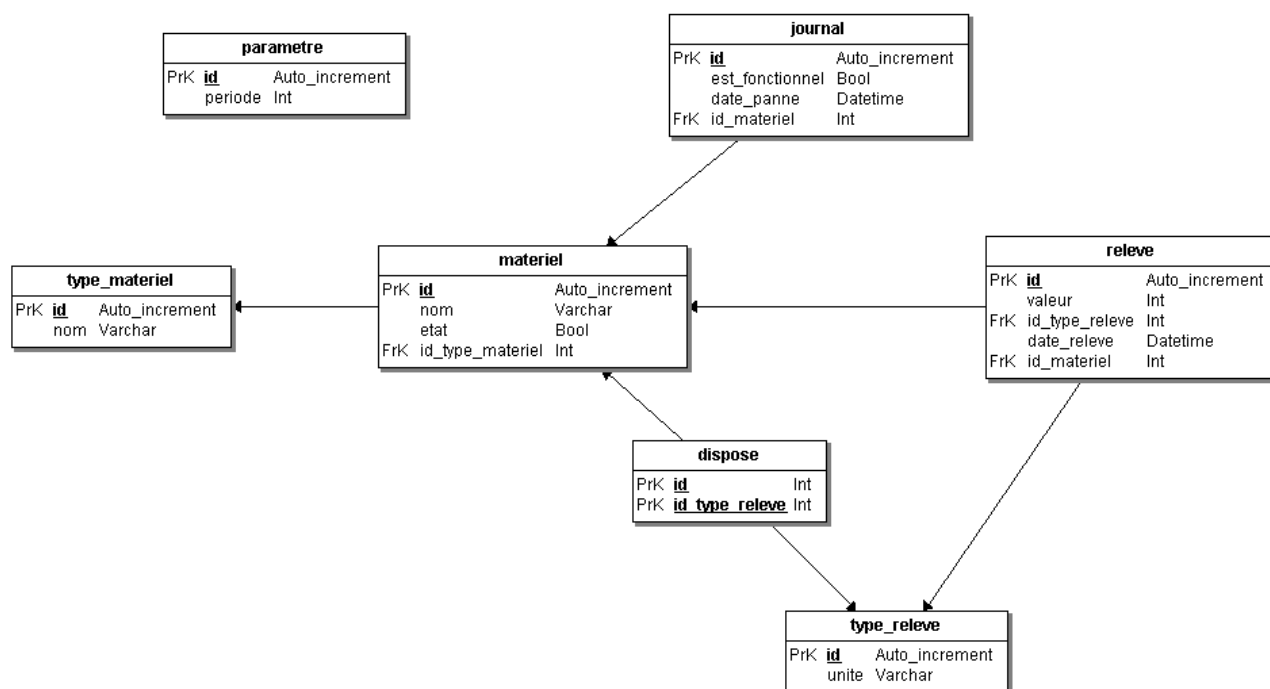
Sur le plan personnel, je pense que toutes ces connaissances acquises constitueront un bagage solide dans le cadre de la poursuite d'études que je souhaite entreprendre. En effet, suite à mon admission à l'ENI en tant que Concepteur Développeur Informatique pour une formation en alternance, beaucoup d'entreprises que j'ai rencontré cherchaient des profils répondant à ce critère.

8. Annexes

Annexe 1 – Diagramme de Gantt



Annexe 2 – Conception de la base de données

**Table journal :**

Cette table contient le journal de chaque matériel, ainsi, nous pouvons savoir si un matériel est fonctionnel, et la date de la panne, s'il y en a une. Cette table sert essentiellement à l'application mobile.

Table materiel :

Cette table contient chaque matériel et les informations permettant de les distinguer au sein du projet pour l'acquisition des mesures.

Table type_materiel :

Cette table contient le type des matériels, car deux matériels peuvent avoir le même type.

Table parametre :

Cette table contient le paramètre pouvant modifier la période.

Table materiel_unite :

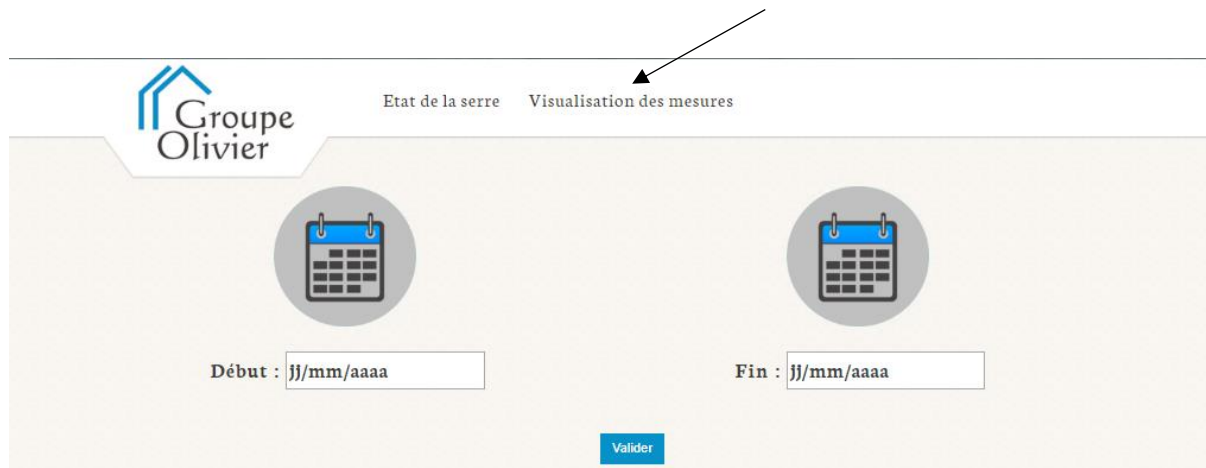
Cette table contient les unités des capteurs. Elle fait le lien entre les relevés et le matériel.

Table releve :


Cette table contient les différents relevés effectués par les capteurs, ils sont datés et ont un ID personnel.

Annexe 3 – Manuel utilisateur de la page evolution_mesure.php

Pour vous rendre sur la page de Visualisation des mesures, il est nécessaire de cliquer sur l'onglet correspondant dans la barre de navigation.




Une fois après avoir cliquer sur l'onglet, vous vous trouverez face à cette page, où deux champs de texte sont présents, avec un bouton validé. Une fois après avoir cliqué dans le premier champs nommé « Début », un calendrier apparaîtra.



Un simple clic sur la date voulu et la date en rentrée dans le champ de texte

Voici le résultat lorsque l'on clique sur la date.

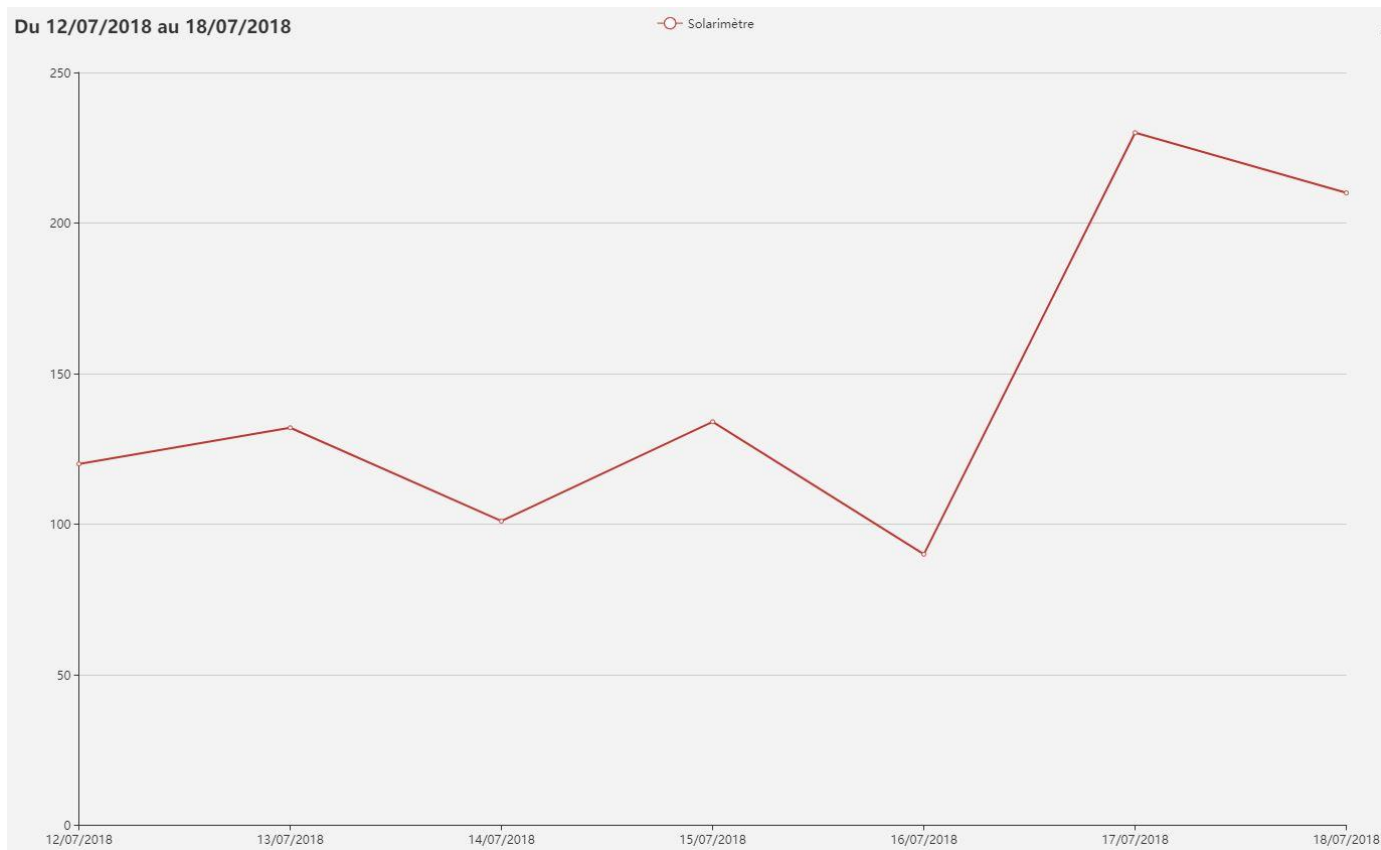
Le calendrier disparaît et auto complète le champ.



Une fois les deux dates choisies, il faudra appuyer sur le bouton

Valider

Si la date de début est inférieure à la date de fin, le graphique avec les données s'affichera de cette façon :



Sinon un message d'erreur apparaîtra. Et vous indiquera que les valeurs sont erronées. Ainsi, il vous suffira de modifier les valeurs dans le formulaire de la période et de réappuyer sur valider.