Dossier technique du projet – Partie personnelle

Etudiant 3 : *Création base de données*
Mise en place sondes température
Mise en place page web

Sommaire

I. Situation dans le projet

1.1) Synoptique de la réalisation

Durant ce projet, mes tâches étaient de mettre en place les sondes de température pour la température sous serre et la température de l'eau des tuyaux de chauffage, de créer la base de données et également de mettre en place le site web, notamment la page web affichant l'état de la serre en temps réel.

1.2) Rappel des tâches de l'étudiant

Comme je le dis précédemment, dans ce projet de supervision d'une serre, j'avais trois tâches.

Le premier point était de mettre en place une base de données répondant aux attentes du site web et de l'application Android. Je me suis donc concerté avec mes collègues de projet afin de savoir quelles tables mettre en place etc… La base de données est hébergée sur un serveur distant loué par Dylan (l'étudiant 4).

Deuxièmement, je devais mettre en place les sondes de températures pour l'eau de tuyaux de chauffages de la serre et pour la serre. Pour se faire, je devais, en commun avec Willy (l'étudiant 2), mettre en place une boucle 4-20 mA (boucle qui permet de transmettre un signal analogique sur une grande distance sans modifier ou perdre ce signal).

Enfin ma troisième tâche était de mettre en place le site web (également avec Willy), plus précisément une page web affichant en temps réel l'état de la serre, c'est-à-dire d'afficher les différentes acquisitions effectuées par tous les capteurs. J'étais tout d'abord partis sur une page avec une serre fictive en fond sur laquelle se trouvait en icônes chaque capteur, et sur lesquels il suffisait de cliquer pour afficher les différentes mesures (celles-ci apparaissant sous forme de popover).

Après réflexion, je suis parti sur une autre idée pour deux raisons :

- Cliquer sur chaque capteur pour voir chaque mesure n'étais pas optimal pour l'exploitant
- La contrainte de pouvoir facilement ajouter un capteur était trop difficile à mettre en place avec les popovers.

1.3) Contraintes liées au développement

Durant le projet, nous avions des contraintes à respecter.

La première contrainte était une **contrainte financière**. Nous avions un budget de 100 euros prévu pour l'achat d'une carte adaptateur 4-20 mA (Shield pour Arduino). Nous avons eu à choisir avec l'étudiant 2, un adaptateur permettant de lire plusieurs canaux (car plusieurs capteurs) afin de mettre en place la boucle 4-20 mA.

La deuxième, la **contrainte de développement**, spécifiant que l'on doit réaliser le site Web sous le patron Modèle-Vue-Contrôleur. Le framework Symfony aurait pu être utilisé, nous n'avons cependant pas choisit de l'utiliser car les requêtes que nous utilisons restent basiques.

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 sera minime, voir automatique. De plus une documentation complète sur le système doit être fournie au client, pour qu'une fois le projet terminé, une autre équipe puisse donner suite à ce projet.

La globalité de ce projet aura pour objet la gestion automatique d'une serre maraîchère et se décomposera en deux parties :

- La supervision de l'état de la serre avec récupération et stockage de l'ensemble des données nécessaires ;
- L'automatisation de la régulation de la température, l'hydrométrie et de l'intensité lumineuse de la serre.

Ce projet sera porté sur deux années. Le projet décrit par la suite se limite donc à la première partie : La supervision de l'état de la serre.

II. Conception et mise en œuvre

2.1) Fonctionnement des sondes température

Durant le projet, j'ai eu à mettre en place deux sondes température :

- Une sonde pour la température de l'eau des tuyaux de chauffage
- Une sonde pour la température sous serre

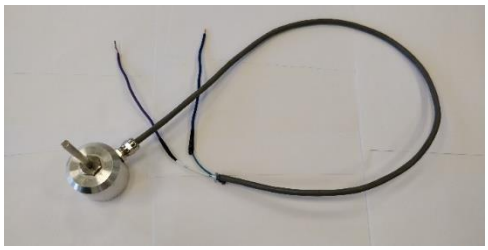
J'avais à ma disposition deux sondes Pt100 classe A de la marque Aria qui sont des sondes industrielles. Du point de vue visuel elles sont identiques, le seul point qui change est la plage de mesure.

Sonde Pt100 classe A pour l'eau des tuyaux de chauffage :

Capteur : Pt100 classe A
Plage : 0-100°C
Précision : +/- 0,3°C
Convertisseur : intégré 4-20mA 2 fils
Alimentation : 20 à 30 volts régulés
Charge typique 500 Ohm
Connexion par bornes à visser

Sonde température pt100 pour l'air dans la serre :

Capteur : Pt100 classe A
Plage : 0-45°C
Précision : +/- 0,3°C
Convertisseur : intégré 4-20mA 2 fils
Alimentation : 20 à 30 volts régulés
Charge typique 500 Ohm
Connexion par bornes à visser



Afin de fonctionner, ces capteurs ont besoin d'être alimentés. Ces deux sondes sont des sondes à résistances dont le principe de mesure est basé sur la variation de la résistance laquelle est directement liées à la variation de la température. Les câbles blanc et bleu, correspondent respectivement au plus et au moins.

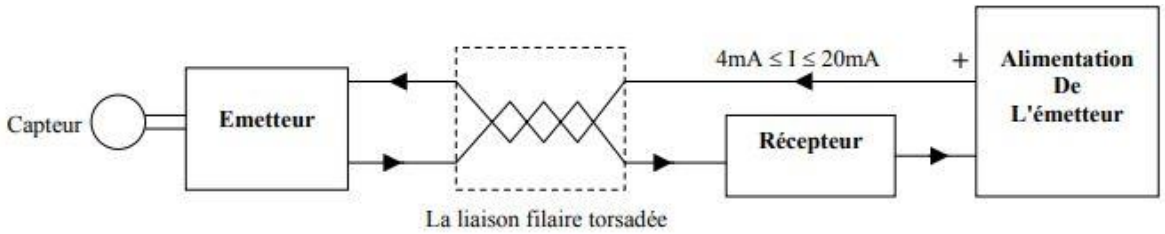
2.2) Fonctionnement de la boucle 4-20 mA

La boucle de courant est une méthode utilisée en contrôle industriel pour communiquer avec des capteurs ou des actionneurs, consistant à faire circuler dans une paire de conducteurs électriques un courant dont l'intensité est l'image du signal à transmettre.

La boucle de courant la plus utilisée dans l'industrie est le 4-20 mA, où 4 mA représente le minimum d'échelle, et 20 mA représente le maximum d'échelle, avec une relation linéaire entre le signal à transmettre et l'intensité du courant (c'est finalement un simple produit en croix).

Son principal avantage est que la précision du signal transmis n'est pas affectée par les pertes en ligne, car le courant circulant dans la boucle est fourni par une source de courant dont l'intensité est régulée pour correspondre au signal à transmettre, quelle que soit la résistance de la ligne.

Le schéma fonctionnel suivant peut représenter le principe de la boucle :



Pour mettre en place la boucle 4-20 mA on a acheté, avec le budget alloué, un adaptateur 4-20 mA. Je parlerais dans la partie 4 de ce choix plus en détail.

2.3) Réalisation du diagramme de classe

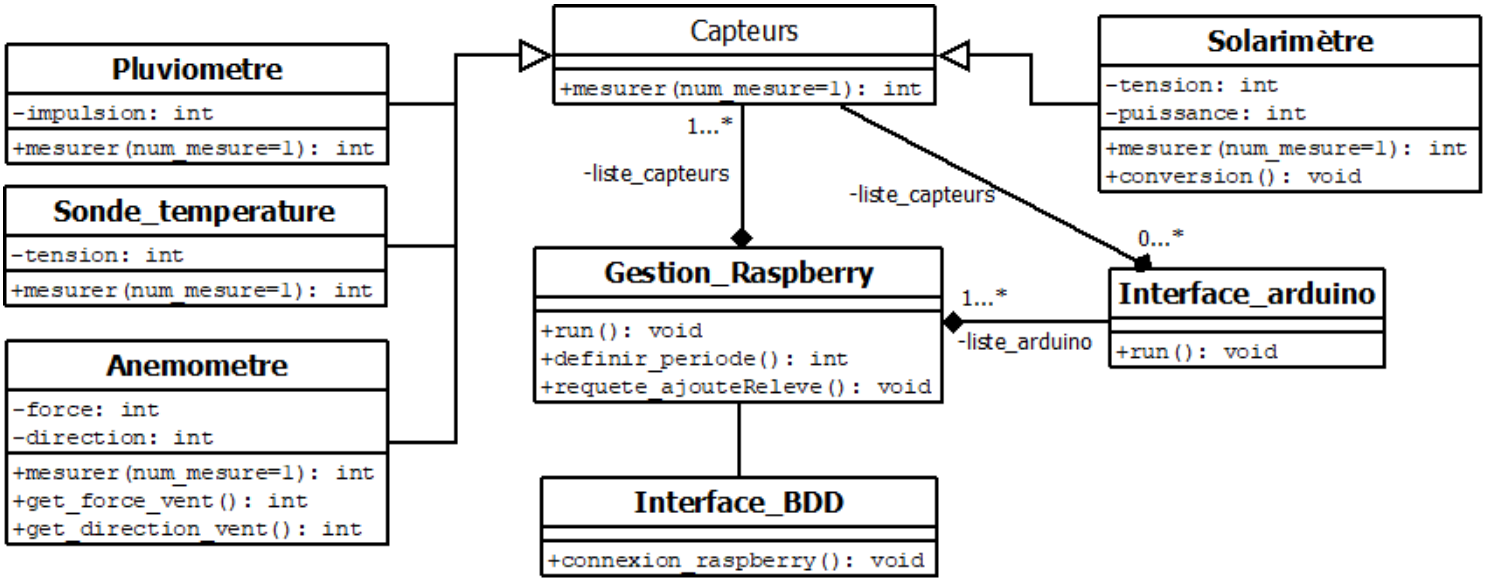


Diagramme de classe

Ce diagramme de classe a été fait en groupe. En fonction de commun nous allions faire le projet nous avons produit ce diagramme de classe qui a connu quelques changements ensuite. Le diagramme de classe présent ci-dessus est le diagramme final. Les classes présentes dans le diagramme sont les classes qui composeront le programme python que fera Steven (étudiant 1).

Toutes les classes ne me concernant pas directement, je parlerais par la suite que de celles auxquelles je suis lié.

La classe Capteurs :

C'est la classe mère de la classe **Sonde_temperature** qui est la classe a laquelle je suis le plus lié. 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 Sonde_temperature :

C'est la classe fille de la classe **Capteurs** car elle hérite de celle-ci. Ainsi cette classe contient la méthode *mesurer()* que je devais coder pour que les mesures effectuées par les sondes température puissent se stockées dans la base de donnée par la suite. Cette classe est donc la classe à laquelle je suis le plus lié.

La classe interface_BDD :

Cette classe contient la méthode connexion_raspberry() qui contient le code effectuant la connexion entre le programme python et la base de données. Cette classe ne me concerne pas vraiment étant pour le programme python que Steven (étudiant 1) code.

III. Mise en place de la base de données

3.1) Modèles entité association base de données

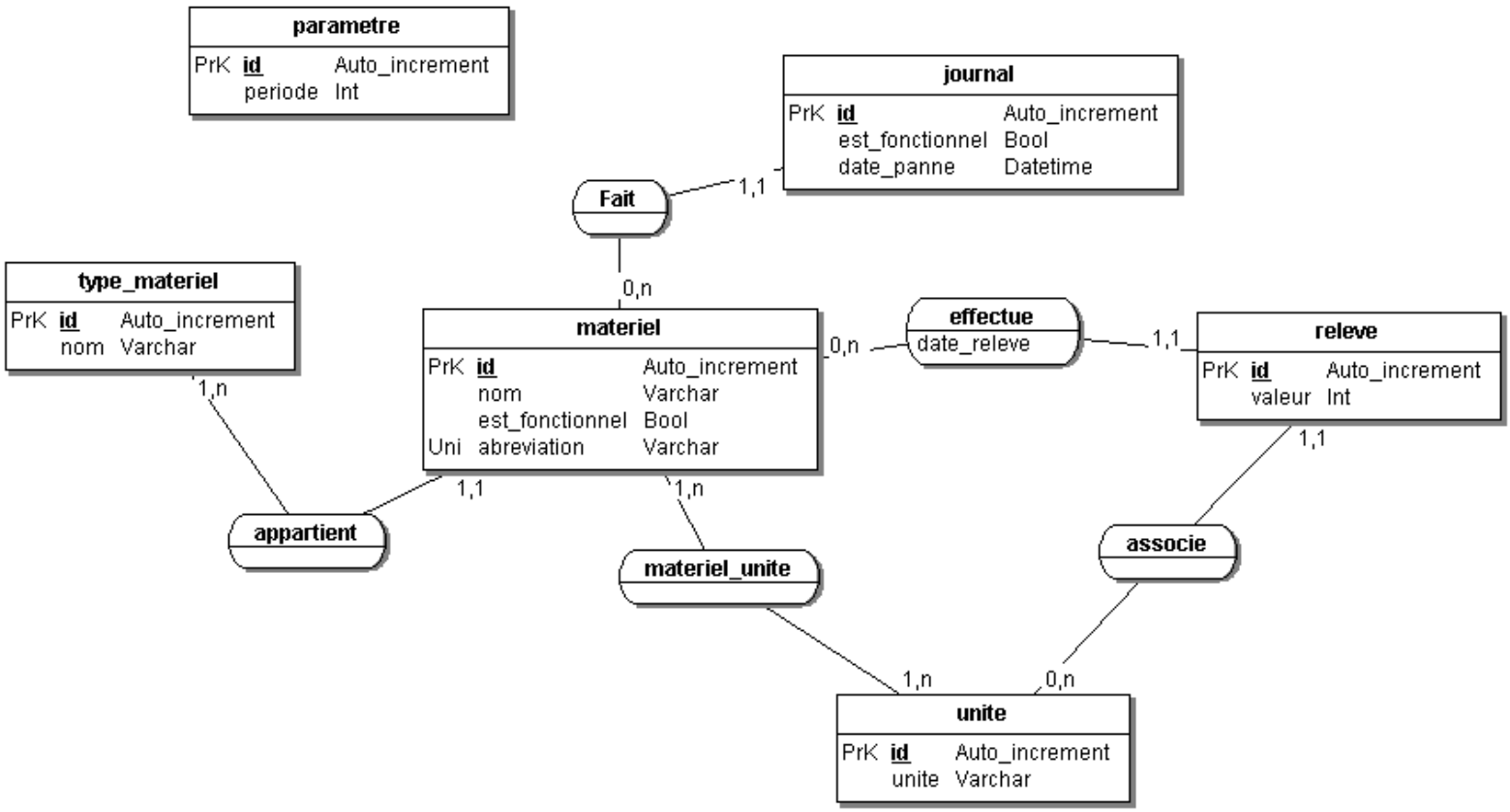
Comme je le disais plus tôt, afin de mener à bien ma tâche qui était de créer la base de données, je me suis concerté avec les autres membres du groupe.

J'organisais en quelques sortes des petites réunions de 5 minutes afin de savoir les éléments dont ils avaient besoin. Etant donné que durant le projet les idées évoluent, la base de données connue plusieurs versions.

Entité association

Le modèle entité association permet de représenter par un schéma standardisé les différents éléments constitutifs du système d'information appelés attributs et les relations qui les unissent appelées associations.
J'ai utilisé le logiciel Jmerise pour pouvoir créer mon modèle entité-association.

Schéma entité association actuel de la base de données



3.2) Description des différentes tables

Table « journal »

Table contenant le journal de chaque matériel, c'est-à-dire s'il est fonctionnel ou pas et la date (nommée date_panne). Cette table est faite pour l'application Android de Dylan (étudiant 4). Son application doit montrer l'état de fonctionnement des différents matériels utilisés pour l'acquisition des mesures.

Nom	Type	Description
id	INT	Clé primaire
est_fonctionnel	INT	Champ définissant si le matériel est fonctionnel : • 0 non fonctionnel • 1 fonctionnel
date_panne	DATETIME	Champ contenant la date et l'heure correspondant à l'état du matériel, ainsi si un matériel vient à être non fonctionnel, on sait la date de la panne.
Id_materiel	INT	Clé étrangère permettant d'obtenir le matériel.

Table « matériel »

Table contenant chaque matériel présent dans le projet pour l'acquisition des mesures.

Nom	Type	Description
id	INT	Clé primaire
nom	VARCHAR(255)	Champ définissant le nom de chaque matériel ajouté à la base de données.
abrevation	VARCHAR(250)	Champ définissant le nom abrégé de chaque matériel car Steven (étudiant 1) en a besoin.
id_type_materiel	INT	Clé étrangère permettant d'obtenir le type de matériel.

Table « type_materiel »

Table contenant le type des matériels, car deux matériels peuvent avoir le même type.

Nom	Type	Description
id	INT	Clé primaire
nom	VARCHAR(255)	Champ contenant le nom du type de matériel

Table « parametre »

Table contenant les paramètres de périodisation.

Nom	Type	Description
id	INT	Clé primaire
periode	INT	Champ contenant la période à laquelle les données s'enregistre dans la base de données.

Table « materiel_unite »

Table faisant le lien entre la table « unite » et la table « materiel ».

Table « unite »

Table contenant les unités correspondantes à chaque matériel de type capteur.

Nom	Type	Description
id	INT	Clé primaire
unite	VARCHAR(255)	Champ contenant l'unité propre à chaque capteur.

Table « releve »

Cette table contient les différents relevés effectués par les capteurs.

Nom	Type	Description
id	INT	Clé primaire
valeur	INT	Champ contenant la valeur des différents relevés effectués
id_unite	INT	Clé étrangère permettant d'obtenir l'unité de chaque relevés
date_releve	DATETIME	Champ contenant la date de chaque relevé effectué.
id_materiel	INT	Clé étrangère permettant de savoir par quel matériel est effectué le relevé.

PhpMyAdmin

PhpMyAdmin est un des outils les plus connus permettant de manipuler des bases de données MySQL.

Il est livré avec WAMP (plate-forme de développement WEB).

Je m'en suis servi créer la base de données, nommée « supervision_serre », que j'expose ci-dessus.

IV. Récupération et envoie des données

4.1) Choix de l'adaptateur pour la boucle 4-20 mA

Afin de mettre en place la boucle 4-20 mA, nous avions besoin de choisir un adaptateur ayant plusieurs canaux et qui s'adapte sur une carte Arduino. Deux adaptateurs sont ressortis de nos recherches :

Adaptateur 4-20 mA 1132_0

Nombre de canaux : 1
Livré avec câble de raccordement.
Température de service : -40°C à +85°C
Dimensions : 46 x 30 x 18 mm.
Module prêt à l'emploi.
Prix : 34€50 + 5€90 pour la livraison

Adaptateur 4-20 mA Current Loop Sensor Board

Nombre de canaux : 4
Livré avec câbles en paire torsadée.
Température de service : 0°C à +85°C
Dimensions : 73 ;5 x 51 x 13 mm.
Module prêt à l'emploi.
Prix : 78€00

L'adaptateur 4-20 mA 1132_0 n'a qu'un seul canal, sachant qu'on avait 3 capteurs à mettre en place sur la boucle 4-20 mA, il aurait fallu en acheter 3, ce qui aurait dépasser la contrainte budgétaire de 100€.

Notre choix s'est donc porté sur l'adaptateur 4-20 mA Current Loop Sensor Board car il rentre dans notre budget et il possède 4 canaux. Ce capteur était donc le mieux adapté pour mettre en place nos capteurs.

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

Pour mettre en place la boucle 4-20 mA nous avons attendu de recevoir l'adaptateur Current Loop Sensor Board. Une fois reçu, nous avons pu mettre en place la boucle. Pour le connecter à la carte Arduino Mega 2560, il suffisait de la placer dessus (shied). Toutes les informations pour mettre en place l'adaptateur était marquées sur le site sur lequel on l'a acheté. Afin de reproduire la boucle de courant nous avons suivi le schéma fonctionnel de celle-ci.

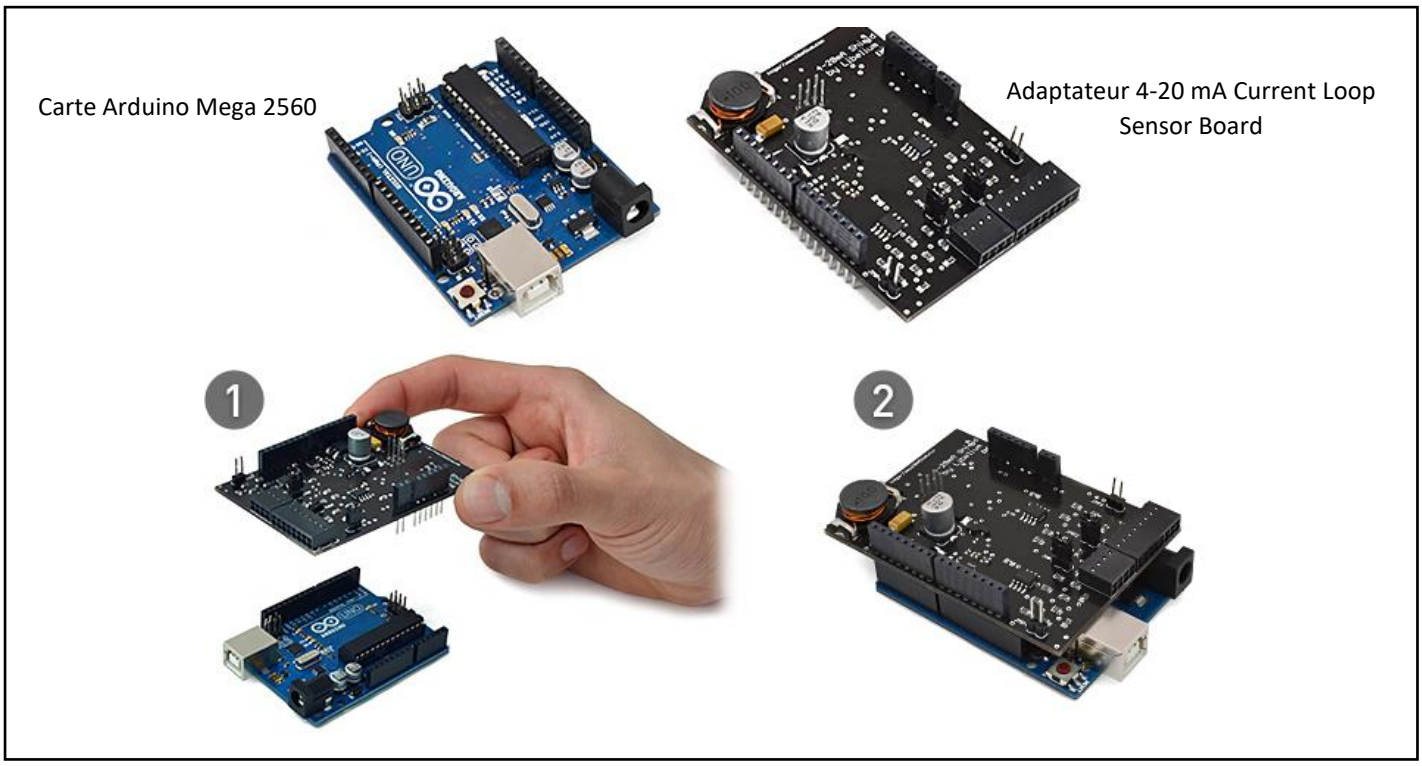


Schéma de connexion adaptateur/arduino

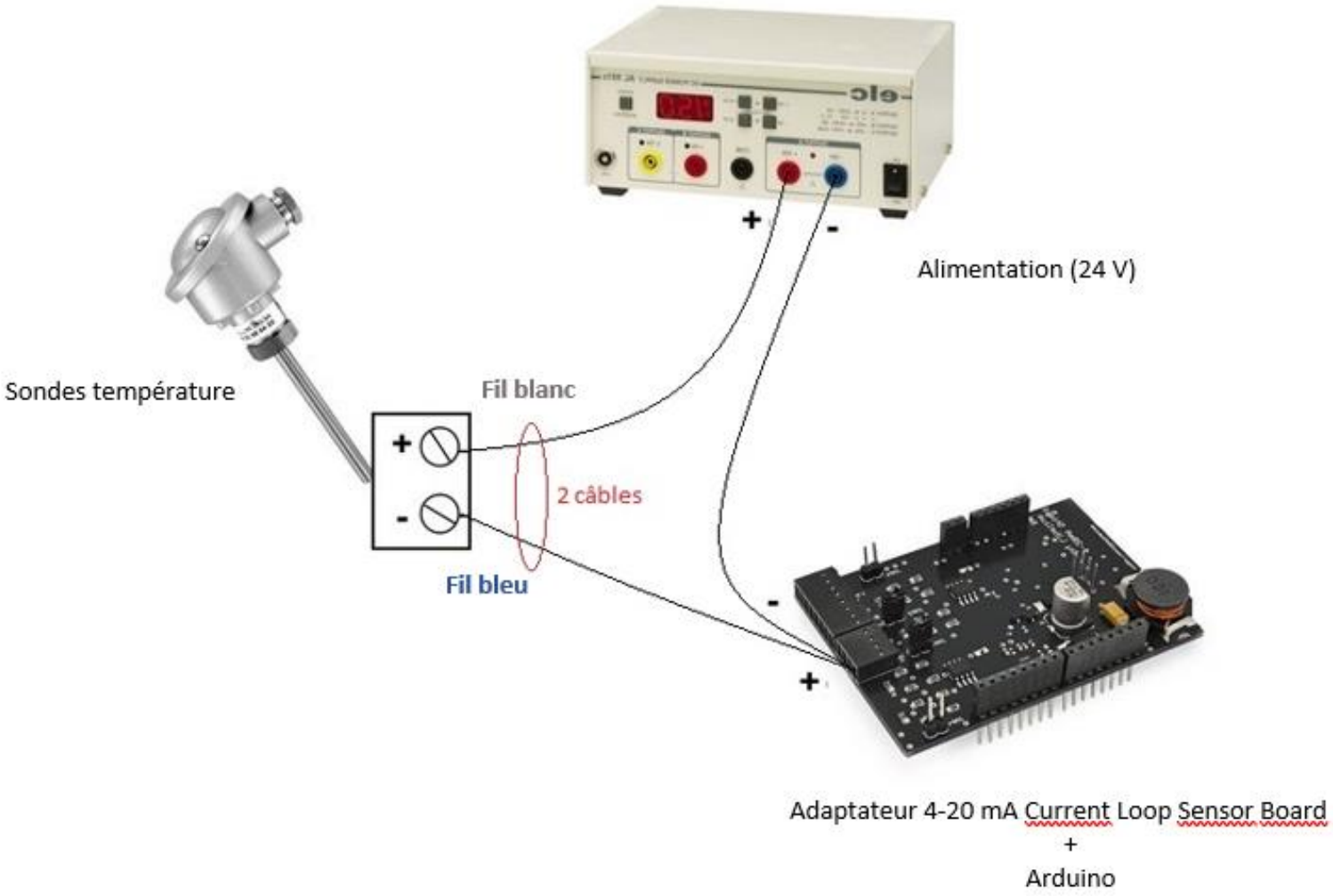
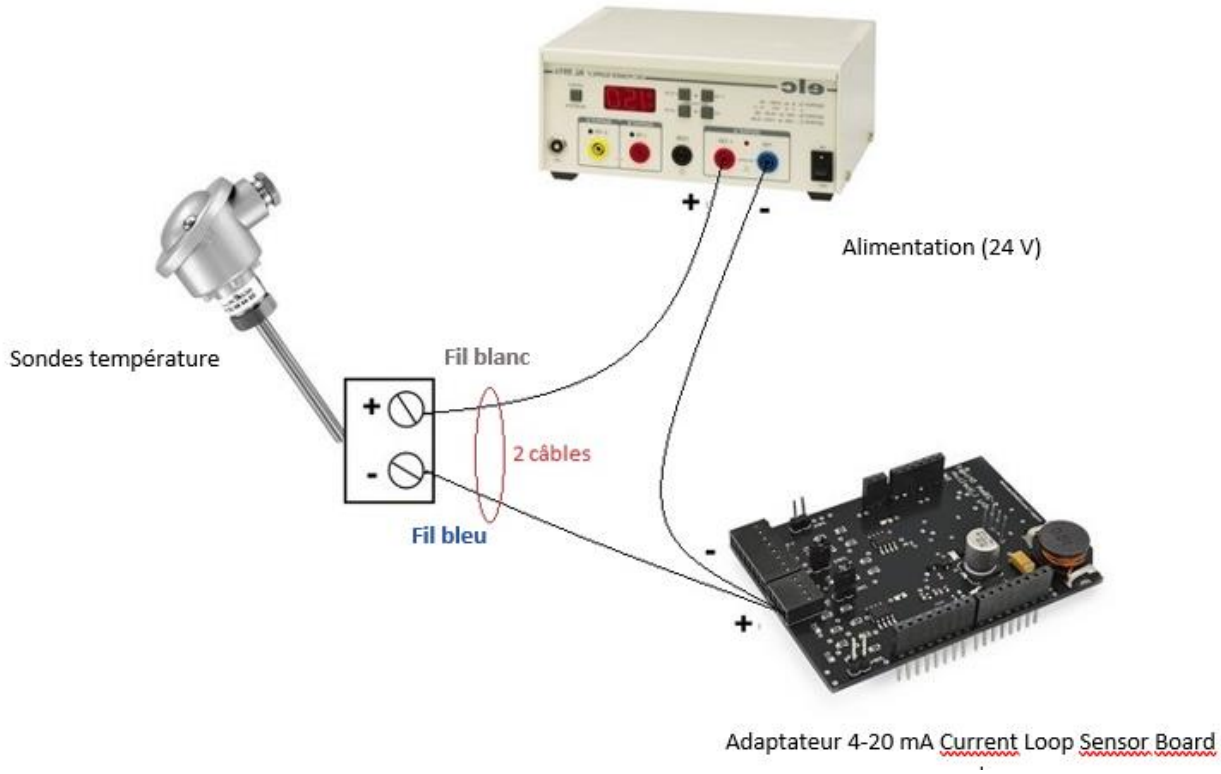
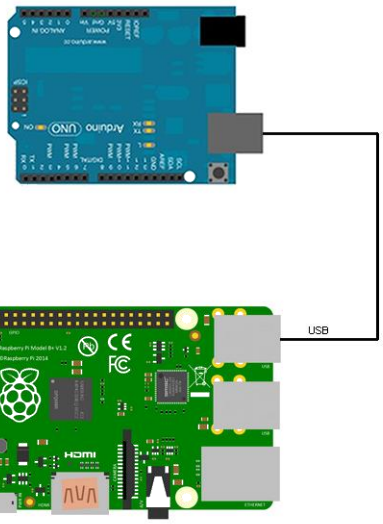


Schéma de câblage de la boucle 4-20 mA



Carte Arduino Mega 2560



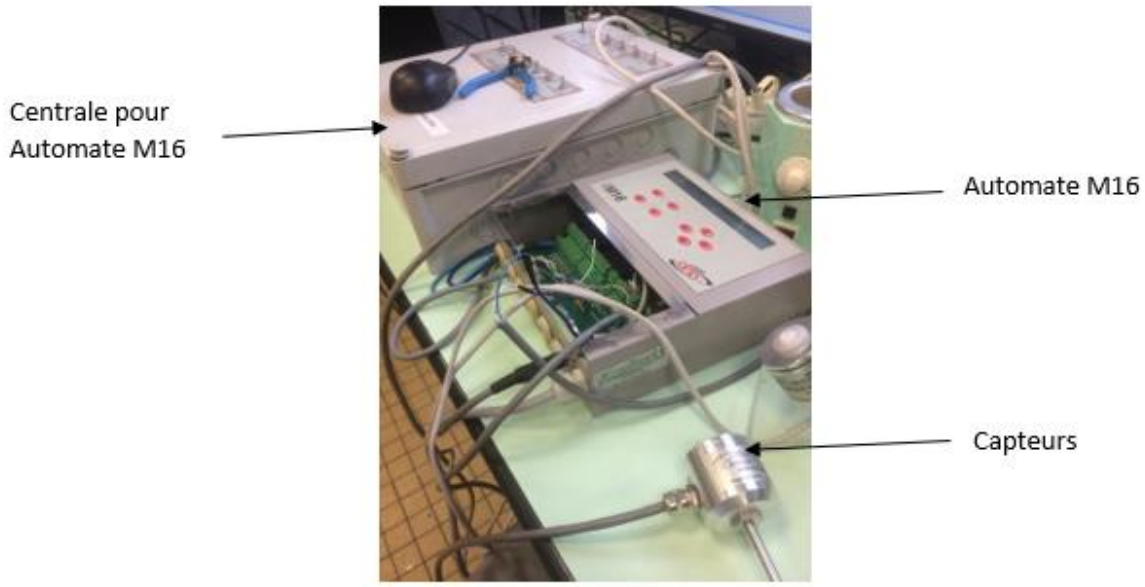
Carte Raspberry

Ensemble St Félix-La Salle Nantes

4.3) Test des sondes température

N'ayant pas reçu l'adaptateur pour mettre la boucle 4-20 mA en place directement, nous avons décidé de tester nos capteurs avec Willy (étudiant 2). Pour se faire nous nous sommes servis de l'ancien automate sur lequel les capteurs étaient mis en place. 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



Nous avons donc fait des phases de test à différentes températures :

Procédure de test à 0 minutes		
Capteurs	T_Intérieure	T_Tuyaux
Valeurs	23,6	20
Calculée	27	50
Puissance	4	6

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

On constate donc que les deux sondes de température fonctionnent bien, la sonde pour la température intérieure de la serre a bien fonctionnée lorsqu'on l'a chauffée (avec une simple pression de la main) et la sonde pour la température de l'eau des tuyaux de chauffage a également bien fonctionnée lorsqu'on l'a mise dans de l'eau chaude.

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

Afin de récupérer les données des capteurs sur la boucle 4-20 mA, il suffit de créer un code Arduino utilisant la bibliothèque Arduino fournit par le site sur lequel l'adaptateur a été acheter.

Lorsque l'on ouvre dans Arduino l'exemple fournit on obtient cela :

_4_20mA_03_several_sensors §

```
// Include this library for using current loop functions
#include <currentLoop.h>

#define TEMPERATURE CHANNEL1
#define HUMIDITY CHANNEL2


float current;

void setup()
{
  // Init Serial for viewing data in the serial monitor.
  Serial.begin(115200);
  delay(100);
  // Switch the 24V DC-DC converter
  sensorBoard.ON();
  delay(2000);
}
void loop()
{

  // Temprature sensor measure
  //=============================================================

  if (sensorBoard.isConnected(TEMPERATURE))
  {
    // Get the sensor value as a curren in mA
    current = sensorBoard.readCurrent(TEMPERATURE );
    Serial.print("Current value read from temperature sensor : ");
    Serial.print(current);
    Serial.println(" mA");
  }
  else {
    Serial.println("Temperature sensor is not connected...");
  }

  Serial.println("****************************************");
  Serial.print("\n");
  delay(100);
```

Lorsqu'on lance le programme avec toute la boucle branchée et mise en place, on obtient les mA envoyés par les capteurs. Il suffit donc de rajouter le produit en croix afin d'obtenir les valeurs dans l'unité que l'on veut, en l'occurrence, pour moi, des °C.

On obtient donc le code suivant :

```
#include <currentLoop.h>

#define TEMPERATURE_AIR CHANNEL1
#define TEMPERATURE_EAU CHANNEL3


float current;
float valeur;

void setup()
{
  // Init Serial for viewing data in the serial monitor.
  Serial.begin(115200);
  delay(100);
  // Switch the 24V DC-DC converter
  sensorBoard.ON();
  delay(2000);
}
void loop()
{

  // Temprature intérieur
  //============================================================

  if (sensorBoard.isConnected(TEMPERATURE_AIR))
  {
    // Get the sensor value as a curren in mA
    current = sensorBoard.readCurrent(TEMPERATURE_AIR );
    Serial.print("La valeur du capteur temperature de l'air est : ");
    Serial.print(current);
    Serial.println(" mA");
    valeur = ((current-4)*45)/16;
    Serial.print("La température de l'air est de");
    Serial.print(valeur);
    Serial.println(" °C");
  }
  else {
    Serial.println("Il n'y a pas de capteur connecté au port 1..");
  }
```

```
  // Temprature eau des tuyaux
  //=========================================================

  delay(100);

  if (sensorBoard.isConnected(TEMPERATURE_EAU))
  {
    // Get the sensor value as a curren in mA.
    current = sensorBoard.readCurrent(TEMPERATURE_EAU);
    Serial.print("La valeur du capteur temperature de l'eau est : ");
    Serial.print(current);
    Serial.println(" mA");
    valeur = ((current-4)*100)/16;
    Serial.print("La température de l'eau est de : ");
    Serial.print(valeur);
    Serial.println(" °C");
  }
  else {
    Serial.println("Il n'y a pas de capteur connecté au port 3..");
  }


  Serial.println("****************************************");
  Serial.print("\n");

  delay(3000);
}
```

Pour la température intérieure :

On sait que la sonde a une plage de température allant de 0 à 45 °C. Or on est sur une boucle 4-20 mA, le 4 mA correspondant donc à 0 °C et le 20 mA correspondant à 45°C. Afin de faire le produit en croix, on applique une soustraction de 4 à la valeur reçue en mA.

Prenons l'exemple de la valeur 14.6. C'est la valeur reçue en mA ici dans le code nommée « current ». On attribut donc à la variable « valeur » le calcul suivant : `valeur = ((current-4)*45)/16;`

Une valeur de 14.6 reçue correspond donc à environ 29.8 °C.
On obtient donc grâce à ce calcul la valeur qu'envoie le capteur en °C.

Pour la température de l'eau des tuyaux de chauffage :

Pour la sonde qui mesure la température de l'eau, c'est le même principe. On sait que se plage de température est de 0 à 100 °C. On applique donc le produit en croix avec ces valeurs-là, ce qui donne le calcul suivant : `valeur = ((current-4)*100)/16;`

Une valeur de 14.6 sur ce capteur correspond donc à 66.25 °C.
On obtient donc également grâce à ce calcul la valeur reçue en °C.

Le programme ainsi codé fonctionne.

Lorsque l'on le lance on obtient donc :

```
La valeur du capteur temperature de l'air est : 14.6 mA
La temperature de l'air est de 29.8 °C

La valeur du capteur temperature de l'eau est : 14.6 mA
La température de l'eau est de : 66.25 °C
```

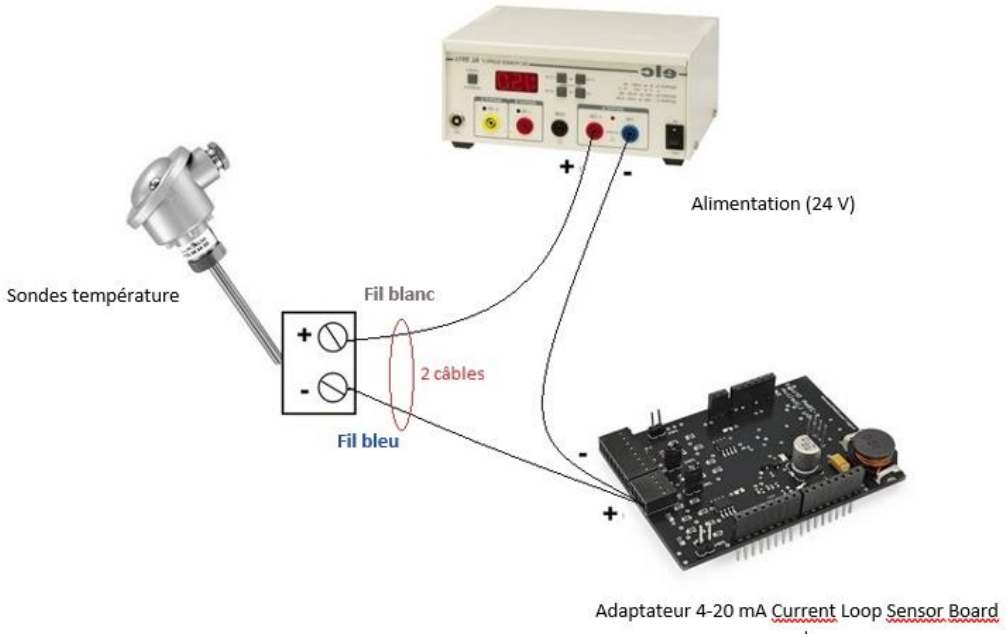
Et si l'un des capteurs n'est pas connecté :

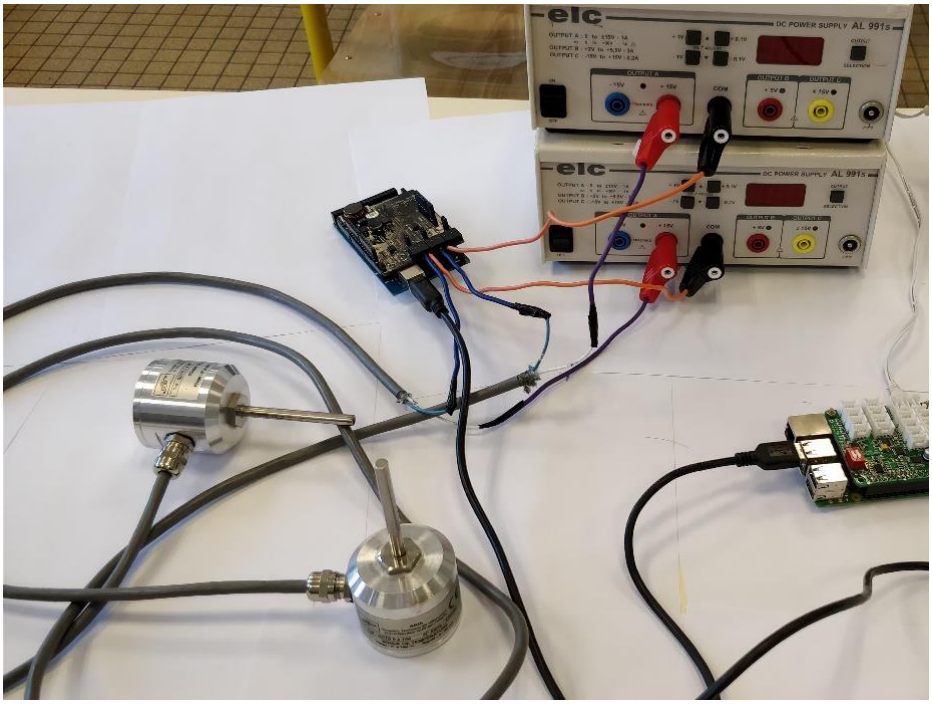
```
Il n'y a pas de capteur connecté au port 1..
```

Le port sans capteur est précisé.

Afin d'envoyer ces données sur la base de données, il suffira de coder la méthode mesurer() du programme de Steven (étudiant 1) pour qu'elle récupère ce que l'Arduino reçoit. Par manque de temps, Steven n'a pas encore fini son programme afin qu'on puisse coder cette méthode. Je n'ai pas mis tout le code ci-dessus, car je mettrais le test unitaire dans la partie 6.

Schéma de câblage final :





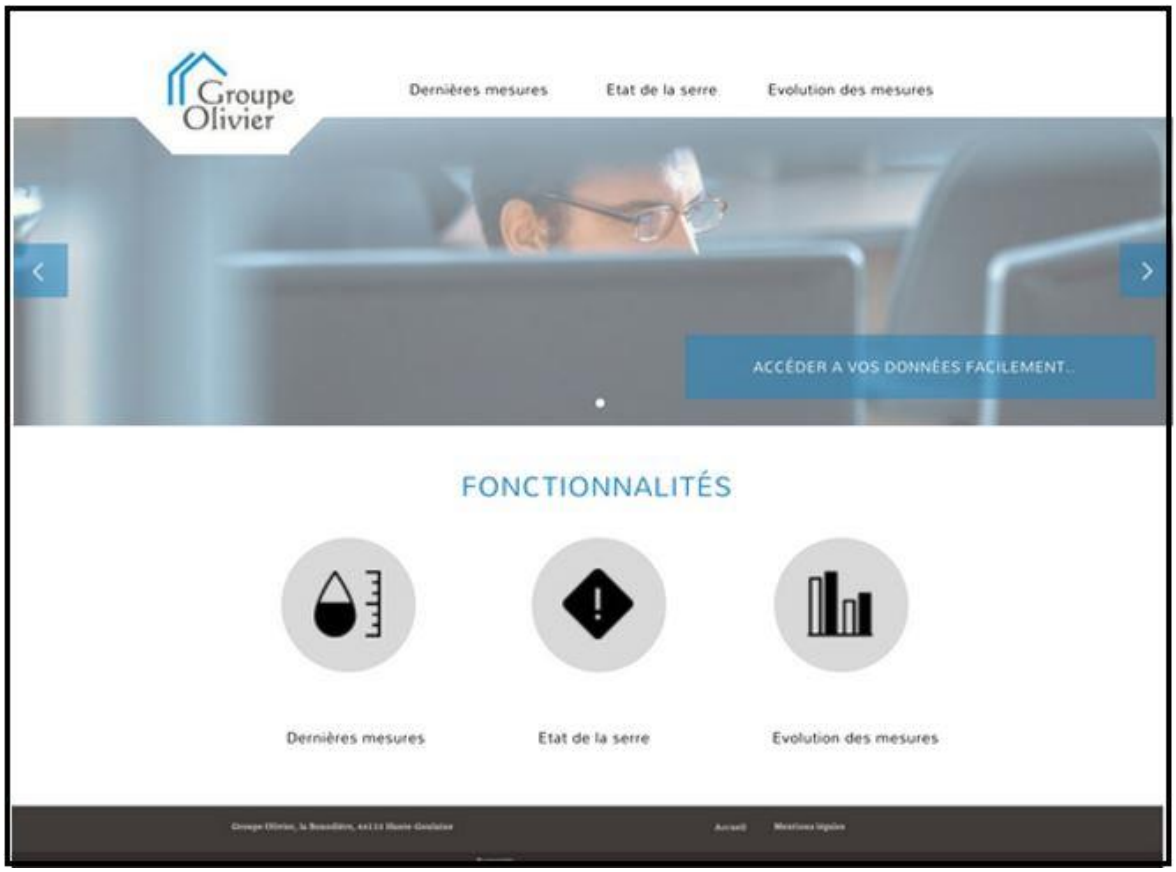
V. Mise en place de l'application web

5.1) Conception de la charte graphique

Pour la charte graphique du site web, nous nous sommes inspirés du site web actuel du Groupe Olivier, entreprise pour laquelle nous faisons le projet.

Nous nous sommes servit du site Canva, qui est un site pour créer facilement des designs, afin de préparer les pages du site en avance et ainsi savoir comment réaliser le HTML et le CSS du site.

Page d'accueil :



C'est la page d'accueil du site, on peut naviguer sur les différentes pages grâce à la barre de navigation mas également en cliquant sur les icônes présentes sous le titre « Fonctionnalités ».

Page évolution des mesures :

Cette page concerne la partie de Willy, l'étudiant 2. Elle affichera les données d'un capteur sur une période définie, sous forme de graphique.

On y voit deux calendriers qui serviront à choisir la période et au-dessus la courbe

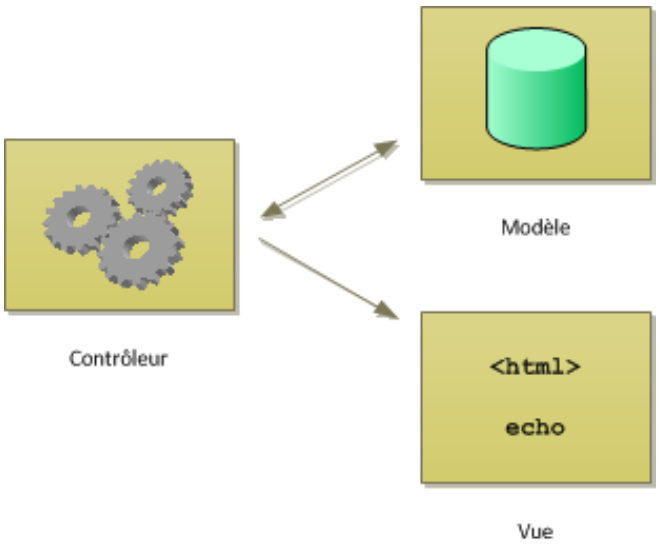
Page état de la serre :

Cette page concerne ma partie application web. On y voit une serre en fond. Cette page affichera les dernières données acquises par chaque capteur sous forme de tableau.

5.2) Architecture de l'application

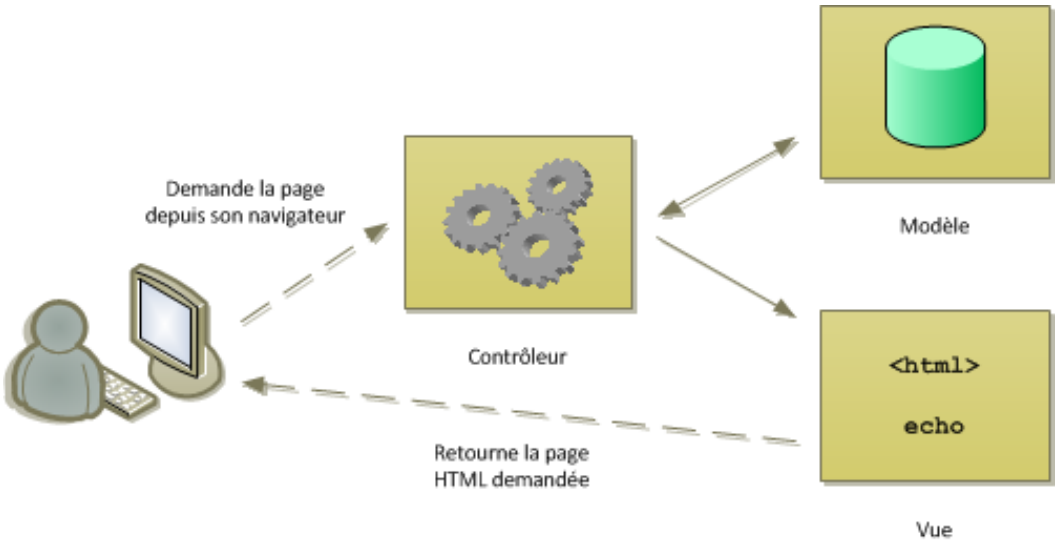
En ce qui concerne l'application, il nous a été imposé de suivre le Modèle-Vue-Contrôleur. Ce modèle sert à avoir un code bien organiser. Pour se faire, le code est divisé en trois grandes parties :

- **Modèle** : Cette partie va chercher les informations dans la base de données. Elle gère les données du site et comprend principalement les requêtes SQL.
- **Vue** : C'est la partie affichage. Elle comporte essentiellement du code PHP et HTML. Dans notre projet, c'est un script.
- **Contrôleur** : C'est la partie qui gère la logique du code. Cette partie fait le lien entre le modèle et la vue. Elle demande les données au Modèle, les analyse et envoie ensuite le texte à afficher à la Vue.



Echanges entre les éléments

L'utilisateur du site web demandera la page au contrôleur et c'est la vue qui lui sera retournée.



Résultat lorsqu'un utilisateur demande une page

5.3) Connexion à la base de données

La connexion à la base de données est effectuée dans un fichier connect.php situé dans un dossier inc. Afin d'effectuer la connexion à la base de données, nous avons des informations à renseigner.

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

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

→ Adresse IP de la base de données

→ Login et mot de passe pour l'utilisateur de la base de données

→ Nom de la base de données

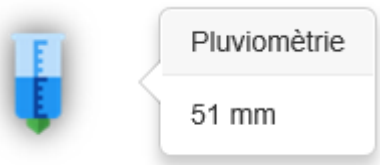
5.4) Affichage de l'état actuel de la serre

5.4.1) Première version de la page web

Comme je le disais au début de ce dossier, mon rôle était de faire une page affichant les dernières mesures en temps réel. J'étais partis sur une idée d'icones placées sur la serre sur lesquelles il nous suffit de cliquer pour affichant les mesures.

Serre avec les icones affichant les mesures

J'avais donc mis en place le système de popover sur chacun des icones et je récupérais les données de la base de données avec des requêtes précises à chaque relevé.



Résultat obtenu pour le pluviomètre

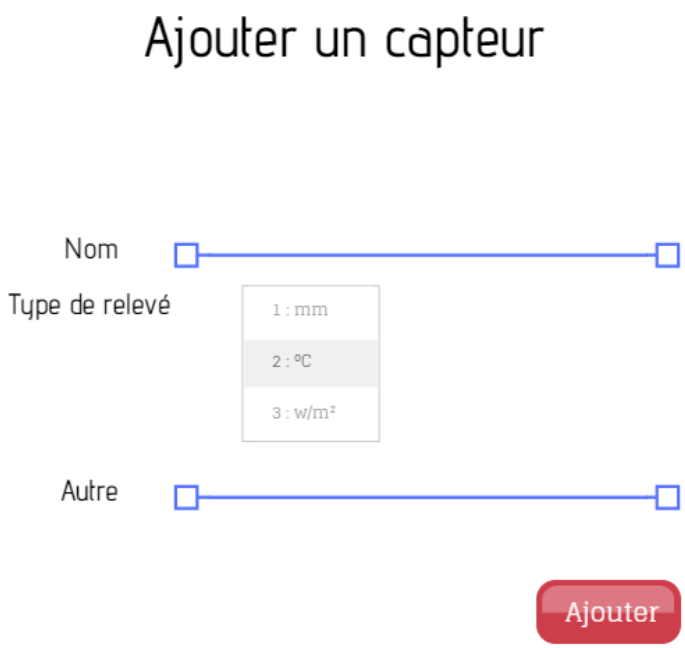
```php
$pluviometres=$bdd->query("SELECT * FROM `releve` WHERE id_materiel=1 ORDER BY `releve`.`date_releve` DESC LIMIT 1");
        $pluviometre = $pluviometres->fetch();
        $tab[] =  $pluviometre['valeur']; // On récupère les données sous forme de tableau, pour récupérer les valeurs.
```

Requête pour obtenir le relevé

Ainsi, chaque capteur avait son icone avec le popover qui affichait le relevé. Je n'avais malheureusement pas pris en compte la contrainte qui est de pouvoir facilement ajouter un capteur pour l'exploitant. Cela est malheureusement trop compliqué à mettre en place avec les popovers, j'ai donc abandonné cette idée après avoir passer un certain temps dessus.

5.4.2) Version finale de la page web

Ayant perdu du temps sur cette histoire de popover, la page finale du site web n'est pas terminée à ce jour. J'ai tout de même trouvé mon idée pour pouvoir ajouter facilement un capteur. Je vais afficher les relevés sous forme d'un tableau qui sera en avant de la serre (qui sera mise plus transparente). Afin de faciliter l'ajout de nouveau capteur, on créera avec Willy, une troisième page sur le site Web qui sera nommée « Ajout d'un capteur ». On ne l'a pas du tout commencé, mais on pense faire une page dans ce style :



Avec un champ pour ajouter le nom du capteur, un pour sélectionner le type de relevé parmi ceux déjà présent dans la base de données (non présents sur le schéma) et enfin un autre champ pour ajouter un nouveau type de capteur. Il n'aura ensuite plus qu'à appuyer sur le bouton ajouter et tout est ajouter dans la base de données et sur ma page.

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)
```

Pour se faire je ferais, comme je le disais, un tableau dans lequel j'afficherais les relevés. J'ai, pour l'instant, effectué une requête qui récupère dynamiquement les derniers relevés présents dans la base donnée. J'obtiens donc les dernières valeurs de relevés comme ceci :

```
array (size=30)
  0 =>
    array (size=8)
      'valeur' => string '58' (length=2)
      0 => string '58' (length=2)
      'id_unite' => string '1' (length=1)
      1 => string '1' (length=1)
      'date_releve' => string '2018-03-31 16:43:58' (length=19)
      2 => string '2018-03-31 16:43:58' (length=19)
      'id_materiel' => string '1' (length=1)
      3 => string '1' (length=1)
  1 =>
    array (size=8)
      'valeur' => string '55' (length=2)
      0 => string '55' (length=2)
      'id_unite' => string '3' (length=1)
      1 => string '3' (length=1)
      'date_releve' => string '2018-03-29 00:00:00' (length=19)
      2 => string '2018-03-29 00:00:00' (length=19)
      'id_materiel' => string '2' (length=1)
      3 => string '2' (length=1)
```

Grâce à cette requête :

```php
function getreleve($dbc) {
    $request = $dbc->prepare ("SELECT valeur, id_unite, date_releve, id_materiel FROM releve");
    return $request->execute() ? $request->fetchAll() : null;
}

$getreleves = getreleve($bdd);
var_dump($getreleves);
```

Par manque de temps, je n'ai pas encore finalisé la page en mettant ces données dans un tableau et en les intégrant au site final sur la page « Etat de la serre ».

VI. Tests unitaires

6.1) Test unitaire de la méthode get_current_eau()

La méthode get_current_eau() est présente dans la programme Arduino qui permet de récupérer les valeurs envoyées par les capteurs dans la boucle 4-20 mA.

Cette méthode sert à récupérer la valeur envoyée par le capteur de température de l'eau des tuyaux en mA. Dans une boucle 4-20 mA, le 4 mA correspond à la valeur la plus basse du capteur, en l'occurrence 0 °C et le 20 mA correspond à la valeur maximale, c'est-à-dire 100 °C.

Lorsque le capteur envoie une valeur en mA, elle doit obligatoirement être entre 4 et 20 mA. Ainsi, mon test unitaire vérifie que la valeur reçue est bien entre 4 et 20 mA.

Pour se faire j'utilise la librairie ArduinoUnit, qui est faite pour les tests unitaires sous Arduino.

Je l'inclus donc dans le programme :

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

#include <currentLoop.h>
```

Ensuite je code mon test unitaire :

```
//test unitaire ma temperature eau
test(current_eau)
{
  float current=get_current_eau;
  int max_ma=20;
  assertLessOrEqual(current,y);
}
```

Ici je vérifie si la valeur retournée par get_current_eau, qui est la valeur en mA que le capteur vient d'envoyer, est bien inférieure à 20 mA.

```
test(current_eau)
{
  float current=get_current_eau;
  int min_ma=4;
  assertMoreOrEqual(current,min_ma);
}
```

Ici je vérifie si la valeur retournée par get_current_eau, qui est la valeur en mA que le capteur vient d'envoyer, est bien supérieure à 4 mA.

```
void setup()
{
  Serial.begin(9600);
  while(!Serial) {} // Portability for Leonardo/Micro
}
```

Ici, je définis le débit de la communication série à 9600 Bauds.

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

Enfin, je lance ici le test unitaire.

Elément testé :	Test de la méthode get_current_eau() du programme Arduino		
Objectif du test :	Vérifier le bon fonctionnement du capteur et de la boucle 4-20 mA		
Nom du testeur :	Samuel GERARD		
Moyens mis en œuvre :	Logiciel : Arduino	Matériel : Montage Capteur et boucle 4-20 mA, ordinateur	Outil de développement : Arduino, ArduinoUnit

Procédure de test			
Description du vecteur de test	**Résultat attendu**	**Résultat obtenu**	**Validation (O/N)**
Tester current_eau_max.	Le test est bon.	`Test current_eau_max passed.`	O
Tester current_eau_min.	Le test est bon.	`Test current_eau_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 entre 4 et 20 mA.		

VII. 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 très enrichissante sur le point de vue technique et personnel. En effet, j'avais déjà une connaissance du langage HTML et PHP, que j'ai pu renforcer en créant avec Willy ce site web. De plus j'ai découvert l'univers industriel en devant mettre en place cette boucle 4-20 mA et les sondes température. J'ai également découvert des outils logiciels et matériels que je ne connaissais pas et cela m'a permis de m'enrichir à ce niveau.

Durant ce projet, j'ai découvert un univers qui m'a beaucoup plu et qui je pense ouvre beaucoup de portes, l'univers de l'informatique industrielle. Devoir mettre en place ces capteurs ainsi que de devoir gérer et créer une base de données en conséquence m'a enrichi en connaissances. J'ai également découvert la boucle de courant 4-20 mA qui m'état inconnue jusqu'à aujourd'hui.

Pour réussir à bien avancer, on a formé des liens solides avec les membres du groupe de projet, plus particulièrement avec Willy, avec qui j'ai passer plus de la moitié du temps afin de mener à bien nos tâches en commun. Ce travail en équipe fut une expérience enrichissante du point de vue sociale comme professionnel.

Finalement, sur le plan personnel, toutes les connaissances acquises durant le projet constitueront un bagage solide pour les années à venir.

VIII. Annexe

Code Arduino pour les sondes température

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

#include <currentLoop.h>

#define TEMPERATURE_AIR CHANNEL1
#define TEMPERATURE_EAU CHANNEL3

float current;
float valeur;

void setup()
{
  // Init Serial for viewing data in the serial monitor.
  Serial.begin(115200);
  delay(100);
  // Switch the 24V DC-DC converter
  sensorBoard.ON();
  delay(2000);
}
void loop()
{

  // Mesure température de l'air
  //============================================================

  if (sensorBoard.isConnected(TEMPERATURE_AIR))
  {
    // Get the sensor value as a curren in mA
    current=get_current_air;
    Serial.print("La valeur du capteur temperature de l'air est : ");
    Serial.print(current);
    Serial.println(" mA");
    valeur=get_temperature_air;
    Serial.print("La température de l'air est de");
    Serial.print(valeur);
    Serial.println(" °C");
  }
  else {
    Serial.println("Il n'y a pas de capteur connecté au port 1..");
  }
```

```
  // mesure température eau
  //=====================================================

  delay(100);

  if (sensorBoard.isConnected(TEMPERATURE_EAU))
  {
    current=get_current_eau;
    Serial.print("La valeur du capteur temperature de l'eau est : ");
    Serial.print(current);
    Serial.println(" mA");
    valeur=get_temperature_eau;
    Serial.print("La température de l'eau est de : ");
    Serial.print(valeur);
    Serial.println(" °C");
  }
  else {
    Serial.println("Il n'y a pas de capteur connecté au port 3..");
  }


  Serial.println("**************************************");
  Serial.print("\n");

  delay(3000);
}
//obtention valeur mA capteur
float get_current_air() {
  current=0;
  if (sensorBoard.isConnected(TEMPERATURE_AIR))
  {
    // Obtenir la valeur du capteur en mA
    current = sensorBoard.readCurrent(TEMPERATURE_AIR );
  }
  return current;
}

//obtention valeur mA capteur
float get_current_eau() {
  current=0;
  if (sensorBoard.isConnected(TEMPERATURE_EAU))
  {
    // Obtenir la valeur du capteur en mA
    current = sensorBoard.readCurrent(TEMPERATURE_EAU);
  }
  return current;
}
```

```
//obtention valeur °C
float get_temperature_air() {
  valeur=0;
  if (sensorBoard.isConnected(TEMPERATURE_AIR))
  {
    // Calcul pour obtenir la valeur en degrés
    valeur = ((get_current_air-4)*45)/16;
  }
  return valeur;
}

//obtention valeur °C
float get_temperature_eau() {
  valeur=0;
  if (sensorBoard.isConnected(TEMPERATURE_EAU))
  {
    // Calcul pour obtenir la valeur en degrés
    valeur = ((get_current_eau-4)*100)/16;
  }
  return valeur;
}
//test unitaire mA température air
test(current_air_max)
{
  float current=get_current_air;
  int max_ma=20;
  assertLessOrEqual(current,y);
}

test(current_air_min)
{
  float current=get_current_air;
  int min_ma=4;
  assertMoreOrEqual(current,min_ma);
}

//test unitaire ma temperature eau
test(current_eau_max)
{
  float current=get_current_eau;
  int max_ma=20;
  assertLessOrEqual(current,y);
}
```

```
test(current_eau_min)
{
  float current=get_current_eau;
  int min_ma=4;
  assertMoreOrEqual(current,min_ma);
}

void setup()
{
  Serial.begin(9600);
  while(!Serial) {} // Portability for Leonardo/Micro
}

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