



*Etudiant 3 : Création base de données*

*Mise en place sondes température*

*Mise en place page web*

Dossier technique du projet – Partie personnelle

Sommair

[I. Situation dans le projet 3](#_Toc514244191)

[1.1) Synoptique de la réalisation 3](#_Toc514244192)

[1.2) Rappel des tâches de l’étudiant 3](#_Toc514244193)

[1.3) Contraintes liées au développement 4](#_Toc514244194)

[II. Conception et mise en œuvre 5](#_Toc514244195)

[2.1) Fonctionnement des sondes température 5](#_Toc514244196)

[2.2) Fonctionnement de la boucle 4-20 mA 6](#_Toc514244197)

[2.3) Réalisation du diagramme de classe 7](#_Toc514244198)

[III. Mise en place de la base de données 9](#_Toc514244199)

[3.1) Modèles entité association base de données 9](#_Toc514244200)

[3.2) Description des différentes tables 10](#_Toc514244201)

[IV. Récupération et envoie des données 12](#_Toc514244202)

[4.1) Choix de l’adaptateur pour la boucle 4-20 mA 12](#_Toc514244203)

[4.2) Mise en place de la boucle 4-20 mA 13](#_Toc514244204)

[4.3) Test des sondes température 15](#_Toc514244205)

[4.4) Récupérer les données d’un capteur sur la boucle 4-20 mA 15](#_Toc514244206)

[V. Mise en place de l’application web 16](#_Toc514244207)

[5.1) Conception de la charte graphique 16](#_Toc514244208)

[5.2) Architecture de l’application 16](#_Toc514244209)

[5.3) Connexion à la base de données 16](#_Toc514244210)

[5.4) Affichage de l’état actuel de la serre 16](#_Toc514244211)

[5.4.1) Première version de la page web 16](#_Toc514244212)

[5.4.2) Version finale de la page web 16](#_Toc514244213)

[VI. Tests unitaires 16](#_Toc514244214)

[6.1) Test unitaire de la méthode Loop() 16](#_Toc514244215)

[VII. Conclusion 16](#_Toc514244216)

[VIII. Annexes 16](#_Toc514244217)

# I. Situation dans le projet

## Une image contenant carte, texte Description générée avec un niveau de confiance très élevé1.1) 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



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 a 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.

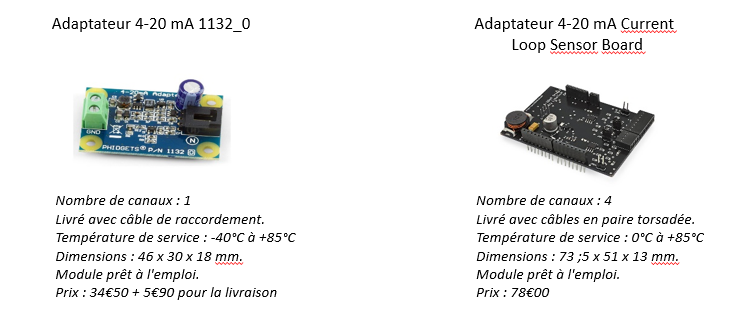
Il est livré avec WAMP (plate-forme de développement WEB).

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 :

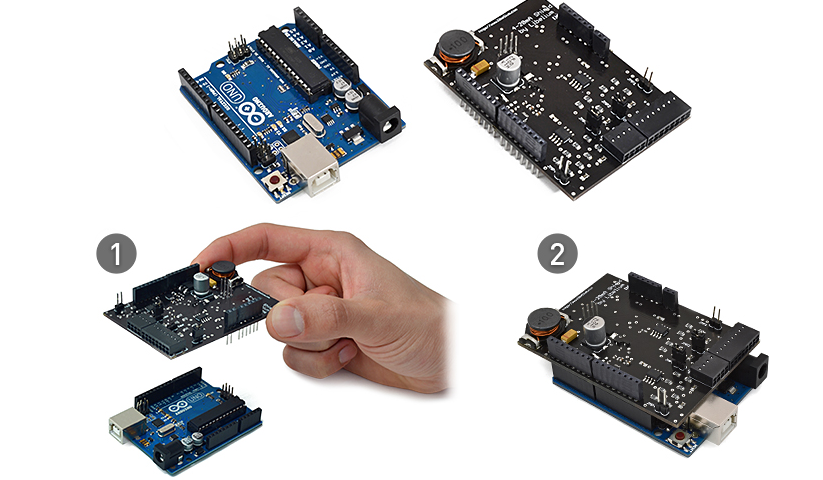


L’adaptateur 4-20 mA 1132\_0 n’a qu’un seul canal, sachant qu’on avait 3 capteurs à mettre en place sur la boucle 4-20 mA, il aurait fallu en acheter 3, ce qui aurait dépasser la contrainte budgétaire de 100€.

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

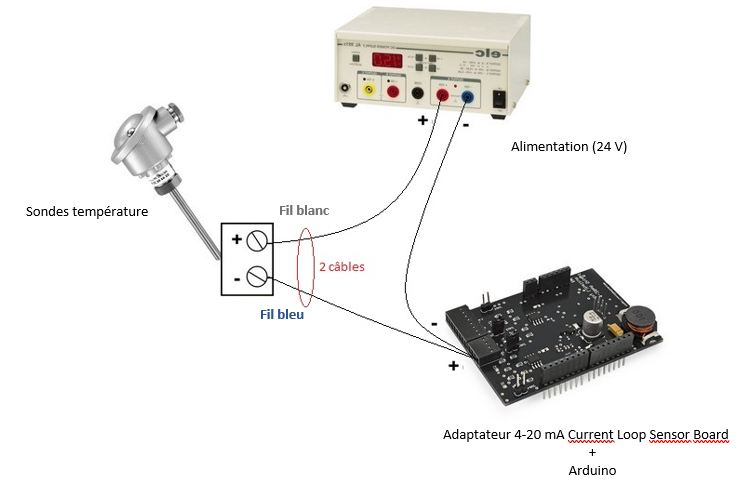
Pour mettre en place la boucle 4-20 mA nous avons attendu de recevoir l’adaptateur Current Loop Sensor Board. Une fois reçu, nous avons pu mettre en place la boucle. Pour le connecter à la carte Arduino Mega 2560, il suffisait de la placer dessus (shied). Toutes les informations pour mettre en place 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.



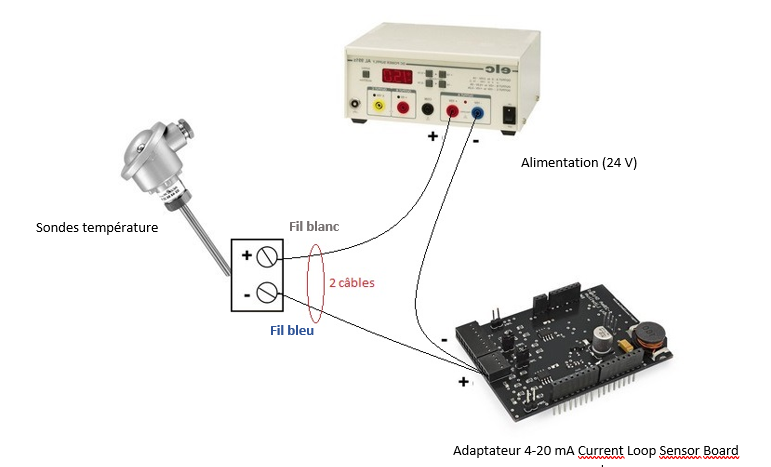
Carte Arduino Mega 2560

Adaptateur 4-20 mA Current Loop Sensor Board

*Schéma de connexion adaptateur/arduino*



*Schéma de câblage de la boucle 4-20 mA*



Une image contenant équipement électronique, circuit

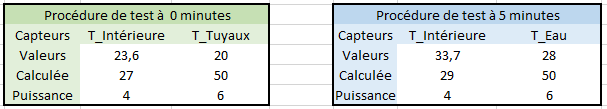
Description générée avec un niveau de confiance très élevé

Carte Arduino Mega 2560

Carte Raspberry

## 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’ancienne machine sur laquelle les capteurs étaient mis en place. Nous avons donc fait des phases de test à différentes températures :

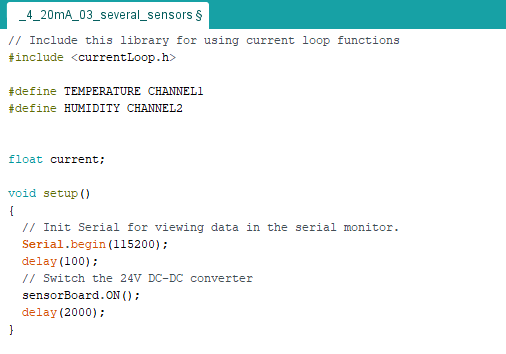


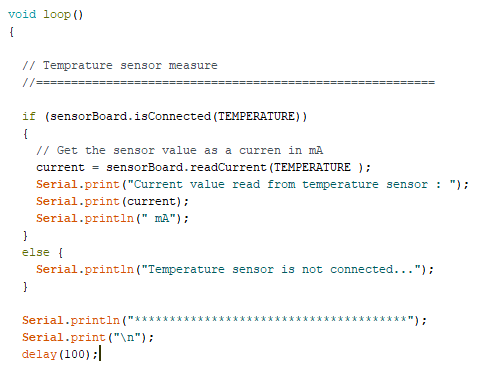
On constate donc que les deux sondes de température fonctionnent bien, la sonde pour la température intérieure de la serre a bien fonctionnée lorsqu’on l’a chauffée (avec une simple pression de la main) et la sonde pour la température de l’eau des tuyaux de chauffage a également bien fonctionnée lorsqu’on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

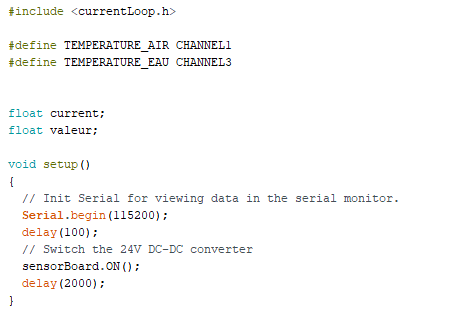
Lorsque l’on ouvre dans Arduino l’exemple fournit on obtient cela :





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 :**







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 :

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 :

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

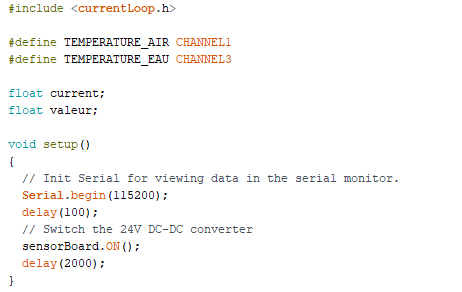
La température de l'eau est de : 66.25 °C

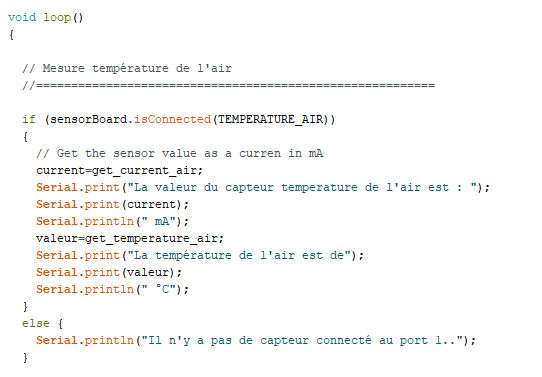
Et si l’un des capteurs n’est pas connecté :

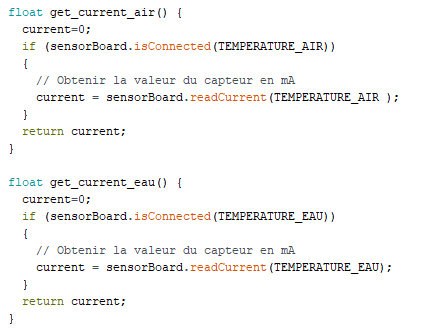
Il n'y a pas de capteur connecté au port 1..

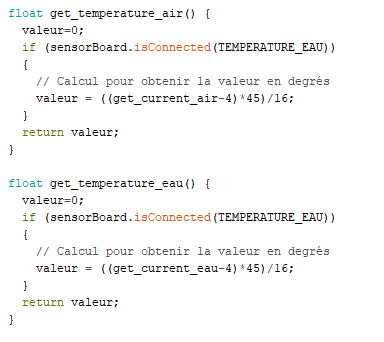
Le port sans capteur est précisé.

**Code final :**









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.

# V. Mise en place de l’application web

## 5.1) Conception de la charte graphique

## 5.2) Architecture de l’application

## 5.3) Connexion à la base de données

## 5.4) Affichage de l’état actuel de la serre

### 5.4.1) Première version de la page web

### 5.4.2) Version finale de la page web

# VI. Tests unitaires

## 6.1) Test unitaire de la méthode Loop()

# VII. Conclusion

# VIII. Annexes