



Brevet de Technicien
Supérieur
SNIR

DOSSIER FINAL SUPERVISION DE SERRE



VILLERMIN Maxime
ROCHER Marvyn
MICHAUD Théo
AUVÉ Killian



Sommaire

A - Cahier des charges.....	12
1. Présentation et situation du projet dans son environnement.....	12
1. Contexte de réalisation.....	12
2. Présentation du projet.....	13
3. Situation du projet dans son contexte.....	13
4. Cahier des charges – Expression du besoin.....	14
2. Spécifications.....	18
1. Diagrammes UML.....	18
2. Contraintes de réalisation.....	20
3. Ressources mises à disposition des étudiants (logiciels / matériels / documents).....	22
3. Répartition des fonctions ou cas d'utilisation par étudiant.....	23
4. Exploitation Pédagogique - Compétences terminales évaluées.....	24
5. Planification (Gantt).....	25
6. Condition d'évaluation pour l'épreuve E6-2.....	26
7. Observation de la commission de Validation.....	27
1. Avis formulé par la commission de validation.....	27
2. Nom des membres de la commission de validation académique.....	27
3. Visa de l'autorité académique.....	28
B- Analyse.....	29
I/ Analyse du besoin.....	29
1. La serre.....	29
2. La demande.....	30
3. Synoptique.....	31
4. Spécifications fonctionnelles.....	32
4.1 Catalogue des acteurs.....	32
4.2 Diagramme des cas d'utilisation.....	33
II/ Les contraintes.....	34
III/ Nos choix concernant le développement du projet.....	35

1. Choix liés au développement du site.....	35
1. Langages utilisés.....	35
2. <i>Choix de l'EDI</i>	35
2. Choix liés à l'application mobile.....	36
1. Langages utilisés.....	36
2. Choix de l'EDI.....	36
3. Choix du système d'exploitation.....	36
3. Choix liés au contrôleur de serre.....	37
1. Objet connecté ESP32.....	37
4. Choix liés au contrôleur de vannes.....	38
1. Le boîtier.....	38
2. Langages utilisés.....	38
3. Choix de l'EDI.....	38
5. Analyse des besoins capteurs.....	39
6. Choix liés à la gestion des alertes.....	40
7. Coût total.....	41
IV/ Base de données.....	42
V/ Plan du site.....	43
C - Conception préliminaire.....	44
I/ Conception préliminaire : Étudiant 1, MICHAUD Théo.....	44
1. Préambule.....	44
2. Tâche n°1 « Commander l'arrosage localement ».....	45
3. Tâche n°2 : « Contrôler les vannes ».....	47
4. Choix technologiques.....	51
1. Choix du boîtier - eZyvalve4 - Antelco.....	51
2. La carte de développement : l'ESP32.....	52
3. Choix de la carte de commande - L298N.....	52
4. Schéma de câblage sur notre ESP32.....	54
5. Tâche n°3 : « Mettre à jour les cycles d'arrosage et les seuils ».....	55
6. Tâche n°4 : « Mesurer l'humidité du sol ».....	56
7. Choix du capteur d'humidité : YL-69.....	58

Schéma de câblage pour programmer ce capteur.....	59
9. Tâche n°5 : «Mesurer le débit d'eau».....	60
10. Choix du capteur - YF-B5.....	61
11. Choix Logiciel.....	62
1. Arduino.....	62
2. AppInventor.....	62
3. Fritzing :.....	64
II/- Conception préliminaire : Étudiant 2, AUVÉ Killian.....	65
1. Préambule.....	65
2. Tâche n°1 « Mesurer la température, l'humidité de l'air et l'ensoleillement ».66	66
3. Choix des technologies.....	67
1. Capteur de température, d'humidité : BME280.....	67
2. Capteur de luminosité : TSL2591.....	68
3.Schéma de câblage du TSL2591.....	69
4. Principes physiques.....	70
1. Température.....	70
2. Hygrométrie.....	70
3.Luminosité.....	70
5.Module RTC : DS3231SN.....	71
6. Bus I2C.....	72
7. Tâche n°2 « Envoyer les données de la serre ».....	73
8. Choix des technologies.....	76
1. Stockage données:Shield SD.....	76
2. Schéma de câblage avec l'ESP32.....	77
3. Bus SPI.....	78
9. Interaction avec un Shield SD.....	79
1. Inclusion des bibliothèques.....	79
2. Définition de la broche.....	79
3. Vérification de l'existence d'un fichier.....	79
4. Ouverture d'un fichier.....	79
5. Bibliothèques Wi-Fi.....	80

10. Tâche n°3 « Mettre à jour la base de données ».....	81
1. Les tables nécessaires dans la base de données.....	82
2. Description des tables nécessaires.....	83
III/- Conception préliminaire : Étudiant 3, VILLERMIN Maxime.....	85
1. Préambule.....	85
2. Le site web.....	86
3. Les outils de développement.....	88
4. Choix et présentations des technologies.....	89
1. Raspberry PI.....	89
2. Apache.....	89
3. PHP.....	90
5. Tâche n°1 : Visualiser les données d'une serre.....	91
1. Les tables nécessaires dans la base de données.....	93
2. Description des tables nécessaires.....	94
3. Visualisation des données.....	97
4. Tâche n°2 : S'authentifier.....	100
5. Tâche n°3 : Programmer les paramètres d'une serre.....	103
1. Paramétrage des seuils.....	108
2. Les tables nécessaires dans la base de données pour paramétrier les seuils	109
3. Configuration des seuils.....	110
4. Paramétrage de l'arrosage.....	114
1. Les tables nécessaires pour configurer l'arrosage.....	115
2. La configuration de l'arrosage.....	116
3. Configuration d'un responsable.....	120
IV/- Conception préliminaire : Étudiant 4, ROCHER Marvyn.....	125
1. Préambule.....	125
2. Les outils de développement.....	126
3. Les installations diverses.....	127
1. Apache.....	127
2. MariaDB.....	127

3. PhpMyAdmin.....	128
4. Configuration générale.....	129
1. Création de la base de données.....	129
2. Création des utilisateurs.....	129
3. Affectation des droits.....	130
4. Accès à la base de données en distant.....	130
5. Gammu.....	131
6. Serveur mail.....	131
7. Tâche n°1 : Contrôler l'état d'une serre.....	134
8. Tâche n°2 : Envoyer une alerte en cas de dépassement de seuil.....	136
D- Conception détaillée.....	138
I/ Conception détaillée: Étudiant 1, MICHAUD Théo.....	138
1. Diagramme de classe.....	138
2. Description des classes.....	139
1. Classe : contrroleurDeSerre.....	139
2. Classe : Zone.....	140
3. Classe : Vanne.....	141
4. Classe : Timer.....	144
II/ Conception détaillée: Étudiant 2, AUVÉ Killian.....	146
1. Synoptique personnel.....	146
2. Diagramme de classe.....	147
3. Description des classes.....	148
1. Classe : ControleurDeSerre.....	148
2. Classe : Capteur.....	150
3. Classe : StockageDonnes.....	151
4. Classe : Timer.....	159
III/ Conception détaillée: Étudiant 3, VILLERMIN Maxime.....	161
1. Préambule.....	161
2. Le site web.....	161
1. Bootstrap.....	161
2 W3Schools.....	162

3 Organisations des fichiers du projet.....	162
2. Jquery.....	163
1. Qu'est-ce que jQuery?.....	163
2. Fonctionnement.....	163
3. Où utilisons-nous la bibliothèque ?.....	163
3. AJAX.....	164
1. Qu'est-ce que AJAX?.....	164
2. Fonctionnement.....	165
3. De quoi avons-nous besoin?.....	166
4. JSON.....	167
1. Qu'est-ce que JSON?.....	167
2. Structure d'un document JSON.....	167
3. Avantage.....	167
4. Utilisation de JSON.....	167
5. JSON dans notre projet.....	167
IV/ Conception détaillée: Étudiant 4, ROCHER Marvyn.....	169
1.Synoptique personnel.....	169
2. Les tables utilisées dans la base de données.....	170
1. Les tables utilisées dans la base de données.....	170
2. Détails des tables.....	171
1. MesuresSerreInterieur.....	171
2. MesuresSerreExterieur.....	171
3. LienSerreUtilisateur.....	171
4. Utilisateur.....	172
5. Incident.....	172
6. MesuresZone.....	172
7. Zone.....	173
8. Serre.....	173
3. Présentation des tables.....	174
1. Le besoin.....	174
2. Déroulement.....	174

3. Réalisation des tâches.....	175
4. Problèmes techniques.....	175
E- Réalisation.....	176
I/ Réalisation personnelle : étudiant n°1, MICHAUD Théo.....	176
1. ControleurDeSerre.....	176
2. Classe Timer.....	177
3. Classe Vanne.....	178
1. Ouvrir.....	179
2. Fermer.....	179
3. Contrôler.....	179
4. Classe Zone.....	181
5. Réalisation de l'application.....	182
6. Conclusion.....	184
II/ Réalisation personnelle : étudiant n°2, AUVÉ Killian.....	185
1. Classe Capteur.....	185
1. Récupération des données météorologiques.....	185
2. Constructeur de la classe.....	185
3. ReleverTemperature.....	186
4. ReleverHygrometrie.....	186
5. Inclusion de bibliothèque.....	186
6. ReleverLuminosite.....	187
2. Classe StockageDonnees.....	188
1. Envoi des données.....	188
2. ConnexionWifi.....	188
3. ConnexionServeur.....	189
4. InsererDonneesMesuresInterieurSerre.....	190
5. InsererDonneesMesuresZone.....	191
6. EnvoyerDonneesBDD.....	192
7. Création du Timer.....	192
8. Lancement du Timer.....	192
3. Stockage des données sur la carte SD.....	193

1. VerifierConnexionWifi.....	193
2. VerifierConnexionServeur.....	193
3. EnregisterDonneesFichierLocal.....	194
4. Récupération du mode de programmation.....	194
4. Création de la base de données.....	195
5. Installation Arduino.....	197
6. Conclusion.....	198
III/ Réalisation personnelle : étudiant n°3, VILLERMIN Maxime.....	199
1. La page « Accueil ».....	199
1. HTML.....	199
2. SQL.....	200
2 La page « Visualiser ».....	203
1.genereNomSerre() ;.....	203
2. La partie graphique.....	206
3 L'authentification.....	210
1. Configuration du fichier .htaccess :.....	210
2. Configuration du fichier .htpassword :.....	210
4 Partie « Programmer ».....	212
1.Gestion des boutons et des champs du site.....	212
5. Conclusion.....	216
IV/ Réalisation personnelle : étudiant n°4, ROCHER Marvyn.....	217
1. Réalisation du programme permettant d'envoyer des SMS.....	217
1.1 Connexion à la base de données.....	217
1.2 Lecture des données.....	218
1.3 Obtention des seuils.....	218
1.4 Concaténation des chaînes.....	219
1.5 Tests de comparaisons.....	222
1.5.1 Comparaison d'entiers.....	222
1.5.2 Comparaison de réels.....	222
1.6 Ajout de l'incident dans la base de données.....	224
1.7 Envoi de SMS.....	225

1.8 Test de connexion à la base de données.....	225
1.9 Échec de la connexion à la base de données.....	226
2. Complément d'alerte : l'alerte mail.....	227
3.CRONTAB.....	228
4. Conclusion.....	229
F - Planning prévisionnel.....	230
1. Revue 1.....	230
2. Étudiant n°1.....	230
3. Étudiant n°2.....	231
4. Étudiant n°3.....	232
5. Étudiant n°4.....	233
G- Les tests unitaires.....	234
I/ Test unitaire : étudiant n°1, MICHAUD Théo.....	234
II/ Test unitaire : étudiant n°2, AUVÉ Killian.....	236
III/ Test unitaire : étudiant n°3, VILLERMIN Maxime.....	237
IV/ Test unitaire : étudiant n°4, ROCHER Marvyn.....	239

**BTS SN****E 6-2 – PROJET TECHNIQUE****Dossier de présentation et de validation du projet**

Groupement académique : Nantes	Session 2019
Lycée : LPO Touchard-Washington	
Ville : Le Mans	
N° du projet : TW5	Nom du projet : Supervision de serres

Projet nouveau	Oui <input type="checkbox"/>	Non <input type="checkbox"/>	Projet interne	Oui <input type="checkbox"/>	Non <input type="checkbox"/>
Délai de réalisation	Session 2019		Statut des étudiants	Formation initiale <input type="checkbox"/>	Apprentissage <input type="checkbox"/>
Spécialité des étudiants	EC <input type="checkbox"/>	IR <input type="checkbox"/>	Mixte <input type="checkbox"/>	Nombre d'étudiants 4	
Professeurs responsables	Didier BERNARD, Philippe CRUCHET , François QUEREC, Philippe SIMIER				

A – Cahier des charges

1. Présentation et situation du projet dans son environnement

1. Contexte de réalisation

Constitution de l'équipe de projet :	Étudiant 1 EC <input type="checkbox"/> IR <input type="checkbox"/>	Étudiant 2 EC <input type="checkbox"/> IR <input type="checkbox"/>	Étudiant 3 EC <input type="checkbox"/> IR <input type="checkbox"/>	Étudiant 4 EC <input type="checkbox"/> IR <input type="checkbox"/>
Projet développé :	Au lycée ou en centre de formation <input type="checkbox"/>	En entreprise <input type="checkbox"/>	Mixte <input type="checkbox"/>	
Type de client ou donneur d'ordre (commanditaire) :	<p>Entreprise ou organisme commanditaire : Oui <input type="checkbox"/> Non <input type="checkbox"/></p> <p>Nom : Serre en direct</p> <p>Adresse : ZA de Saint Antoine 29270 Carhaix Plouguer</p> <p>Contact : M Claude COURTET</p> <p>Origine du projet :</p> <ul style="list-style-type: none"> ➤ Idée : Lycée <input type="checkbox"/> Entreprise <input type="checkbox"/> ➤ Cahier des charges :Lycée <input type="checkbox"/> Entreprise <input type="checkbox"/> ➤ Suivi du projet :Lycée <input type="checkbox"/> Entreprise <input type="checkbox"/> 			
Si le projet est développé en partenariat avec une entreprise :	<p>Adresse site : https://www.serre-en-direct.fr/</p> <p>Tél. : 02 98 93 61 31</p> <p>Courriel : serre.en.direct@orange.fr</p>			

2. Présentation du projet

La culture sous-serre désigne la pratique qui consiste à cultiver des végétaux, soit en culture maraîchère ou horticole à l'intérieur d'un abri spécifique transparent en verre, polycarbonate, ou bâche plastique. L'effet de serre, ainsi créé, permet de réunir des conditions hygrométriques, caloriques et photopériodiques idéales pour cultiver certains végétaux en dehors de leur région de prédilection ou de rallonger la période de culture.

La culture sous abris demande une rigueur importante au niveau de l'arrosage, de la surveillance de la température et de l'hygrométrie. L'arrosage des végétaux doit se faire de manière régulière et contrôlée, trop d'eau, ou pas assez, peut nuire à la croissance et au goût de la production. Une température supérieure à 35° peut entraîner la destruction des plants. De même, si elle passe en dessous de 0°, certains végétaux peuvent être détruits par le gel. Lorsque l'hygrométrie dépasse un certain seuil, par exemple 80 % pendant plus de trois heures, cela peut faire apparaître des maladies cryptogamiques.



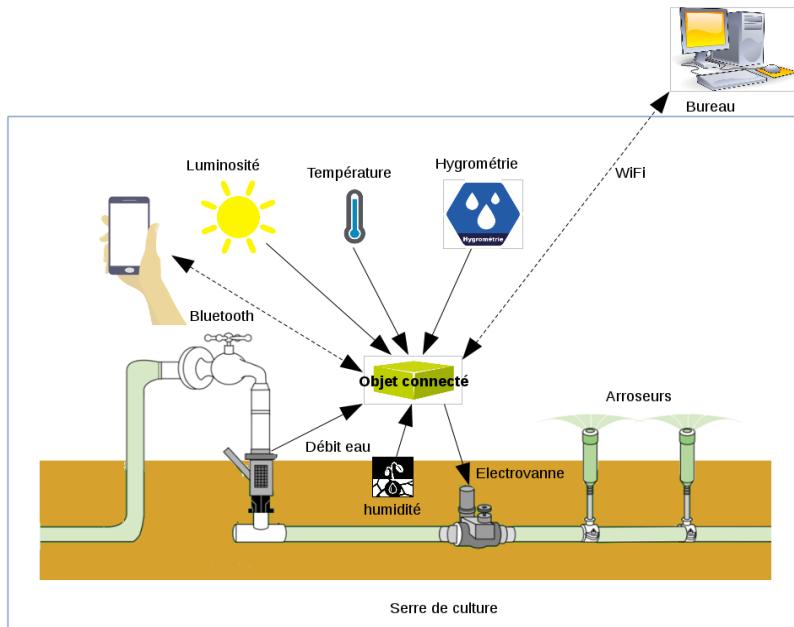
Cette constatation a conduit notre commanditaire, la société Serre en direct basée à Carhaix (29), à s'intéresser au domaine de la supervision de serres en utilisant des objets connectés. Il nous propose donc cette étude afin d'étendre son offre.

3. Situation du projet dans son contexte

Domaine d'activité du système support d'étude :

- télécommunications, téléphonie et réseaux téléphoniques ;
- informatique, réseaux et infrastructures ;
- multimédia, son et image, radio et télédiffusion ;
- mobilité et systèmes embarqués ;
- électronique et informatique médicale ;
- mesure, instrumentation et microsystèmes ;
- automatique et robotique.

4. Cahier des charges – Expression du besoin



Le principe de l'installation est donné par le synoptique suivant :

Un objet connecté dialogue avec le smartphone du maraîcher à proximité immédiate. Ce qui lui permet de piloter les électrovannes en mode de marche manuel et d'afficher les grandeurs relevées.

L'objet connecté transmet également ses relevés luminosité, température de l'air, hygrométrie, humidité du sol et quantité d'eau distribuée à un serveur situé dans le bureau de l'exploitation, pour faire un suivi régulier et assurer une surveillance. Lorsque des seuils d'alerte sont atteints en température, hygrométrie ou fuite d'eau, le maraîcher est informé par SMS.

De ce même bureau, il est également possible de programmer des cycles d'arrosage.

Pour cette étude, les électrovannes retenues sont du type 9V à impulsion pour réduire la consommation électrique. Elles sont regroupées par quatre permettant ainsi l'arrosage de quatre zones distinctes. Chaque vanne est équipée d'un capteur de débit à effet Hall pour détecter les fuites et mesurer la quantité d'eau distribuée. Le système d'irrigation est complété pour chacune des zones d'un capteur humidité pour le sol pour réaliser un arrosage en mode automatique par manque d'eau.



eZyvalve4 - Antelco



YF-B5 Capteur de débit d'eau



Capteur d'humidité du sol

Les fonctions majeures et les contraintes d'utilisation sont indiquées dans le tableau ci-après.

Fonctions de contrôle des vannes

Fonction	Description	Contrainte
Pilotage de vannes	Assure l'ouverture et la fermeture des vannes suivant le mode de fonctionnement.	Les électrovannes sont de type 6-12V à impulsion. Chaque électrovanne à un fonctionnement indépendant. Les heures d'ouverture et de fermeture des vannes sont mémorisées. Si une fuite est détectée sur un circuit d'arrosage, la vanne correspondante est fermée. L'incident est mémorisé.
Mesure du débit d'eau	Pour chaque sortie, le débit est mesuré.	Le débit est compris entre 1 et 30 litres par minutes. Un seuil doit être fixé pour déterminer la détection de fuite sur l'installation. La quantité d'eau distribuée est mémorisée.
Mesure de l'humidité du sol	Permet de déterminer la nécessité de l'arrosage en mode automatique.	Ce capteur peut ne pas être présent sur chaque vanne, dans ce cas l'arrosage en mode automatique n'est pas possible. Le seuil de manque d'eau est à déterminer.
Paramétrage des cycles d'arrosage	Les cycles d'arrosage sont utilisés dans le mode arrosage programmé.	Chaque zone est indépendante. Il est nécessaire de prévoir une date de début et de fin de période d'arrosage, la périodicité et sa durée. Le système doit être autonome une fois les ordres transmis.
Commande de l'arrosage	Assure l'interface de contrôle des vannes et la visualisation les mesures courantes, de température, d'hygrométrie, de luminosité, de l'état des vannes.	Permet de commander localement une des vannes du système d'irrigation localement. Les modes de marche possibles sont : manuel, automatique et programmé. En cas d'arrêt manuel, si le mode programmé est actif, l'arrosage reprend son cycle à la prochaine échéance. Elle se fera à partir d'une application Android.

Fonctions de mesure des conditions climatiques

Fonction	Description	Contrainte
Mesure de la température	Mesure périodique de la température intérieure de la serre.	<p>La périodicité de la mesure est à déterminer, une base de 15 minutes peut être un exemple. Seul un changement de température est mémorisé.</p> <p>Mesure entre -20° et +60°, la précision est fixée au 1/10 de degré.</p> <p>Un deuxième capteur est envisageable pour la mesure de la température extérieure de la serre.</p>
Mesure de l'hygrométrie	Mesure périodique de l'humidité relative à l'intérieur de la serre.	<p>La périodicité est identique à la mesure de température. Seul un changement du taux d'humidité est mémorisé.</p> <p>Mesure de l'humidité relative, la précision de la mesure est fixée à ±3 %.</p> <p>Un deuxième capteur est envisageable pour la mesure de l'hygrométrie extérieure de la serre.</p>
Mesure de la luminosité	Mesure périodique de la luminosité	<p>La périodicité est identique à la mesure de température. Seul un changement de luminosité est mémorisé.</p> <p>Gamme de mesures entre 0 et 100 000 lux</p>

Fonctions d'exploitation et de paramétrage

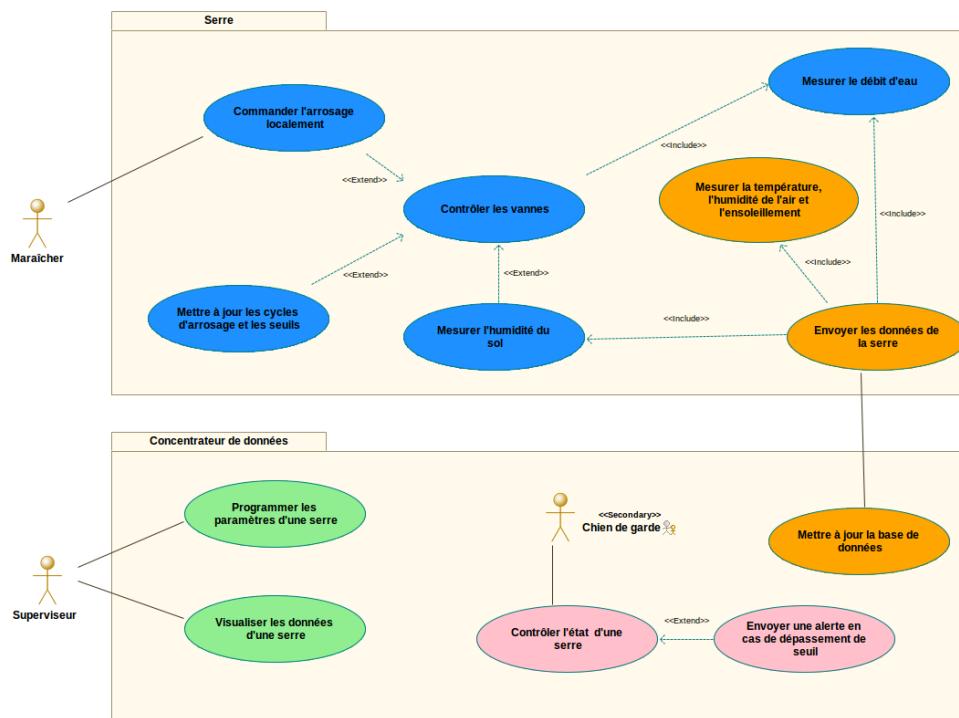
Fonction	Description	Contrainte
Stockage des données	Réceptionne les données en provenance de la serre et les enregistrements en base de données.	Seuls les changements de valeurs sont mémorisés. Une perte de connexion avec une serre constitue un incident, il doit-être mémorisé.
Programmation des paramètres de la serre	Permet de fixer les cycles d'arrosage de chaque zone indépendamment. Permet de fixer les seuils de déclenchement des alertes. Permet de renseigner les personnes à prévenir en cas d'alerte.	Pour chaque serre : Chaque zone est indépendante. Il est nécessaire de prévoir une date de début et de fin de période d'arrosage, la périodicité et la durée de l'arrosage. Pour chaque grandeur, il est possible de fixer un seuil mini et un seuil maxi de déclenchement d'alerte. Le nom, prénom, numéro de téléphone et mail sont renseignés pour chaque responsable de serre. Il peut y avoir plusieurs responsables par serre. Le paramétrage se fera à partir d'une page web. Un accès sécurisé sera mis en place pour l'accès au paramétrage.
Visualisation des données de la serre	Sous forme graphique, visualisation des périodes d'arrosage, des courbes température, ensoleillement, hygrométrie.	La période de visualisation est paramétrable. Les courbes sont superposables. L'affichage se fera sur une page web.
Surveillance de la serre	À intervalle régulier, les données de la serre sont comparées aux seuils prédéfinis en cas de dépassement, un SMS et/ou un courriel sont envoyés aux responsables de la serre, de même si un incident a été détecté.	L'intervalle d'interrogation est paramétrable, il est en relation avec la période de rafraîchissement des données et incident.

2. Spécifications

1. Diagrammes UML

Diagramme des cas d'utilisation

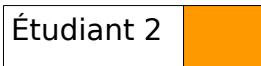
L'analyse du cahier des charges fait apparaître les cas d'utilisation proposés à la figure suivante avec la répartition par étudiant.



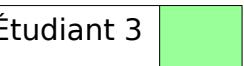
Étudiant 1



Étudiant 2



Étudiant 3

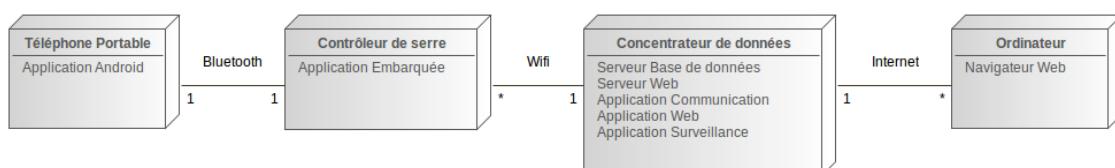


Étudiant 4



Diagramme de déploiement

Du point de vue de l'architecture matériel, le projet se décompose sous la forme de plusieurs sous-ensembles présentés par la figure ci-après.

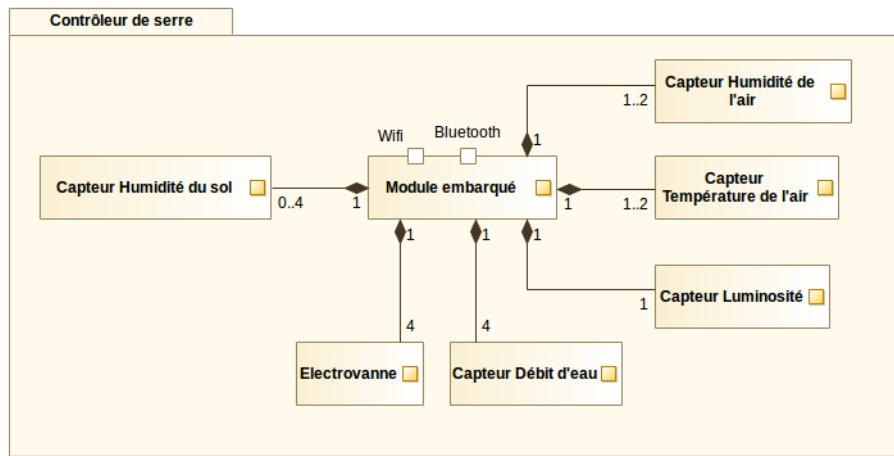


La liaison Wi-Fi entre les contrôleurs de serres et le concentrateur de données peut varier de 50 à 100 mètres. Il sera nécessaire de faire une étude spécifique pour la transmission des données.

Le téléphone portable doit être appairé pour contrôler une serre. La distance entre le contrôleur et le téléphone n'excède pas 10 mètres.

Diagramme de bloc d'un contrôleur de serre

Un contrôleur de serre se décompose en plusieurs blocs comme le montre la figure suivante. Le cœur du système est représenté sous la forme d'un module embarqué, l'ensemble constituera un objet connecté.



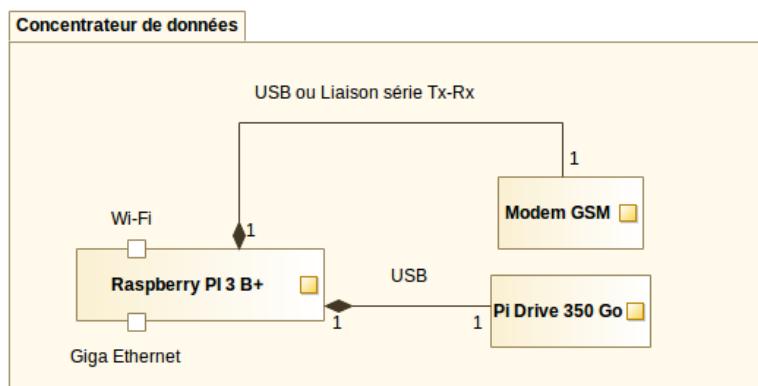
Le système possède au moins un capteur humidité et température de l'air. Le second est optionnel et permet de mesurer l'humidité et la température extérieure.

Les capteurs d'humidité du sol sont également optionnels. En cas d'absence, l'arrosage automatique en fonction du manque d'eau n'est pas possible.

Le module embarqué est basé sur une carte ESP32 à base de microcontrôleur, elle dispose d'une interface Wi-Fi et d'une interface Bluetooth nativement.

Diagramme de bloc d'un concentrateur de données

Le concentrateur de données est basé sur un Raspberry Pi équipé d'un disque dur type PI Drive de Western Digital et d'un modem GSM communiquant avec le Raspberry soit en USB, soit par une liaison série type Tx-Rx. Le Raspberry est relié à Internet via une Box d'un fournisseur d'accès en Wi-Fi ou avec un câble RJ45.



2. Contraintes de réalisation

Contraintes financières (budget alloué) :

L'ensemble du matériel nécessaire à la mise en œuvre de cette application est fourni par le demandeur le budget matériel est estimé à 300 €. Les logiciels utilisés pour le développement sont du domaine libre.

Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :

Pour tous les étudiants, Modelio est utilisé pour la modélisation UML, la gestion de version est réalisée avec Github et la gestion de projet est faite avec ProjectLibre et Trello.

Le développement du **contrôleur de serre** est réalisé en C et C++ avec l'environnement Arduino ou l'environnement Eclipse, de nombreuses bibliothèques sont à disposition.

Le développement du **concentrateur de données** sera réalisé avec NetBeans en HTML 5, CSS 3, PHP 5 et JavaScript pour la partie web, un framework tel que Bootstrap pourra être utilisé pour le « responsive design ». La partie base de données utilisera MariaDB et le langage SQL. Les applications de communication et de surveillance sont réalisées sous QT en C++ à l'aide de la chaîne de développement croisée.

L'application Android sur le **téléphone portable** est développée en ligne avec APP Inventor ou QML dans environnement QT selon le choix de l'étudiant et le temps disponible.

Contraintes qualité (conformité, délais ...) :

Les applications sont destinées à des personnes sans connaissance particulière en informatique, l'ergonomie doit être intuitive.

Les applications doivent-être configurable par fichier.

Le code doit répondre aux critères de qualité suivants en utilisant le standard Doxygen :

- Entête de fichier précisant auteur, date de création, de dernière modification, outils de production utilisés.
- Entête de fonctions précisant le rôle et l'utilisation des paramètres.
- Tous les commentaires nécessaires à une bonne compréhension du code.

La documentation doit être complète, homogène et non redondante. L'auteur de chaque page est identifiable, et, elle comporte :

Un dossier commun avec :

- Une rubrique **analyse** permettant de déterminer le périmètre du projet, le bilan complet des entrées/sorties, les besoins d'échange entre le téléphone portable, le contrôleur de serre et le concentrateur de données. Les prototypes des IHM, le recensement des données à mémoriser dans la base de données et la planification des différentes étapes du projet ainsi que le cahier de recette.
- Une rubrique **conception préliminaire** permettant de définir l'architecture logicielle, l'organisation des données du système, les protocoles d'échanges entre chaque module et la préparation de la phase d'intégration.

Un dossier individuel avec :

- Une **mise en situation** de la tâche dont l'étudiant est responsable.
- Une rubrique **conception** détaillée regroupant les algorithmes des modules complexes et les fiches de test unitaire permettant de valider les parties en charge.
- Un **dossier de réalisation** expliquant les technologies utilisées, les points clés du codage sans pour autant le reprendre et les résultats de test unitaire.

En commun également :

- Un **guide-utilisateur** pour réaliser l'installation, le déploiement et l'utilisation des applications.

Contraintes de fiabilité, sécurité :

Le système est destiné à un environnement agricole, il doit être robuste et protégé de l'humidité pour le contrôleur de serre. L'accès au concentrateur de données doit être sécurisé par mot de passe. Le système est destiné à fonctionner 24 heures sur 24. Il doit-être accessible en permanence. La reprise en cas de redémarrage doit être automatique.

3. Ressources mises à disposition des étudiants (logiciels / matériels / documents)

Chaque étudiant dispose d'un ordinateur pour réaliser le développement qu'il a en charge avec les logiciels nécessaires.

Le groupe dispose du matériel microcontrôleur, capteurs et actionneur. Une maquette est prévue pour le temps de développement.

Étudiant 1	Logiciel : <ul style="list-style-type: none"> • EDI Arduino et/ou Eclipse sous Linux • App Inventor en ligne ou QML dans l'environnement QT Matériel : <ul style="list-style-type: none"> • ESP32, capteurs d'humidité du sol, capteurs de débit d'eau à effet hall, ponts en H, Électrovannes. • Téléphone portable Android
Étudiant 2	Logiciel : <ul style="list-style-type: none"> • EDI Arduino et Eclipse sous Linux, • QT 5 sous Linux dans un environnement de développement croisé • Serveur de base données MariaDB. Matériel <ul style="list-style-type: none"> • ESP32, capteurs BME280, • Raspberry Pi.
Étudiant 3	Logiciel : <ul style="list-style-type: none"> • EDI NetBeans, avec HTML, CSS, PHP JavaScript, • Serveur Web Apache et PHP pour Raspbian, • Serveur de base données MariaDB. Matériel : <ul style="list-style-type: none"> • Raspberry Pi
Étudiant 4	Logiciel : <ul style="list-style-type: none"> • QT 5 sous Linux dans un environnement de développement croisé. • EDI NetBeans, avec HTML, CSS, PHP JavaScript Matériel : <ul style="list-style-type: none"> • Raspberry Pi • Modem GSM

3. Répartition des fonctions ou cas d'utilisation par étudiant

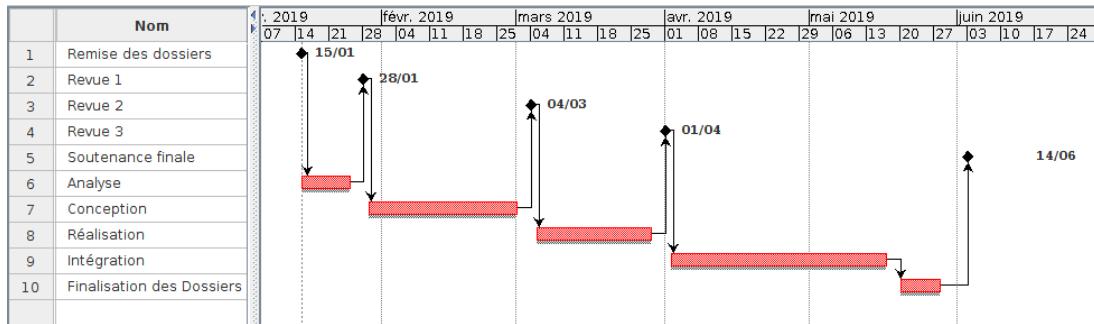
	Fonctions à développer et tâches à effectuer	
Étudiant 1 EC <input type="checkbox"/> IR <input checked="" type="checkbox"/>	<p><i>Liste des fonctions assurées par l'étudiant</i></p> <p>Sous système Contrôleur de serre</p> <ul style="list-style-type: none"> • Contrôler les vannes • Mesurer l'humidité du sol • Mettre à jour les cycles d'arrosage et les seuils. <p>Langage C, C++ dans l'environnement ESP32</p> <p>Sous système Téléphone Android</p> <ul style="list-style-type: none"> • Commander l'arrosage localement <p>APP Inventor ou QML</p>	<p>Installation: <i>Chaîne de développement EDI Arduino et Eclipse pour ESP32</i></p> <p>Mise en œuvre: <i>pilotage des électrovannes à impulsion, pont en H</i></p> <p>Réalisation: <i>Les 4 cas d'utilisation en charge.</i></p> <p>Documentation: <i>Responsable du Guide-utilisateur décrivant l'installation et l'utilisation du système dans la serre.</i></p>
Étudiant 2 EC <input type="checkbox"/> IR <input checked="" type="checkbox"/>	<p><i>Liste des fonctions assurées par l'étudiant</i></p> <p>Sous système Contrôleur de serre</p> <ul style="list-style-type: none"> • Mesurer la température, l'humidité de l'air et l'ensoleillement • Envoyer les données de la serre <p>Langage C, C++ dans l'environnement ESP32</p> <p>Sous système Concentrateur de données</p> <ul style="list-style-type: none"> • Mettre à jour la base de données <p>Langage C++ avec QT,</p>	<p>Installation: <i>Chaîne de développement EDI Arduino et Eclipse pour ESP32</i></p> <p>Mise en œuvre: <i>Transmission Wi-Fi avec antenne directionnelle</i></p> <p>Configuration: <i>Chaîne de développement croisé Serveur de base de données MariaDB</i></p> <p>Réalisation: <i>Les 3 cas d'utilisation en charge.</i></p> <p>Documentation: <i>Responsable du Dossier d'analyse.</i></p>
Étudiant 3 EC <input type="checkbox"/> IR <input checked="" type="checkbox"/>	<p><i>Liste des fonctions assurées par l'étudiant</i></p> <p>Sous système Concentrateur de données</p> <ul style="list-style-type: none"> • Visualiser les données de la serre • Programme les paramètres d'une serre <p>Langage HTML, CSS, PHP, JavaScript,</p>	<p>Installation: <i>Serveur web sur le Raspberry Pi,</i></p> <p>Mise en œuvre: <i>Transmission Wi-Fi avec antenne directionnelle</i></p> <p>Configuration: <i>Serveur web</i></p> <p>Réalisation: <i>Les deux cas d'utilisation en charge.</i></p> <p>Documentation: <i>Responsable du Dossier de conception</i></p>
Étudiant 4 EC <input type="checkbox"/> IR <input checked="" type="checkbox"/>	<p><i>Liste des fonctions assurées par l'étudiant</i></p> <p>Sous système Concentrateur de données</p> <ul style="list-style-type: none"> • Contrôler l'état d'une serre • Envoyer une alerte en cas de dépassement de seuil. <p>Langage C++ avec QT, script Batch Linux.</p>	<p>Installation: <i>Serveur de base de données Rasbian</i></p> <p>Mise en œuvre: <i>Modem GSM</i></p> <p>Configuration: <i>Serveur de base de données MariaDB Chaîne de développement croisé</i></p> <p>Réalisation: <i>Les deux cas d'utilisation en charge.</i></p> <p>Documentation: <i>Cohérence des documentations individuelles</i></p>

4. Exploitation Pédagogique – Compétences terminales évaluées

Informatique et Réseaux		Étudiant 1	Étudiant 2	Étudiant 3	Étudiant 4	
<hr/>						
C2.1	Maintenir les informations	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C2.2	Formaliser l'expression du besoin	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C2.3	Organiser et/ou respecter la planification d'un projet	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C2.4	Assumer le rôle total ou partiel de chef	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C2.5	Travailler en équipe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<hr/>						
C3.1	Analyser un cahier des charges	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.3	Définir l'architecture globale d'un prototype ou d'un système	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.5	Contribuer à la définition des éléments de recette au regard des contraintes du cahier des charges	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.6	Recenser les solutions existantes répondant au cahier des charges	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.8	Élaborer le dossier de définition de la solution techniquement	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.9	Valider une fonction du système à partir d'une maquette réelle	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C3.10	Réaliser la conception détaillée d'un module matériel et/ou logiciel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<hr/>						
C4.1	Câbler et/ou intégrer un matériel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.2	Adapter et/ou configurer un matériel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.3	Installer et configurer une chaîne de développement	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.4	Développer un module logiciel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.5	Tester et valider un module logiciel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.6	Intégrer un module logiciel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C4.7	Documenter une réalisation matérielle / logicielle	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

5. Planification (Gantt)

Début du projet	semaine 3 (15/01/2019)
Revue 1	semaine 5 (28/01/2019)
Revue 2	semaine 10 (04/03/2019)
Revue 3	semaine 14 (01/04/2019)
Remise des dossiers	semaine 22 (28/05/2019) à confirmer
Livraison	semaine 22 (27/05/2019)
Soutenance finale	semaine 24 (du 10/06/2019 au 14/06/2019) à confirmer



Vacances scolaires : Hiver du 09/02/2019 au 25/02/2019

Printemps du 06/04/2019 au 23/04/2019

6. Condition d'évaluation pour l'épreuve E6-2

Disponibilité des équipements

L'équipement sera-t-il disponible ? Oui Non

Atteintes des objectifs du point de vue du client

Le maraîcher peut piloter les électrovannes en mode manuel avec son smartphone. Les électrovannes fonctionnent de manière automatique à partir des informations obtenues des capteurs d'humidité du sol. Les cycles d'arrosage et les seuils d'alertes sont réceptionnés et stockés par le contrôleur de serre. Les électrovannes fonctionnent en mode programmé à partir d'un cycle d'arrosage. Si une fuite est détectée, un incident est déclenché. Les données environnementales sont affichées sur le smartphone ainsi que l'état des électrovannes.

Les données environnementales sont obtenues des capteurs et envoyées au concentrateur de données. Elles sont présentes ainsi que les relevés d'incidents dans la base de données. Chaque donnée est horodatée.

Le superviseur peut programmer des cycles d'arrosage. Il peut fixer des seuils d'alertes. Ces deux types d'information sont envoyés à une serre. Le superviseur ou un internaute autorisé peut visualiser les courbes environnementales et les périodes d'arrosage avec son navigateur web. Les périodes de visualisation sont paramétrables.

La base de données est scrutée périodiquement. La comparaison aux seuils fixés est réalisée pour chaque grandeur mesurée. En cas de dépassement de seuils ou de détection d'incident, un SMS est envoyé.

Avenants :

Date des avenants : Nombre de pages :

7. Observation de la commission de Validation

Ce document initial : comprend 13 pages et les documents annexes suivants :

(À remplir par la commission de validation qui valide le sujet de projet) a été utilisé par la Commission Académique de validation qui s'est réunie à LAVAL, le 30/11 /2018

Contenu du projet :	Défini <input type="checkbox"/> Insuffisamment défini <input type="checkbox"/>	Non défini <input type="checkbox"/>
Problème à résoudre :	Cohérent techniquement <input type="checkbox"/>	Pertinent / À un niveau BTS SN <input type="checkbox"/>
Complexité technique : <i>(liée au support ou au moyen utilisé)</i>	Suffisante <input type="checkbox"/> Insuffisante <input type="checkbox"/>	Exagérée <input type="checkbox"/>
Cohérence pédagogique : <i>(relative aux objectifs de l'épreuve)</i>	Le projet permet l'évaluation de toutes les compétences terminales <input type="checkbox"/> Chaque candidat peut être évalué sur chacune des compétences <input type="checkbox"/>	
Planification des tâches demandées aux étudiants, délais prévus ... :	Projet ... Défini et raisonnable <input type="checkbox"/> Insuffisamment défini <input type="checkbox"/> Non défini <input type="checkbox"/>	
Les revues de projet sont-elles prévues : <i>(dates, modalités, évaluation)</i>	Oui <input type="checkbox"/>	Non <input type="checkbox"/>
Conformité par rapport au référentiel et à la définition de l'épreuve :	Oui <input type="checkbox"/>	Non <input type="checkbox"/>

Observations :

1. Avis formulé par la commission de validation

- | | | |
|--|---|--|
| <input type="checkbox"/> Sujet accepté | <input type="checkbox"/> Sujet à revoir : | <input type="checkbox"/> Conformité au Référentiel de Certification / Complexité |
| en l'état | | <input type="checkbox"/> Définition et planification des tâches |
| | | <input type="checkbox"/> Critères d'évaluation |
| | | <input type="checkbox"/> Autres : |

- Sujet rejeté

Motif de la commission :

2. Nom des membres de la commission de validation académique

Nom	Établissement	Académie	Signature

3. Visa de l'autorité académique

(nom, qualité, Académie, signature)

Nota :

Ce document est contractuel pour la sous-épreuve E6-2 (Projet Technique) et sera joint au « Dossier Technique » de l'étudiant.

En cas de modification du cahier des charges, un avenant sera élaboré et joint au dossier du candidat pour présentation au jury, en même temps que le carnet de suivi.



B- Analyse

I/ Analyse du besoin

1. La serre



La serre est un abri exploitant le rayonnement solaire, destiné à la culture et à la protection des plantes. L'objectif étant de créer un environnement propice à leur développement en prenant en compte l'influence du climat. En créant un microclimat, la serre permet d'influencer le cycle végétatif des plantes. La température augmentera plus rapidement dans une serre et les gelées surviennent lorsque les températures extérieures sont largement négatives. Cela offre donc la possibilité de cultiver plus facilement des végétaux en culture maraîchère ou horticole. Les récoltes sont bien différentes sous serre. Le rendement et la précocité sont grandement améliorés.

Grâce à une serre de ce type, le maraîcher peut mieux gérer ses plantations, mais en revanche il doit se déplacer dedans avec son tuyau d'arrosage et va arroser comme il le souhaite.

La culture sous abris demande une rigueur importante (au niveau de l'arrosage, de la surveillance, de la température et de l'hygrométrie).

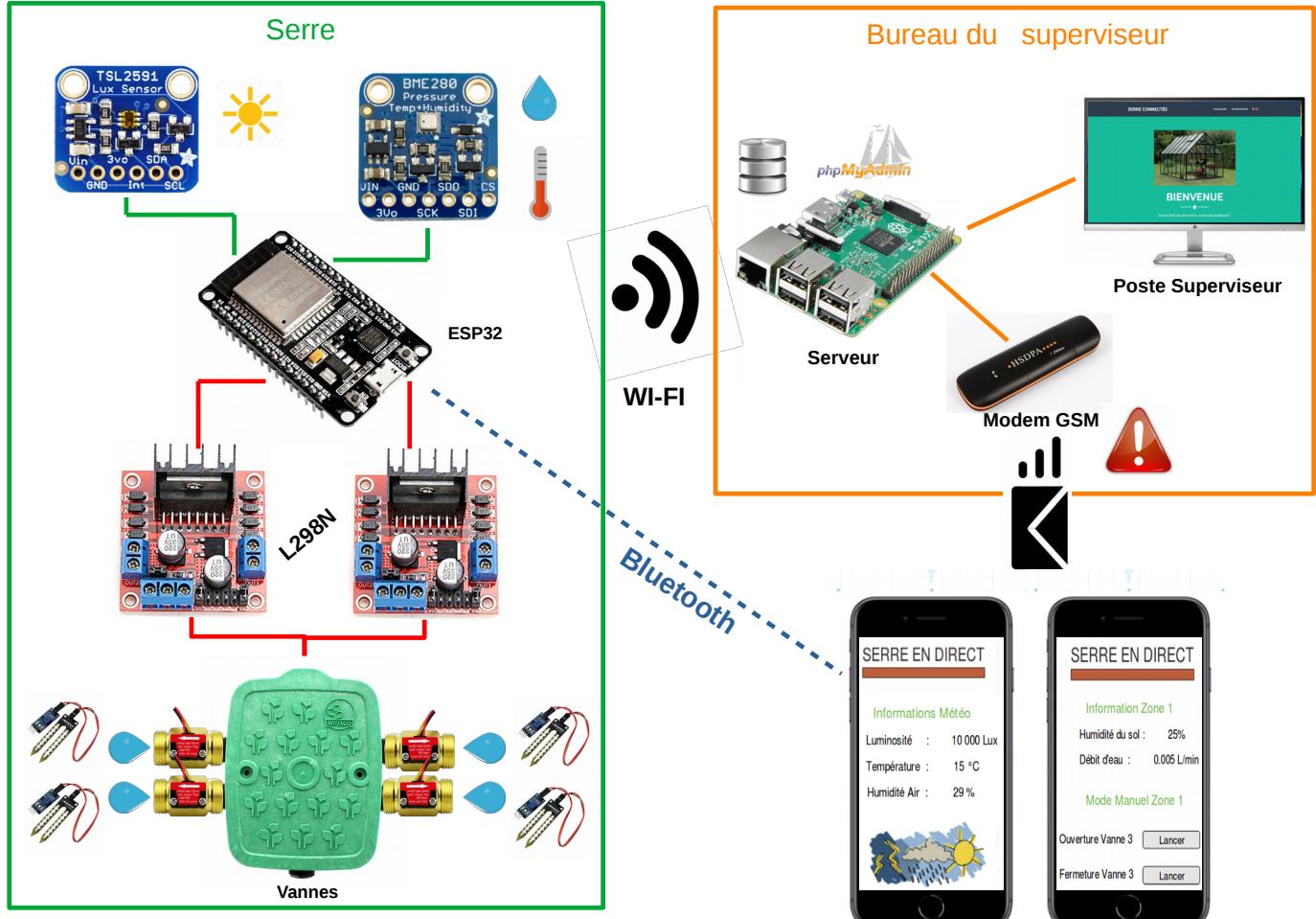
2. La demande

Cette constatation a conduit notre commanditaire, la société Serre en direct basée à Carhaix (29), à s'intéresser au domaine de la supervision de serres en utilisant des objets connectés. Il nous propose donc cette étude afin d'étendre son offre en offrant un service avec une irrigation connectée.

Le besoin exprimé par le commanditaire du projet est simple : rendre une serre connectée par le biais de divers capteurs, de technologies croisées. Afin de pouvoir connecter notre serre, nous avons à étudier, et à utiliser divers capteurs : capteur d'humidité d'air, d'éclairement lumineux, de température, d'hygrométrie, des capteurs de fuite d'eau...

Le besoin est donc de rendre connectée la serre, pour ce faire nous devons créer une application mobile afin de gérer l'ouverture et la fermeture des vannes par connexion Bluetooth. Nous devons également créer un site Internet afin de paramétriser la serre, mais également visualiser par le biais de courbes les tendances des données reçues et présentes sur la base de données.

3. Synoptique



Nous avons choisi de réaliser un nouveau synoptique afin de rendre compte des technologies dont nous avons besoin. Ainsi nous pouvons voir que ce projet est un projet de technologies croisées : nous y trouvons un Raspberry PI ; des ESP32, des Arduino, un téléphone ayant une application dédiée, un modem GSM. L'ensemble de ces technologies sera détaillé dans ce dossier.

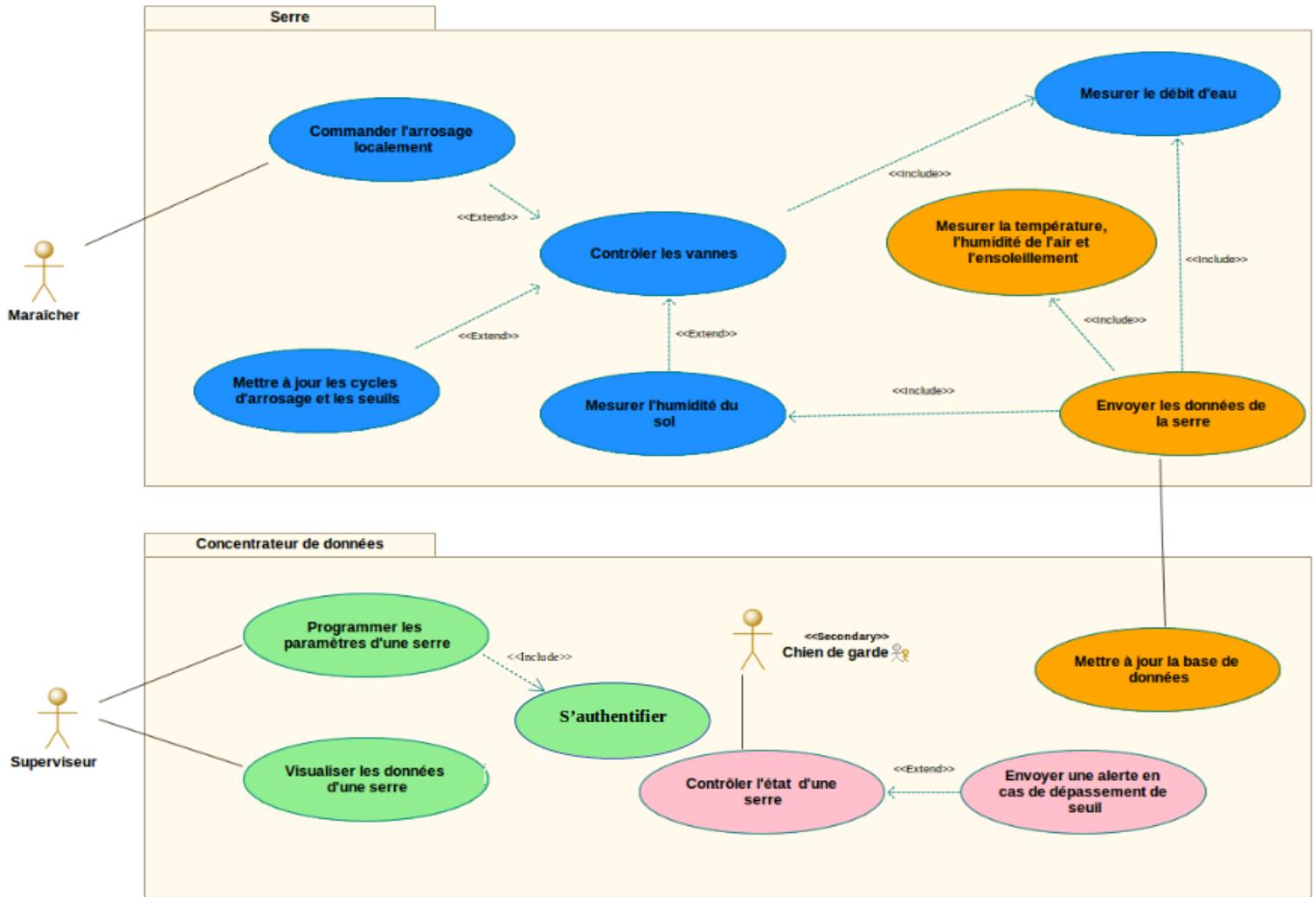


4. Spécifications fonctionnelles

4.1 Catalogue des acteurs

Acteur	Rôle
Maraîcher	Il contrôle les vannes : commande l'arrosage localement, mesure le débit d'eau, l'humidité du sol, il met à jour les cycles d'arrosage et les seuils.
Superviseur	Il programme les paramètres d'une serre. Il visualise les données d'une serre.
Chien de garde	Il contrôle l'état d'une serre. Il envoie une alerte en cas de dépassement de seuil.

4.2 Diagramme des cas d'utilisation



En regardant le diagramme de cas d'utilisation, nous pouvons d'ores et déjà annoncer la répartition des étudiants sur les tâches du projet.

- **Michaud Théo** s'est chargé de contrôler les vannes (commander l'arrosage localement, mettre à jour les cycles d'arrosage, mesurer l'humidité du sol, mesurer le débit d'eau)
- **Auvé Killian** lui s'est chargé d'envoyer les données à la serre (mesurer la température, l'humidité de l'air, met à jour la base de données)
- **Villermin Maxime** a programmé les paramètres d'une serre. Il a permis la visualisation les données d'une serre par l'élaboration du site Internet entièrement conçu pour le suivi de la serre.
- **Rocher Marvyn** a contrôlé l'état d'une serre en envoyant une alerte en cas de dépassement de seuil.

II/ Les contraintes

Pour réaliser notre projet, nous devons penser aux contraintes de développement, mais aussi aux contraintes matérielles qui rendent compte de la manière dont nous allons réaliser ce projet.

Le budget matériel ne doit pas excéder 300€, nous verrons plus tard le coût total de nos choix, pour ce projet. Ce projet doit pouvoir être utilisé par des personnes sans connaissance particulière en informatique (intuitif). Nous devons aussi prendre en compte l'environnement où sera placé le matériel, le système doit être robuste et protégé de l'humidité. Le système de gestion de données doit être protégé par un mot de passe pour qu'il soit sécurisé.

Le matériel que nous allons utiliser se décompose de la manière suivante :

- Raspberry PI : afin de créer un serveur de base de données, un serveur web.
- Un modem GSM afin de paramétrier les alertes en cas de dépassement de seuils.
- Un téléphone portable Android afin de tester l'application créée.

Les logiciels que nous pouvons utiliser sont open source, et sont :

- Modelio (création de divers diagrammes, modélisation UML),
- ProjectLibre pour la gestion du projet,
- Netbeans (développement du site Internet),
- QT5 (QML pour l'application mobile) , ou AppInventor pour la création de l'application en fonction du temps disponible.

III/ Nos choix concernant le développement du projet

1. Choix liés au développement du site

1. *Langages utilisés*

Pour le développement du site web, nous avons utilisé le langage PHP ('*PHP Hypertext Preprocessor*'). C'est un langage de scripts libre, qui permet de rendre le site dynamique par un serveur.

Il est spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

Ensuite, nous utilisons le langage HTML ('*HyperText Markup Language*'), langage de balise qui permet de structurer le contenu d'une page WEB. On utilise aussi le langage HTML pour faire des formulaires. Il est souvent utilisé avec des feuilles de style ou des scripts JavaScript.

Nous utiliserons donc ces deux autres langages: les feuilles de style, ou plus communément le CSS ('*Cascading Style Sheets*'). Ce sont des pages qui permettent de mettre en forme le visuel d'une page, de décrire la présentation des pages WEB. Le JavaScript, lui, est un langage de script qui agit sur l'ordinateur de l'internaute. Il est utilisé par exemple pour vérifier ou créer des formulaires, mais également pour animer des éléments du site. Chacun des langages que nous avons choisis a été vu lors de notre cursus scolaire.

2. *Choix de l'EDI*

Afin de développer l'interface du site, nous devons utiliser Netbeans d'après le cahier des charges. Devenu en 2000 open source, il permet de développer des applications dans de nombreux langages (*Java, XML, PHP, HTML, C, C++, etc.*). L'avantage de l'utiliser pour le développement de notre projet est que nous l'avions déjà utilisé auparavant. En plus de proposer une large liste de langages de programmation, c'est un EDI simple d'utilisation, rapide et intuitif.

2. Choix liés à l'application mobile

1. Langages utilisés

QML est un langage scripté de haut niveau. Ses commandes, ou plus précisément, ses *éléments*, associés à la puissance et à l'efficacité des bibliothèques Qt, rendent la programmation plus intuitive. Dessiner un rectangle, afficher une image à une position donnée, etc. : derrière ces éléments se trouvent des bibliothèques C++ complexes qui effectuent ces actions efficacement.

2. Choix de l'EDI

Afin de développer l'interface de l'application, notre choix s'est fait entre deux EDI différents.

App Inventor, un logiciel en ligne qui permet de créer des applications pour appareils Android à travers une interface purement visuelle (WYSIWYG pour « What You See Is What You Get ») et de configurer les actions de l'application par un système de blocs logiques.

QtCreator, un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++.

Pour nous, le choix s'est tourné dans un premier temps sur App Inventor. L'avantage de ce choix est de pouvoir créer l'interface fonctionnelle rapidement malgré les limitations de ce choix puisqu'on ne peut pas créer de procédure et cela nous contraint à beaucoup de répétitions dans les blocs à disposer pour le code.

Dans un second temps, on va choisir Qt Quick, ce qui va nous permettre de créer une application ergonomique et d'avoir une bonne qualité de code.

3. Choix du système d'exploitation

L'application sera installée sur une tablette ou un Smartphone sous Android. Cela permet de téléverser facilement entre notre ordinateur et notre smartphone.

3. Choix liés au contrôleur de serre

1. *Objet connecté ESP32*

L'ESP32 sera notre objet connecté dans la serre. Il peut faire la liaison Wi-Fi et Bluetooth. Il possède un grand nombre de pins(36 pins) qui sera utile pour brancher tous les capteurs, contrôleur de vannes...

Voici un tableau des caractéristiques de l'ESP32 que nous avons retenu.

Nombre de cœurs	2
Architecture	32 bits
Fréquence CPU	160 MHz
Wi-Fi	Oui
Bluetooth	Oui
RAM	512 KB
FLASH	16 MB
GPIO PINS	36
Bus	SPI, I2C



ESP32

4. Choix liés au contrôleur de vannes

1. Le boîtier

Nous devons choisir un boîtier qui nous permet de gérer plusieurs vannes indépendamment et grâce à une seule carte de commande et en faisant attention au budget. Les électrovannes sont de type 9V à impulsion pour réduire la consommation.

Il doit être également étanche et qu'il puisse tenir avec le temps.

2. Langages utilisés

Une carte de commande va être sélectionnée pour commander les vannes. Elle va nous permettre d'ouvrir ou de fermer les vannes selon le choix du maraîcher.

Nous allons utiliser le C++, c'est un langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique). Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique.

3. Choix de l'EDI

Arduino est une plateforme de prototypage électronique, qui est open source avec un langage proche du C, avec une section « setup », une section « loop » et la possibilité de créer des classes.

Arduino c'est aussi de très nombreuses bibliothèques qui permettent d'utiliser des fonctions élaborées sur la carte ESP32, mais aussi de pouvoir interfaçer et utiliser toutes sortes de matériels (afficheur LCD, clavier matriciel, etc.)

5. Analyse des besoins capteurs

Les capteurs d'humidités vont nous permettre de mesurer l'humidité du sol et de la retransmettre au maraîcher. Nous aurons besoin d'autant de capteurs que de vannes puisqu'à chaque sortie de vanne il y aura un capteur d'humidité. Nous programmerons ce capteur en C++ sous Arduino.

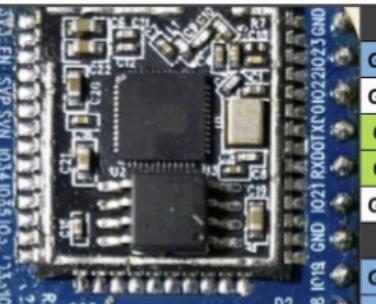
Les capteurs de débit d'eau vont nous permettre de mesurer le débit d'eau à chaque sortie de vanne et de l'afficher sur l'application. Il faudra choisir un capteur qui est capable de se mettre à la sortie de la vanne. Nous programmerons ce capteur en C++ sous Arduino.

Nous avons aussi besoin de capteurs capables de pouvoir relever la température,d'hygrométrie, la luminosité de la serre afin de les transmettre au serveur pour pouvoir la superviser.

ESP32 Dev Board PINMAP

			3.3V					
(pu)			RESET	EN				
SVP		ADC0		GPIO36				
SVN		ADC3		GPIO39				
		ADC6		GPIO34				
		ADC7		GPIO35				
	TOUCH9	ADC4		GPIO32				
	TOUCH8	ADC5		GPIO33				
DAC1		ADC18	Impulsion vanne4	GPIO25				
DAC2		ADC19	Impulsion vanne3	GPIO26				
	TOUCH7	ADC17	Impulsion vanne2	GPIO27				
	TMS	TOUCH6	ADC16	HSPI SCK	GPIO14			
(pd)	TDI	TOUCH5	ADC15	HSPI MISO	GPIO12			
					GND			
	TCK	TOUCH4	ADC14	HSPI MOSI	GPIO13			
					FLASH D2	GPIO9		
					FLASH D3	GPIO10		
					FLASH CMD	GPIO11		
					5V			

Humidité



Nous pouvons donc voir que sur une carte ESP32 et pour une serre à gérer, nous pouvons brancher tous nos capteurs et nos vannes.

Si l'on branche nos 4 vannes, nos 4 capteurs d'humidité, nos BME280 et TSL2591 ainsi que notre carte SD, il nous reste les 4 broches pour les 4 capteurs de débit d'eau.

6. Choix liés à la gestion des alertes

Dans le cadre du projet, une partie est dédiée à l'envoi d'alerte et notamment des alertes sous forme de SMS.

Pour ce faire, à disposition nous avions un modem GSM ; un HSUPA Modem. Lequel était fourni par le commanditaire du projet.

Ce modem est un modem GSM USB 2.0 ; il est compatible avec les ordinateurs ainsi avec

Raspbian. Ce modem est conçu afin d'y insérer une carte SIM pour pouvoir envoyer des SMS.

Afin de gérer les alertes, le modem GSM présenté est celui qui fut utilisé. En ce qui concerne le principe de la gestion des alertes, celle-ci est simple : dans le cahier des charges était exprimé le souhait de pouvoir alerter le responsable en cas d'incident ; c'est-à-dire lorsqu'une valeur est en dehors des seuils minimums/maximums qui ont été définis, ou bien que le débit d'eau est trop élevé.

Afin de prévenir le responsable, il sera averti par SMS, ce SMS contiendra la valeur en dehors des seuils définis, le rappel de la valeur autorisée, ainsi que la nature de l'incident, et la serre ou la/les zones où l'incident s'est produit.

Ce rôle d'alerte sera en lien avec le Raspberry PI, afin de pouvoir envoyer des alertes la base de données va être interrogée toutes les 15 minutes afin de contrôler la serre dans le but d'envoyer ou non des alertes, ce modem GSM sera également branché directement en USB sur le Raspberry PI.

Dans le but d'établir le programme permettant l'envoi d'alerte, le langage utilisé sera le script Bash Linux. Ce programme sera directement dans les dossiers racines du Raspberry PI afin de pouvoir le lancer de manière automatique. L'environnement propre aux alertes est donc le Raspberry tant en interrogeant la base de données afin de pouvoir comparer, tant en envoyant des SMS de manière complètement automatique au numéro de téléphone du responsable (présent dans la base de données).

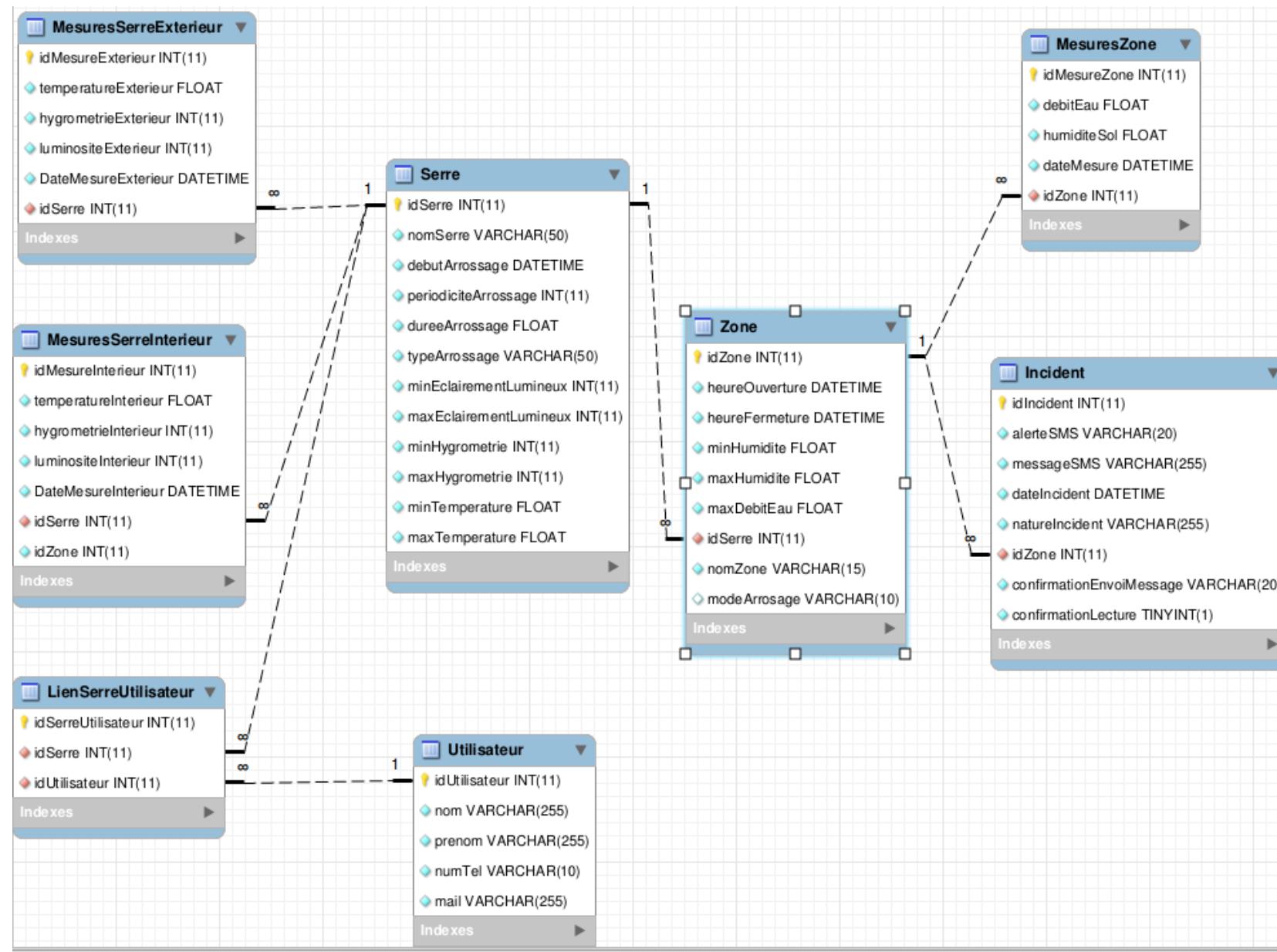
7. Coût total

Matériel	Nombre	Prix
Raspberry Pi 3	1	30€
Modem GSM HSUPA	1	10€
BME280	1	8€
L298N	2	10€
TSL2591	1	9€
Antelco eZyvalve4	1	110€
YL-69	4	32€
ESP32	1	7€
Module RTC	1	4€
Shield SD	1	7€
Carte Sim	1	10€
Forfait carte Sim	Tous les Mois	2€
YF-B5	4	28€

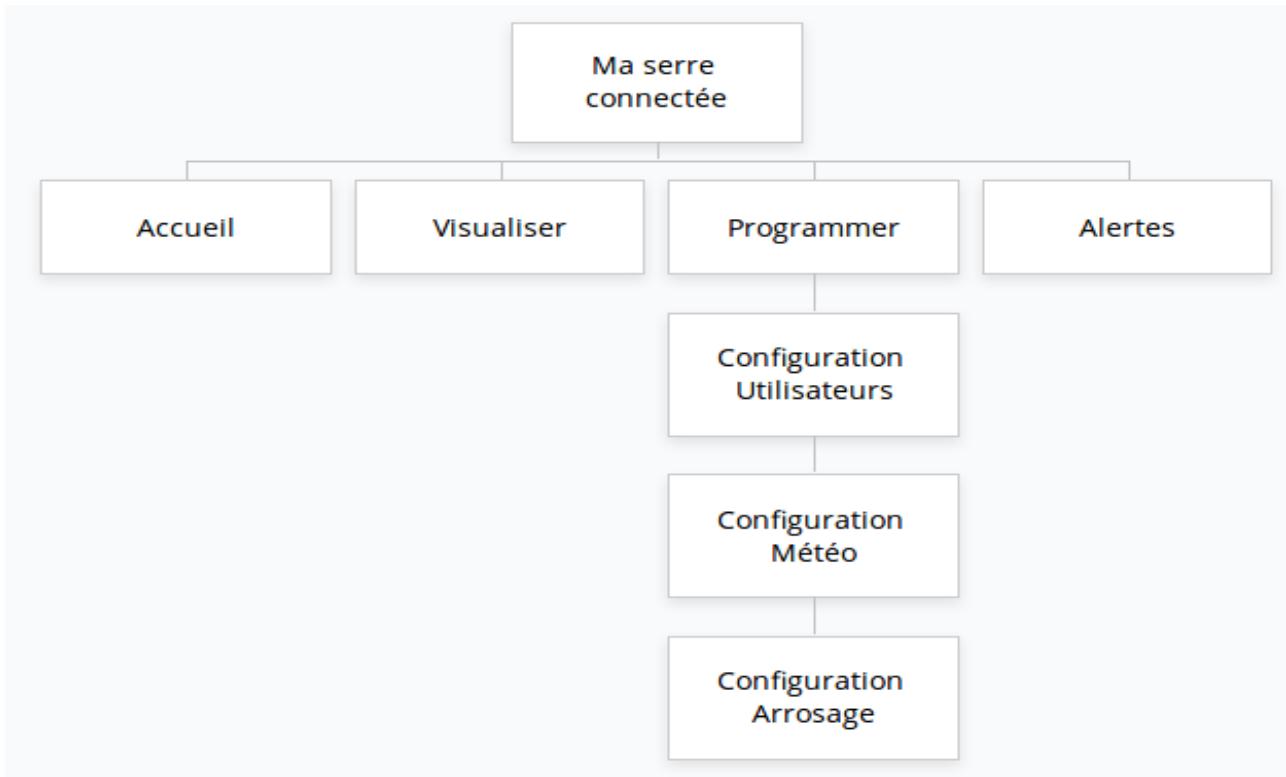
Le coût total est donc de 265€ et rajouter 2€/mois.

On constate donc que notre projet ne dépasse pas le coût total prévu par le cahier des charges.

IV/ Base de données



V/ Plan du site



En suivant le plan du site « *Ma serre connectée* », nous voyons que le choix du visuel du site web se décompose en 4 onglets. Le premier : « Accueil » est comme son nom l’indique la page d’accueil du site web. L’onglet « Visualiser » est dédié à la visualisation des différentes données (grandeur : température, humidité, etc.), contenus dans la base de données et mesurer par les différents capteurs.

L’onglet « Programmer » contient trois autres pages :

- « Configuration Utilisateurs » : qui permet d’ajouter, lister, modifier ou supprimer un utilisateur
- « Configuration météo » : qui permet de sélectionner la serre où l’on veut fixer les seuils d’alertes, une partie configuration de la serre, et une partie dédiée aux différentes zones qui compose la serre sélectionnée

- « Configuration Arrosage » : permet aussi de sélectionner la serre, de configurer les différents paramètres d’arrosage des différentes zones de la serre, ainsi que le mode d’arrosage. Une partie « choix du responsable » est disponible et permet si on le souhaite de modifier les informations d’un des responsables disponibles dans la liste.

En dernier, nous avons un module d’alerte qui permet de lister les alertes envoyées en cas de dépassement de seuil prédéfini sur le site web.

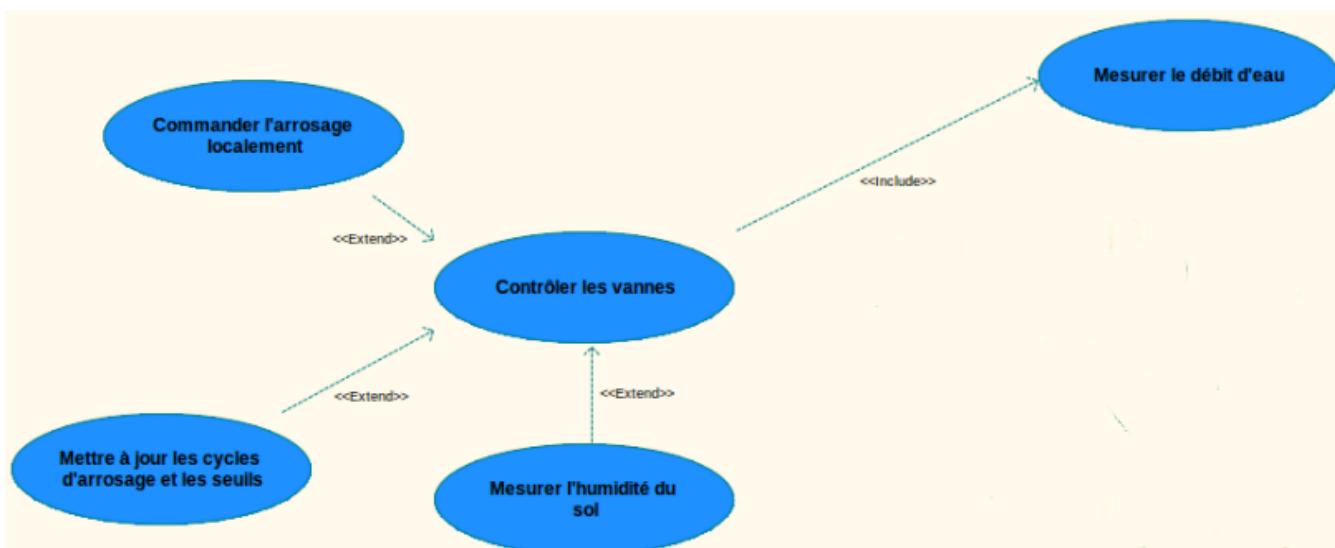
C - Conception préliminaire

I/ Conception préliminaire : Étudiant 1, MICHAUD Théo

1. Préambule

Conformément au cahier des charges, mais aussi au diagramme de classe, l'étudiant n°1 a été affecté aux tâches :

- ✗Contrôler les vannes
- ✗Commander l'arrosage localement
- ✗Mesurer l'humidité du sol
- ✗Mettre à jour les cycles d'arrosage et les seuils
- ✗Mesurer le débit d'eau



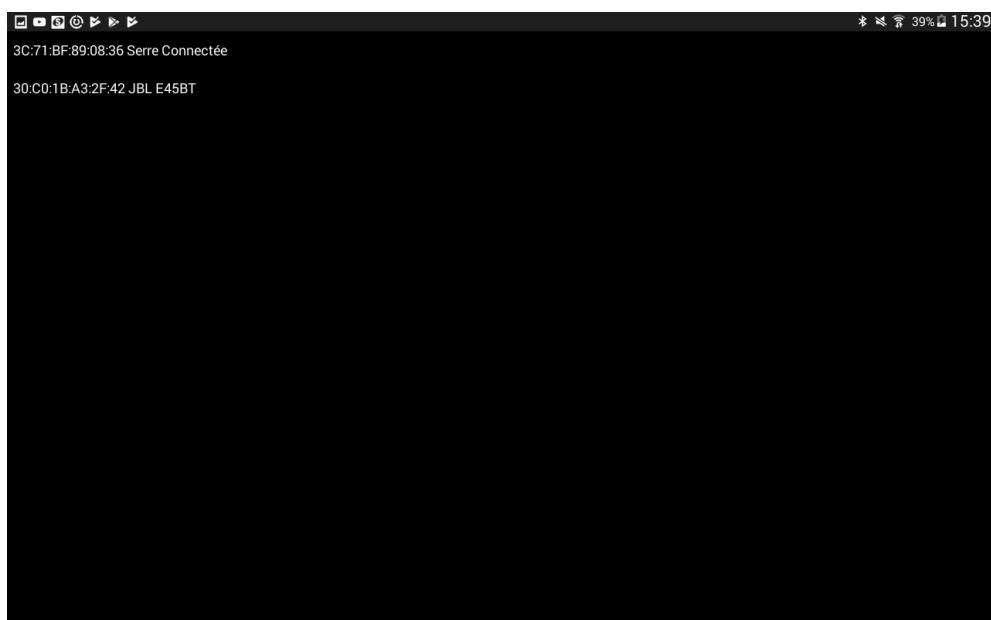
2. Tâche n°1 « Commander l'arrosage localement »

Pour comprendre comment marche ce projet, on peut regarder l'application qui va nous permettre de visualiser le projet. Ceci est un prototype créé avec AppInventor, l'objectif étant d'avoir une interface utilisable, mais faite rapidement pour une question de temps.

Une première interface nous permet de sélectionner à l'aide du Bluetooth la serre que l'on veut gérer.



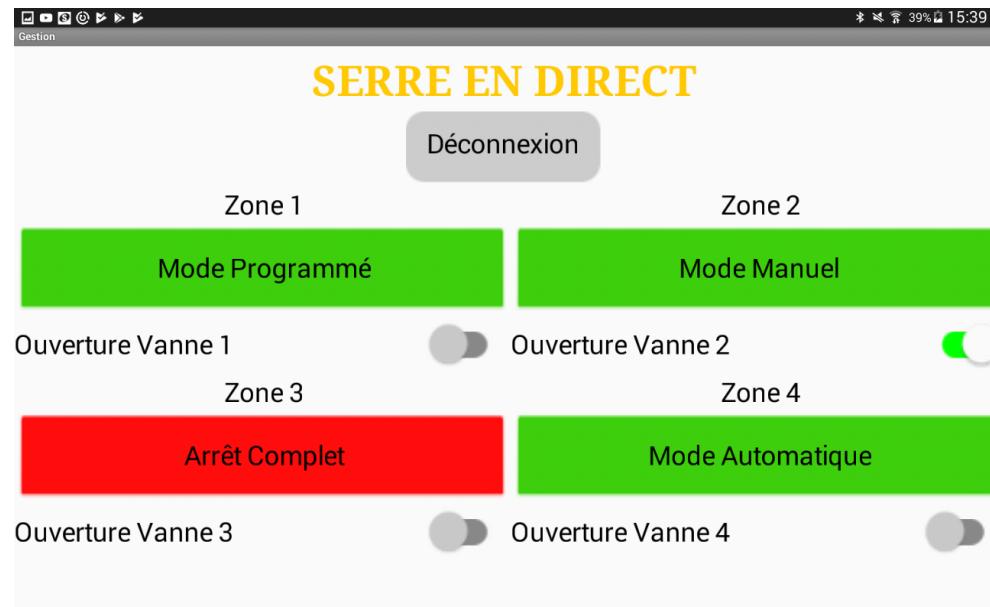
Par exemple, ici, on va sélectionner « Serre connectée »



Lorsque l'on arrive à se connecter sur la serre, toutes les vannes sont sur le mode « Arrêt complet » et le switch « Ouverture de vanne » est inutilisable.



Ensuite, on peut choisir le mode que l'on veut enclencher, par exemple, « Mode programmé » pour la zone 1 et « Mode Manuel » pour la zone 2, en choisissant ce dernier mode, le switch Ouverture Vanne 2 devient utilisable et l'on peut choisir d'ouvrir ou de fermer la vanne 2 en fonction du choix fait par le maraîcher.

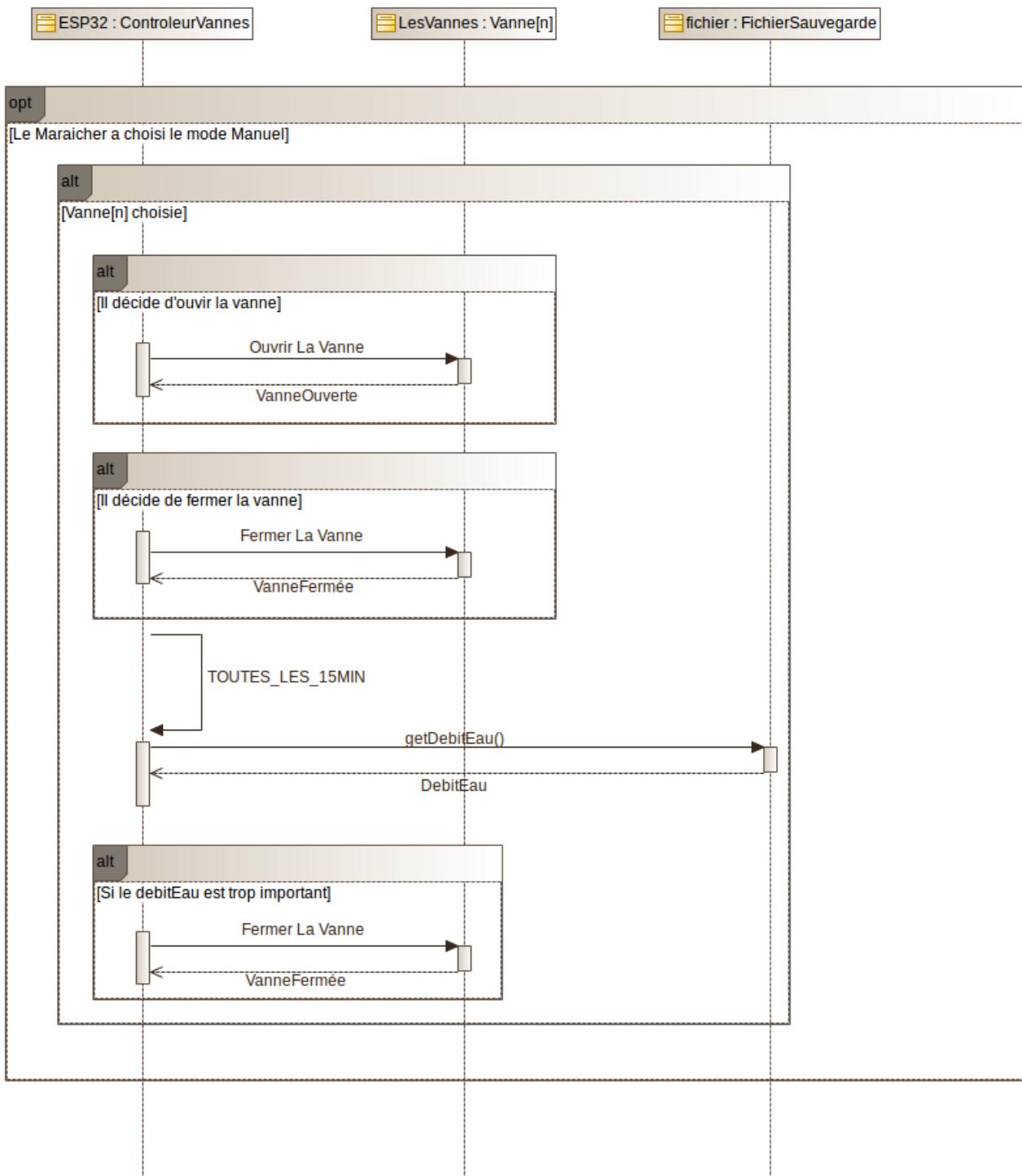


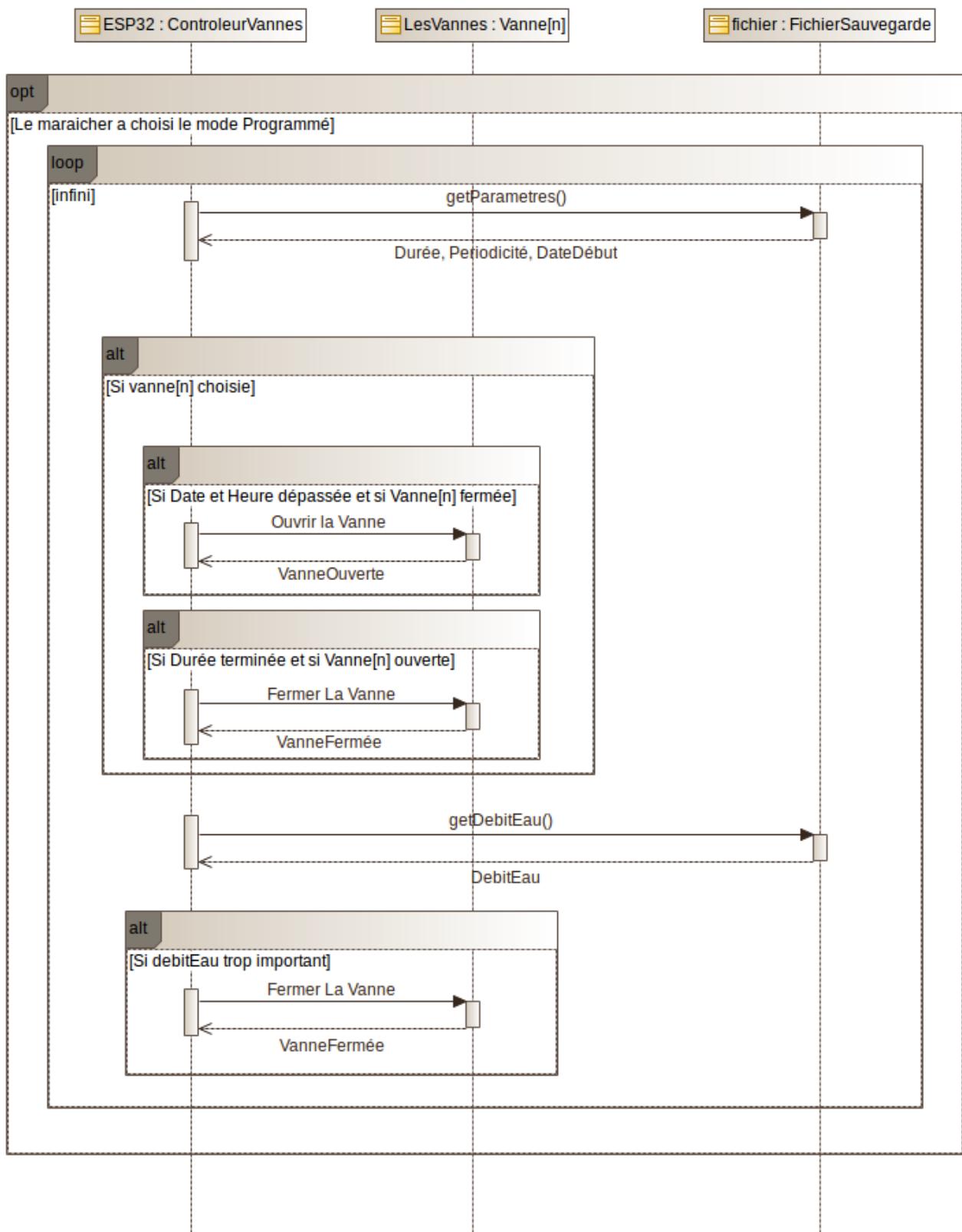
3. Tâche n°2 : « Contrôler les vannes »

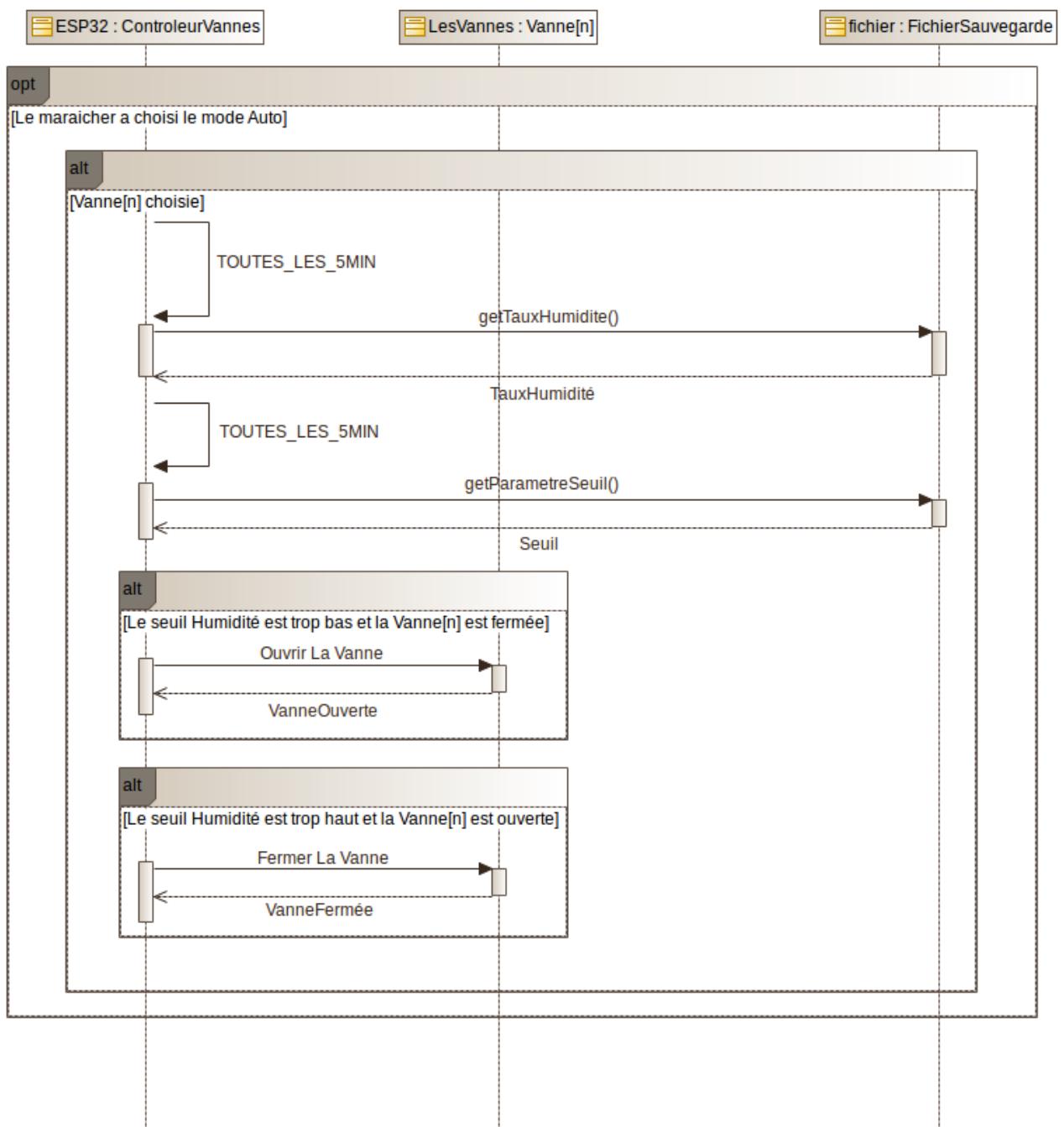
Afin de rendre compte de la conception de ce cas d'utilisation sera détaillée la description du cas, mais également le diagramme de séquence associé.

Description du cas d'utilisation

Nom du cas d'utilisation	Contrôler les vannes
Utilisateur concerné	Maraîcher
Étudiant en charge	Étudiant numéro 1, Théo MICHAUD
Préconditions	<p>Une ESP32 avec un programme qui permet de gérer les vannes.</p> <p>Un fichier d'arrosage dans lequel sont fixées la date et l'heure de départ, la périodicité, la durée est présent pour réaliser le mode programmé, mais aussi les seuils d'humidité pour le mode automatique.</p> <p>Un capteur d'humidité du sol est présent pour le mode automatique</p>
Scénario nominal	<p>On va pouvoir contrôler plusieurs zones à arroser. Après avoir choisi la zone à arroser nous allons avoir 3 modes de fonctionnement qui vont nous permettre de piloter les vannes : un mode manuel, un mode programmé et un mode automatique.</p> <p>Si le mode manuel est sélectionné, l'utilisateur va pouvoir ouvrir ou fermer sa vanne comme il le souhaite.</p> <p>Si le mode programmé est choisi, en fonction des cycles qui ont été programmés auparavant et qui sont enregistrés dans un fichier sur l'ESP32, l'ouverture ou la fermeture des vannes vont se faire aux heures choisies et pendant une durée définie ainsi qu'une période définie.</p> <p>Si le mode automatique est sélectionné, la vanne choisie va s'ouvrir ou se fermer automatiquement en fonction des seuils humidité du sol enregistrés dans un fichier sur l'ESP32, si le sol est trop humide on ferme la vanne, s'il est trop sec on ouvre la vanne.</p> <p>Enfin, on peut choisir : de sélectionner Arrêt complet qui fermera la vanne directement.</p>
Post-condition(s)	Les vannes sont pilotées en fonction du mode dans lequel elles sont positionnées

Diagrammes de séquenceMode Manuel

Mode programmé



Mode automatique

4. Choix technologiques

1. Choix du boîtier - eZyvalve4 – Antelco



C'est un boîtier de vannes compact avec 4 électrovannes internes qui se connectent à un contrôleur d'irrigation à verrouillage. L'installation est simple et eZyvalve®4 offre un moyen sans problèmes de distribuer de l'eau vers un système de micro-irrigation.

Ce boîtier fonctionne avec des impulsions de 250ms pour ouvrir 1 vanne.

Caractéristiques

Pré assemblé sans raccord de tuyau interne.

Matériaux stabilisés aux UV à haute résistance.

Comprends un joint de câble et des connecteurs de fils.

Résistant à l'eau, à la poussière et aux parasites.

Couvercle antidérapant texturé avec joint torique et vis en acier inoxydable.

Raccordements d'entrée et de sortie filetés BSP 3/4 "BSP.

Dimensions

LONGUEUR: 105 mm

PROFONDEUR: 182 mm

LARGEUR: 182 mm

On va contrôler ce boîtier et les 4 vannes qui sont à l'intérieur grâce à l'ESP32 (présentée plus bas).

Pour gérer les vannes, il faudra se connecter en Bluetooth sur l'ESP32 .

On aura besoin de brancher 2 fils pour le sens et 1 fil pour l'impulsion pour chaque vanne.

2.La carte de développement : l'ESP32

Le choix de l'ESP32 est un plus pour notre serre puisqu'il y a beaucoup de broches sur celle-ci, mais il y a surtout un module Bluetooth intégré.



Pour pouvoir gérer les vannes, on se connecte sur notre ESP32, en Bluetooth avec notre application, ce qui va permettre à notre maraîcher de choisir de gérer une zone en particulier facilement à proximité de celle-ci.

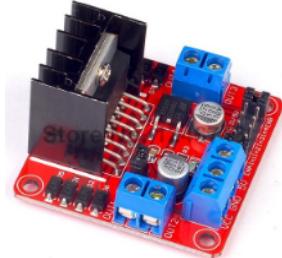
Deux types de Bluetooth dans le module ESP32:

le Bluetooth classique et le Bluetooth BLE Bluetooth Low Energy

Le Bluetooth Low Energy, comme son nom l'indique, consomme moins d'énergie que le Bluetooth classique. Pour ce faire, vous pouvez envoyer des données au besoin avec des mises à jour périodiques prédéfinies. Mais contrairement au Bluetooth classique, il n'est pas utilisé pour transférer des fichiers ou de la musique.

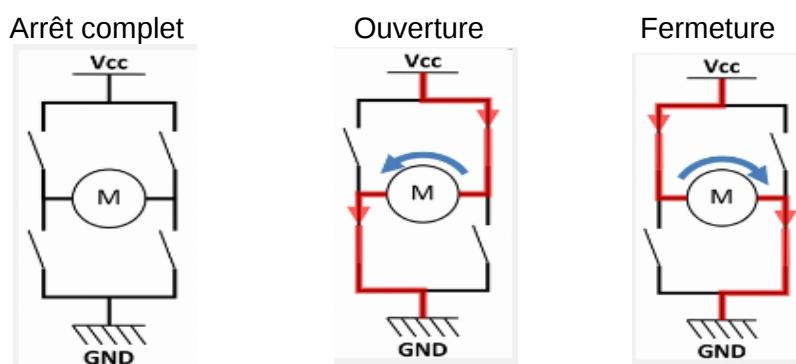
3. Choix de la carte de commande - L298N

Cette carte de contrôle va nous permettre de gérer 2 vannes en même temps, elle fonctionne comme un double pont en H.



Elle sera branchée sur l'ESP32, sur 3 broches.

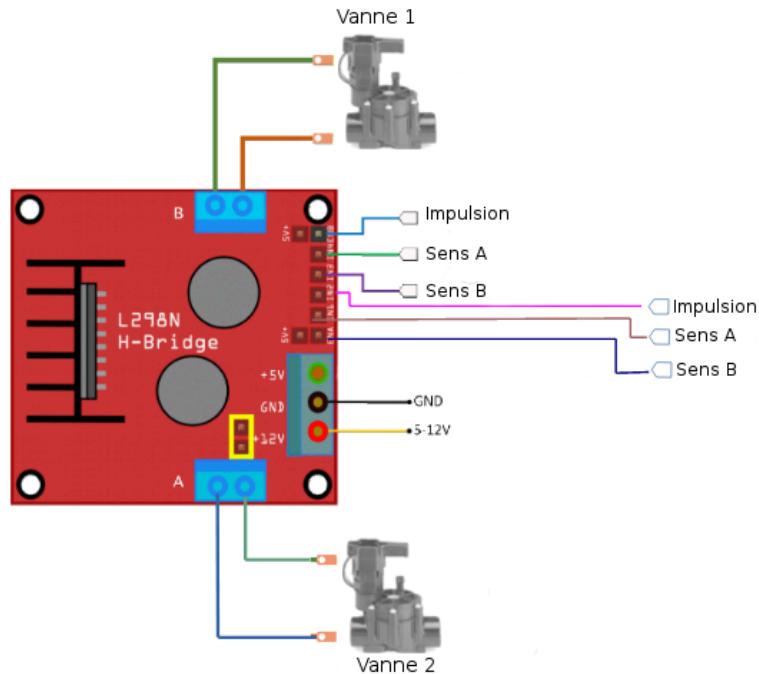
→ Le double Pont en H :



Lorsque les deux interrupteurs sont ouverts, le système est arrêté totalement.

En revanche si l'on ouvre deux interrupteurs opposés et que les deux autres sont fermés alors cela va ouvrir la vanne. Et si l'on inverse les ouvertures des interrupteurs alors on va fermer la vanne.

Pour mieux comprendre le câblage :

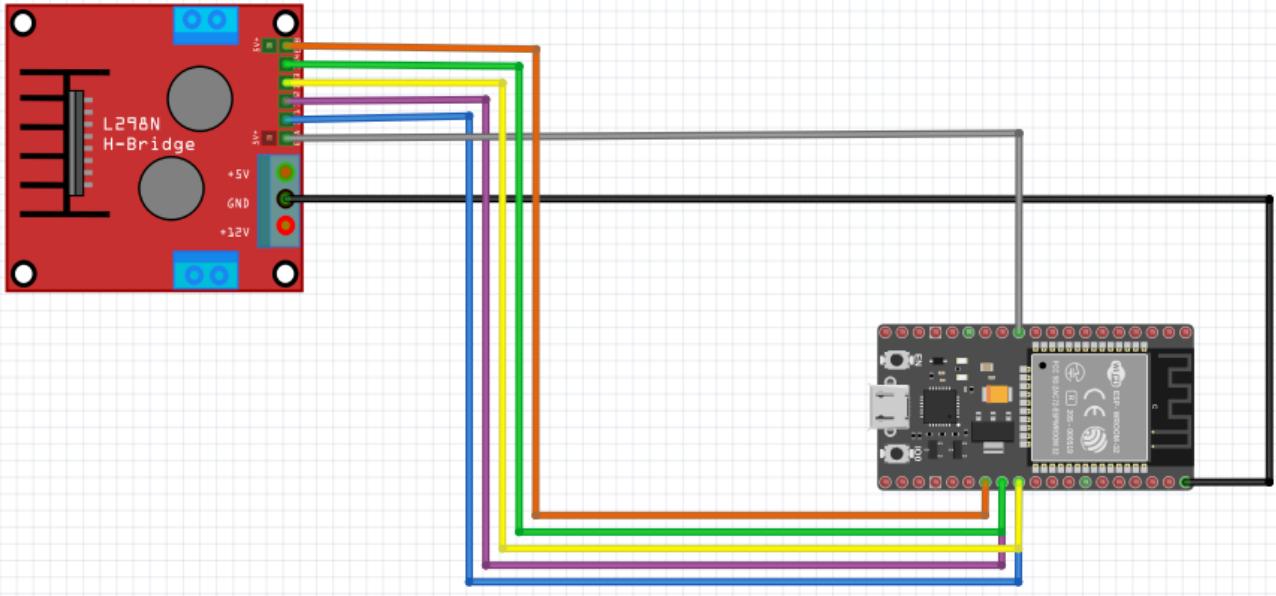


On voit bien qu'avec une seule carte on peut commander 2 électrovannes en même temps. Pour piloter l'ouverture ou la fermeture des vannes, on va devoir brancher 6 fils sur l'ESP32, pour chaque vanne on a une sortie pour l'impulsion et 2 autres pour le sens.

Mais en étudiant le fonctionnement de cette carte, on peut finalement occuper seulement 4 broches de l'ESP32 puisqu'on va brancher tous les fils correspondant au sens au même endroit donc au lieu d'utiliser 8 broches de sens pour 4 vannes, on va en utiliser que 2.

Ce qui nous permet un gain de place sur notre ESP32.

4. Schéma de câblage sur notre ESP32



Orange	EnB	Gpio04
Vert	in4	Gpio16
Jaune	in3	Gpio17
Violet	in2	Gpio16
Bleu	in1	Gpio17
Gris	EnA	Gpio27
Noir	GND	GND

Les broches EnA et EnB correspondent aux broches impulsions, EnA pour la vanne 1 et EnB pour la vanne 2.

in1 et in3 sont les broches de sens B.

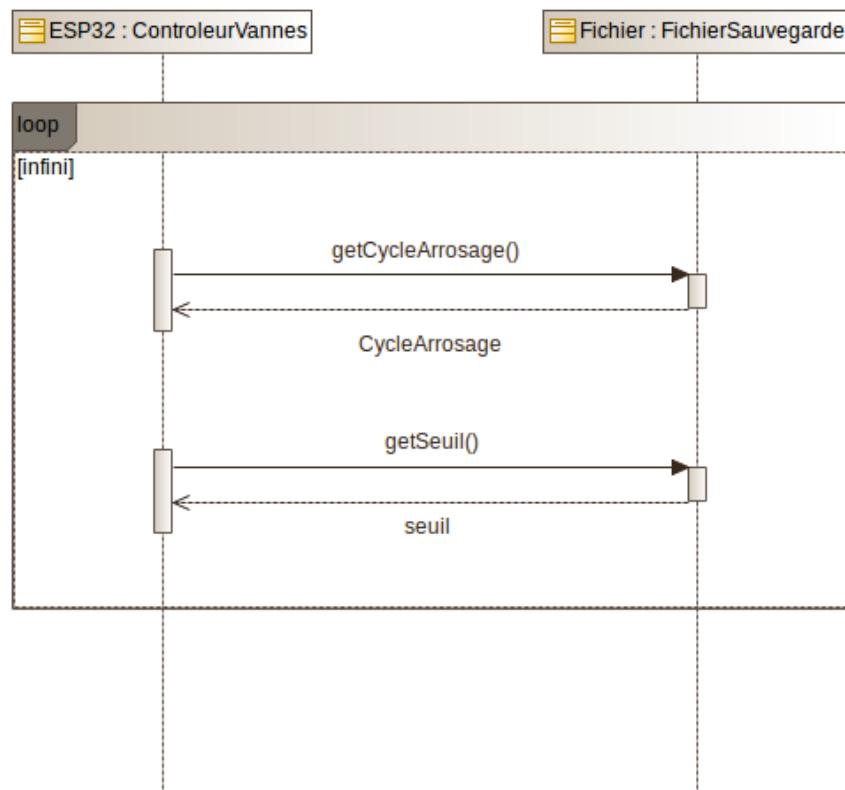
in2 et in4 sont les broches de sens A.

5. Tâche n°3 : « Mettre à jour les cycles d'arrosage et les seuils »

Description du cas d'utilisation

Nom du cas d'utilisation	Mettre à jour les cycles d'arrosages et les seuils
Utilisateur concerné	Maraîcher
Étudiant en charge	Étudiant numéro 1, Théo MICHAUD
Préconditions	Avoir une carte SD banchée sur l'ESP32 dans lequel se trouve déjà les cycles d'arrosage et les seuils
Scénario nominal	Il s'agit d'aller lire périodiquement pour voir si un nouveau cycle d'arrosage est disponible ou un changement de seuil dans un fichier qui se trouve dans la carte SD.
Post-condition(s)	les cycles d'arrosages et les seuils sont à jour

Diagramme de séquence



6. Tâche n°4 : «Mesurer l'humidité du sol»

Description du cas d'utilisation

Nom du cas d'utilisation	Mesurer l'humidité du sol
Utilisateur concerné	Maraîcher
Étudiant en charge	Étudiant numéro 1, Théo MICHAUD
Préconditions	Le capteur est présent et fonctionne correctement
Scénario nominal	Lire l'état du capteur toutes les 15 minutes. Une autre personne du groupe viendra chercher la valeur et la rangera dans le fichier sur la carte SD et dans la BDD.
Post-condition(s)	Pour chaque changement d'état du capteur, l'information est transmise

Diagramme de séquence

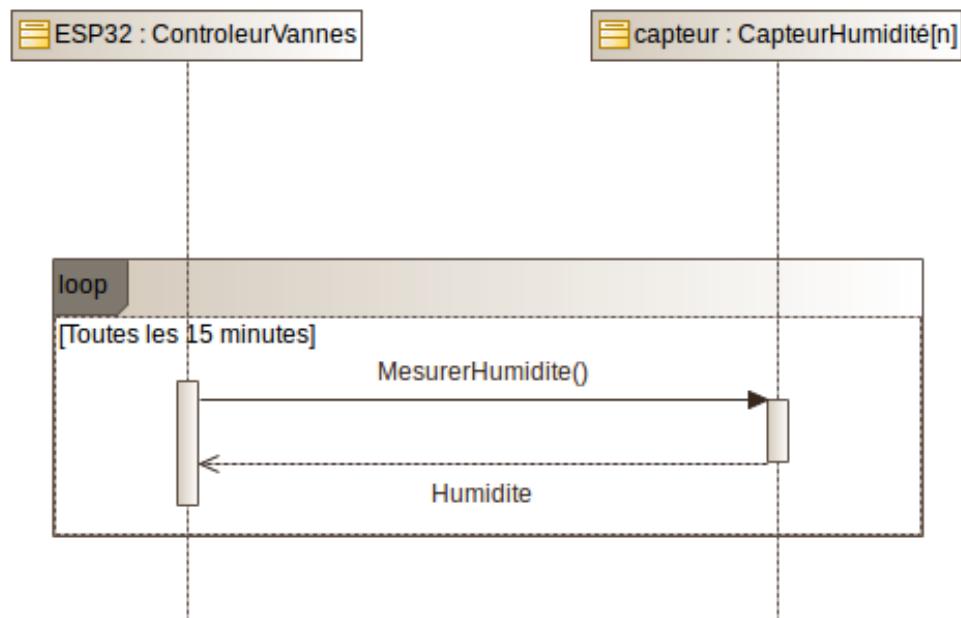
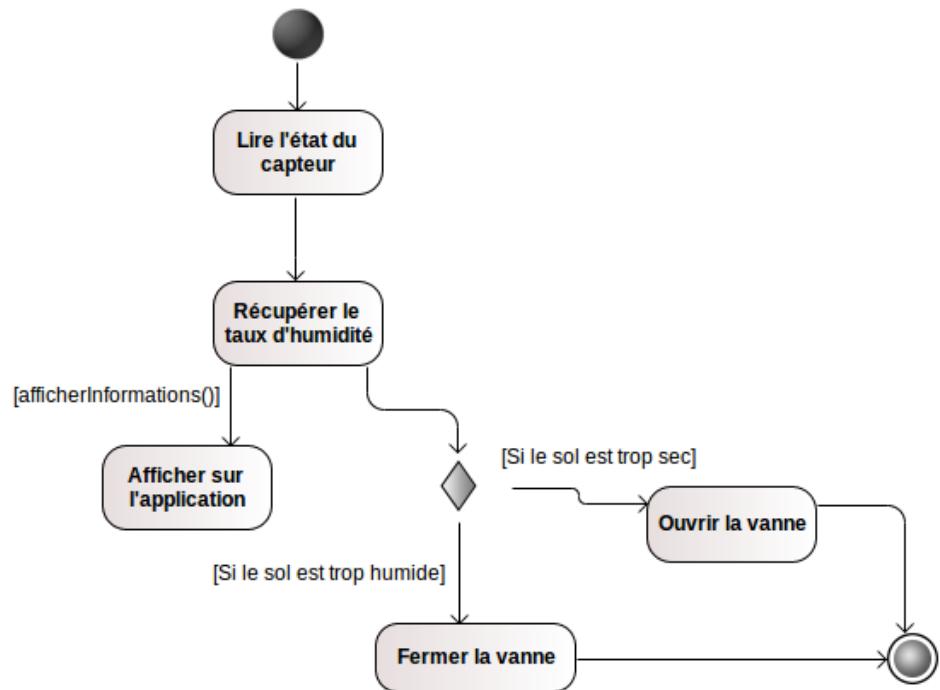
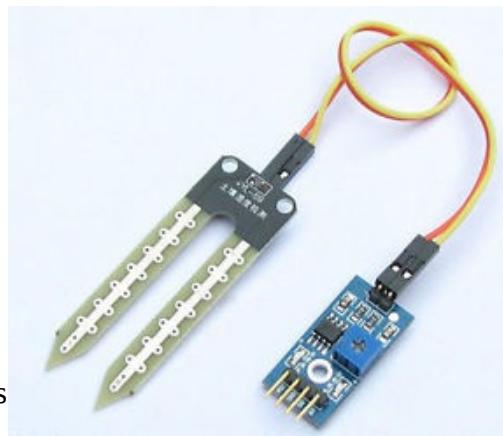


Diagramme d'état transition

7. Choix du capteur d'humidité : YL-69

Ce capteur mesure l'humidité du sol à partir des **changements de conductivité électrique de la terre** (la résistance du sol augmente avec la sécheresse).

- Une sortie digitale avec un seuil réglable par potentiomètre permet de déclencher une pompe d'arrosage ou une alarme par exemple.
- Une seconde sortie analogique permet de suivre les fluctuations précises de l'humidité du sol.



La fourche du capteur se plante verticalement dans la terre (pot de fleurs, jardin...). On mesure la résistance électrique entre les deux électrodes. Un comparateur à seuil active une sortie digitale quand un seuil réglable est dépassé.

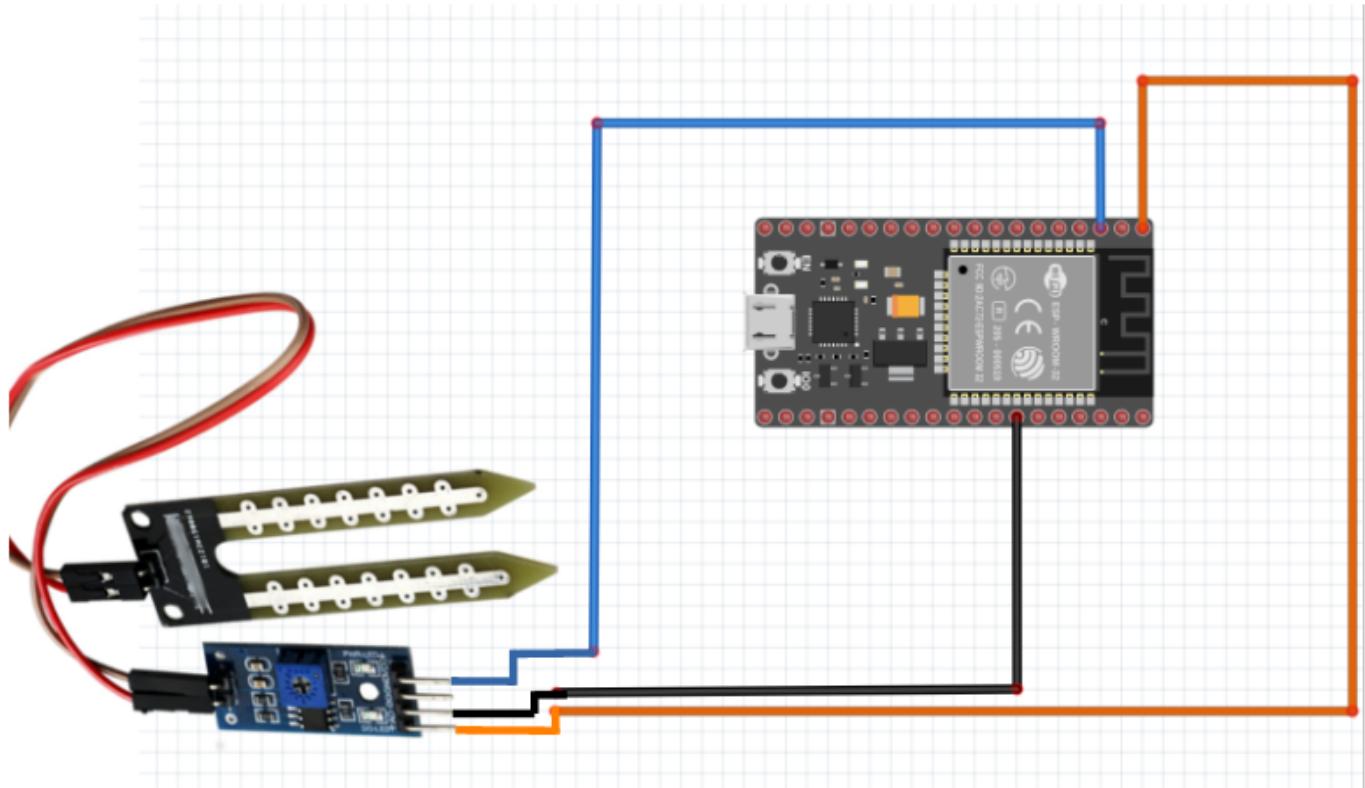
Le capteur se comporte comme une résistance variable.

Le seuil d'humidité du sol sera programmé en fonction de ce qui a été planté (carotte ...) et doit être programmable.

Il faut donc un retour de la valeur du taux d'humidité : en analogique.

Le retour de cette valeur étant impossible en digital qui permet seulement de régler un seuil à l'aide du potentiomètre implanté sur la carte

<u>Données techniques du capteur d'humidité Arduino YL-69</u>	
<i>Tension de fonctionnement</i>	3,3V-5V
<i>Module avec 2 sorties</i>	Une analogique et une numérique. La sortie numérique est plus précise.
<i>Taille du PCB</i>	3cm * 1.6cm
<i>Indicateur de tension</i>	(LED rouge)
<i>Indicateur de sortie numérique</i>	(LED verte)
<i>Description des entrées / sorties</i>	VCC : 3V-5V GND : GND DO : digital output interface (0 ou 1) AO : Analog Output Interface

Schéma de câblage pour programmer ce capteur

Bleu	A0
Noir	gnd
Orange	3.3V

Sur ce schéma, on branche notre capteur sur 3 broches disponibles des 4 sur celui-ci.

La Gpio36 correspond à notre broche A0 sur le capteur.

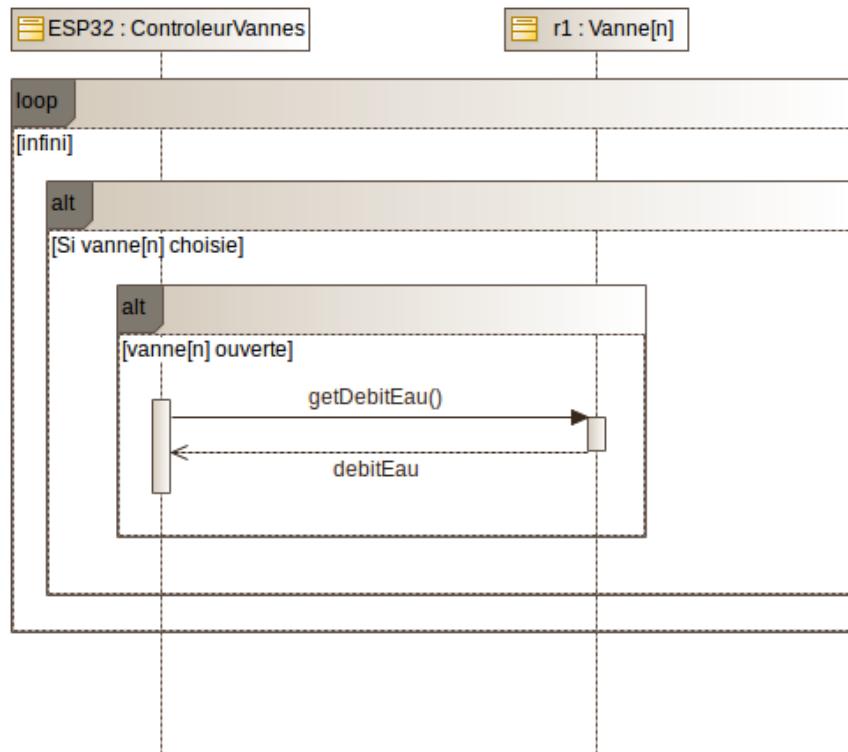
Nous pouvons aussi en brancher 3 de plus sur cette carte. Il suffit de répéter ce montage, mais en branchant les nouvelles broches A0 sur les Gpio39 / 34 / 35.

9. Tâche n°5 : «Mesurer le débit d'eau»

Description du cas d'utilisation

Nom du cas d'utilisation	Mesurer le débit d'eau
Utilisateur concerné	Maraîcher
Étudiant en charge	Étudiant numéro 1, Théo MICHAUD
Préconditions	Que les vannes soient ouvertes et que l'eau passe dedans
Scénario nominal	<p>Pour chaque vanne, le nombre d'impulsions en provenance du capteur associé est comptabilisé pendant un certain temps.</p> <p>Le débit d'eau est calculé et mémorisé.</p> <p>Si le débit dépasse le seuil fixé sur la période, un signal est envoyé à contrôler les vannes pour fermer la vanne en question.</p>
Post-condition(s)	Le maraîcher est prévenu en cas de débit inhabituel

Diagramme de séquence



10. Choix du capteur – YF-B5

Le capteur de débit d'eau est composé d'un corps en cuivre, d'un rotor à eau et d'un capteur à effet Hall.

Quand l'eau circule à travers le rotor, le rotor roule. Sa vitesse change avec un débit différent. Le capteur à effet hall produit le signal d'impulsion correspondant. Celui-ci est adapté pour détecter le débit dans l'eau.



C'est le même système que dans un distributeur ou une machine à café.

Il va pouvoir mesurer de 1 à 30L/min.

Un capteur à effet Hall permet de mesurer une variation de champ magnétique. C'est donc un capteur inductif

. Nous avons besoin de 4 capteurs comme celui par boîtier, puisqu'à chaque sortie de vanne on veut calculer le débit d'eau.

<u>Données techniques du capteur YF-B5</u>	
<i>Courant maximum</i>	5V-24V
<i>Poids</i>	43g
<i>Plage de débit</i>	1 à 30 L/min
<i>Température de fonctionnement</i>	0°C à 80°C
<i>Température du liquide</i>	< 120°C
<i>Pression de fonctionnement</i>	< 2,0 MPa
<i>Température de stockage</i>	-25°C a +80°C

11. Choix Logiciel

1. Arduino

Avec cet outil, je vais pouvoir gérer l'ESP32, et donc contrôler les vannes.

Mais aussi pouvoir mesurer l'humidité du sol et mesurer le débit d'eau.



2. AppInventor

App Inventor est un outil de développement en ligne pour les téléphones et les tablettes sous Android. App Inventor est un OS créé par Google, et concurrent de l'Ios d'Apple qui équipe l'iPad, iPod touch et iPhone.



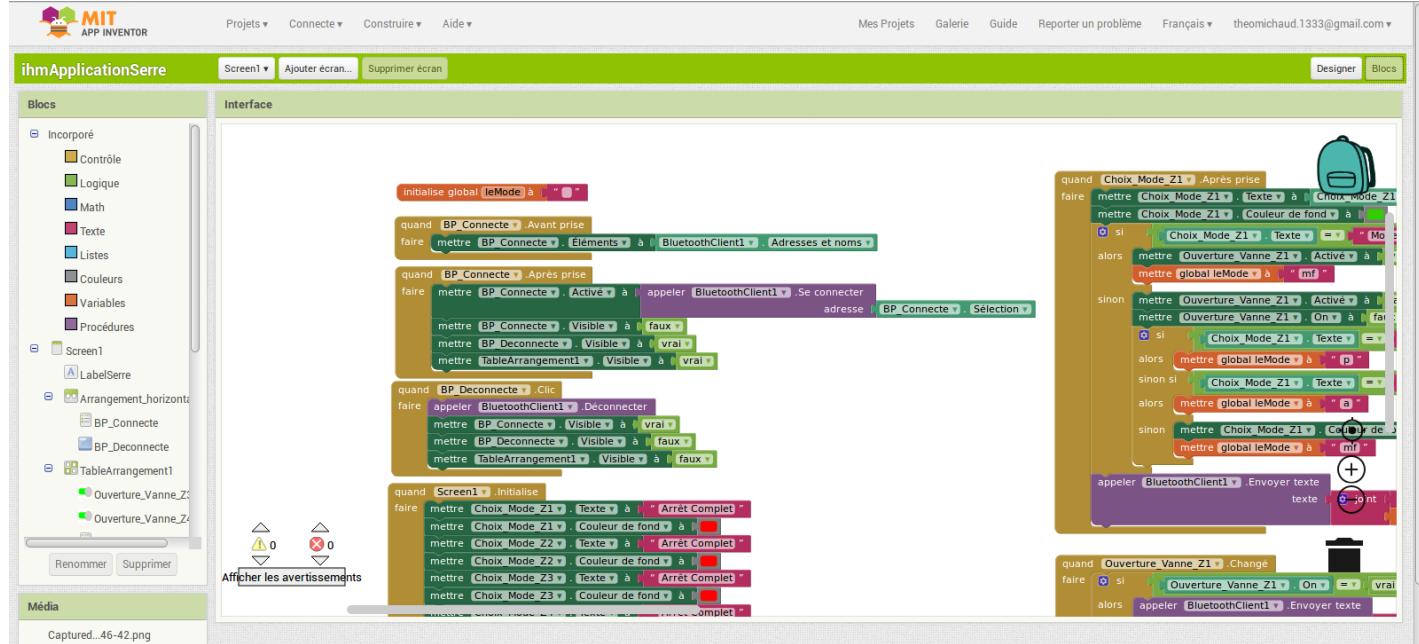
Cet outil m'a permis de créer mon application rapidement et simplement. En nous connectant dessus à l'aide d'un compte Google, nous pouvons créer une interface graphique facilement.

2 fenêtres sont proposées pendant le développement :

- Une pour la création de l'interface homme-machine : ce sera l'allure de votre application

The screenshot shows the MIT App Inventor interface with the project titled "ihmApplicationSerre". The central area displays a smartphone screen with a user interface for "SERRE EN DIRECT" (Greenhouse in Direct). The interface includes a header with "Gestion" and connection buttons, followed by four sections labeled "Zone 1" through "Zone 4", each with a "Choix du Mode" button and an "Ouverture Vanne" switch. The left sidebar shows the "Interface utilisateur" palette with various components like Bouton, Case à cocher, Sélectionneur de date, etc. The right sidebar shows the "Composants" and "Propriétés" panels for the current screen, including properties for "AccentColor", "Alignment horizontal", "Alignment vertical", and "PrimaryColor".

- Une pour la programmation par elle-même. Elle permettra par l'assemblage de blocs de créer le comportement de l'application ;



Il suffit ensuite de connecter son smartphone, grâce à l'application MIT AI2 Companion, et de téléverser l'application sur le smartphone pour l'utiliser.

3. Fritzing :

C'est un logiciel libre de conception de circuit imprimé qui permet de concevoir de façon entièrement graphique le circuit et d'en imprimer le typon.

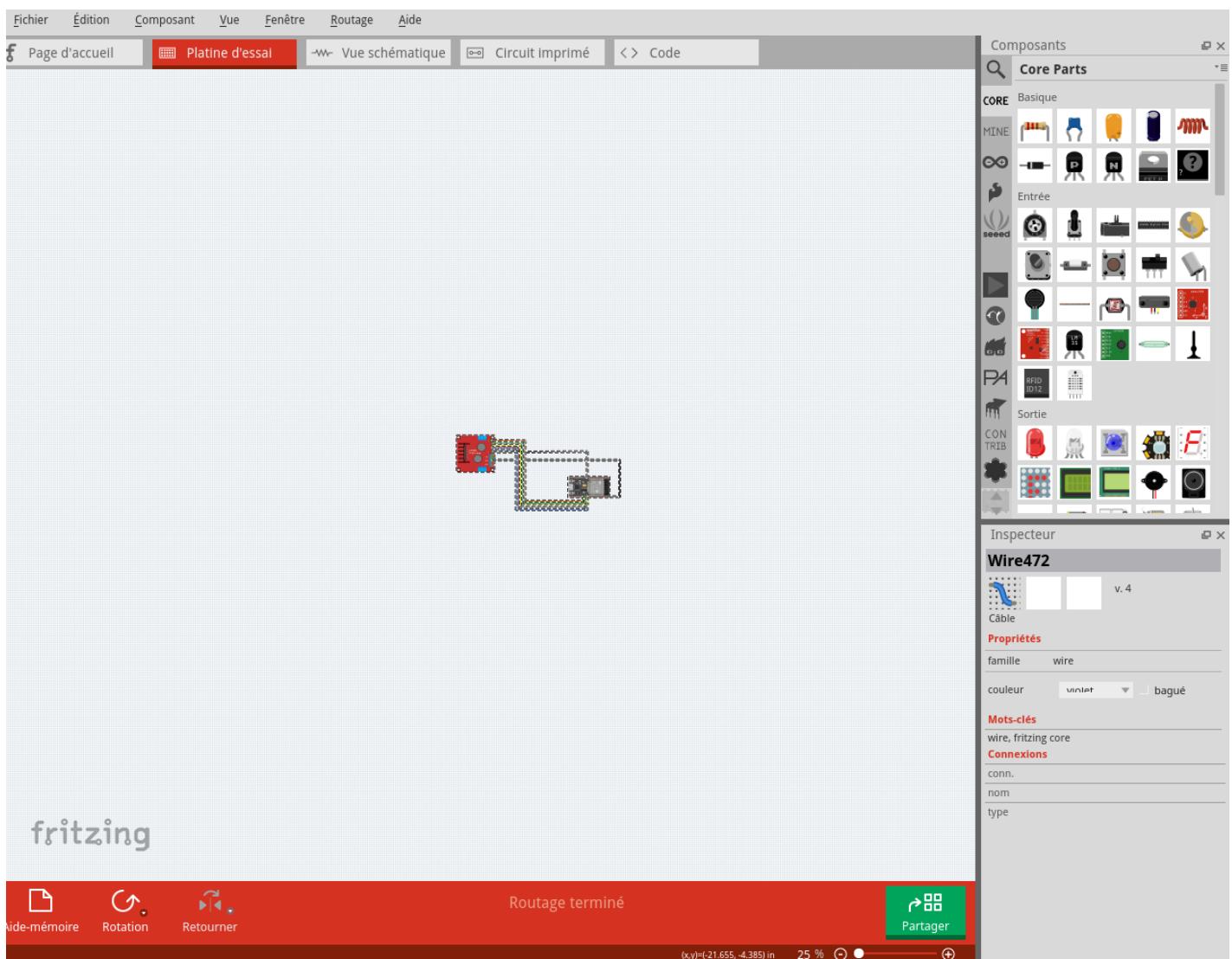
Cela m'a permis de créer mes schémas de câblage des différents éléments que j'ai programmé.



Les composants de bases fournis par le logiciel sont de type Arduino, il a donc fallu chercher sur Internet pour trouver le composant ESP32 et le rajouter dans les composants du logiciel.

Voici un aperçu du logiciel :

Avec à droite nos choix de composants; au milieu notre platine, afin créer les schémas.



II/- Conception préliminaire : Étudiant 2, AUVÉ Killian

1. Préambule

Conformément au cahier des charges, mais aussi au diagramme de classe, l'étudiant n°2 a été affecté aux tâches :

✗ Mesurer la température, l'humidité de l'air et l'ensoleillement

✗ Envoyer les données de la serre

✗ Mettre à jour la base de données

Pour effectuer l'envoi des données de la serre, l'étudiant n°2 va devoir réutiliser les tâches de l'étudiant n°1.

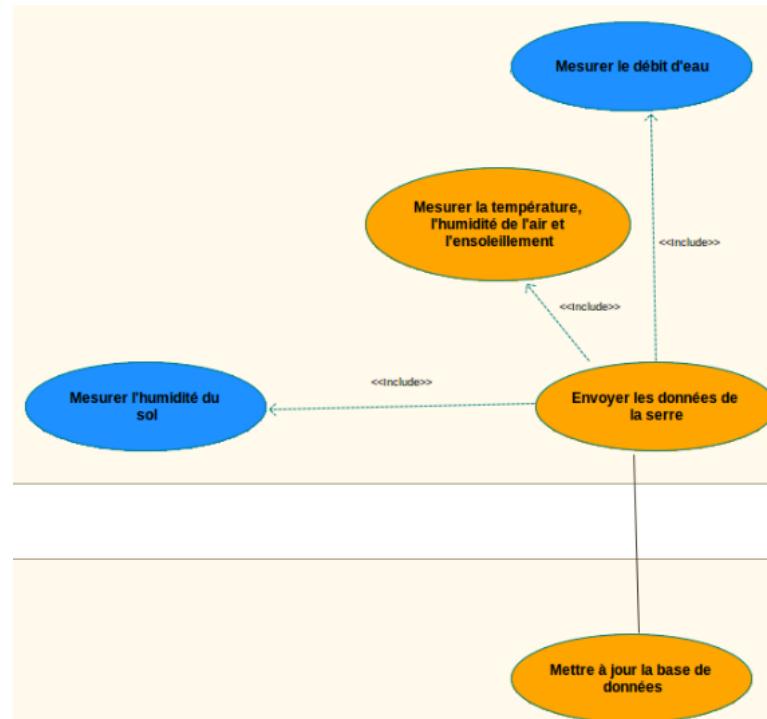


Diagramme des cas d'utilisation : étudiant n°2

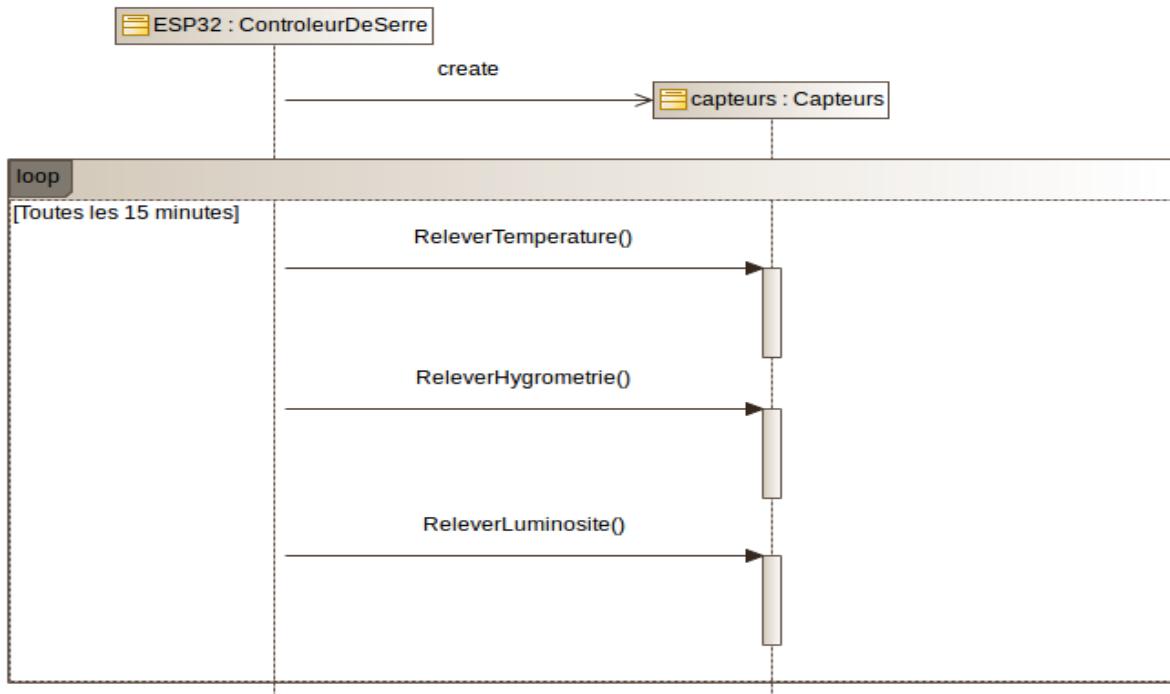
2. Tâche n°1 « Mesurer la température, l'humidité de l'air et l'ensoleillement »

Afin de pouvoir gérer la serre, nous avons besoin de récupérer les différentes données météorologiques obtenues grâce aux différents capteurs. Cette partie s'effectue de manière automatique, c'est-à-dire que toutes les 15 minutes, l'objet connecté va récupérer les mesures et les garder en mémoire pour pouvoir les exploiter.

Description du cas d'utilisation

Nom du cas d'utilisation	Mesurer la température, l'humidité de l'air et l'ensoleillement
Étudiant en charge	Étudiant numéro 2, Killian AUVÉ
Préconditions	Les différents capteurs sont branchés
Scénario nominal	Toutes les 15 minutes, la température, l'humidité de l'air et l'ensoleillement sont relevés à l'aide de capteurs (BME280 pour la température et l'humidité et TSL2591 pour la luminosité) afin de vérifier si les conditions sont bonnes pour le développement des plantes. Ils sont récupérés sur l'objet connecté ESP32.
Post-condition(s)	Les données ont été récupérées

Diagramme de séquence



3. Choix des technologies

1. Capteur de température, d'humidité : BME280

Le capteur BME280 permet de mesurer la température, l'hygrométrie, la pression de l'air ainsi que l'altitude. Dans notre cas, il va nous permettre de relever la température et l'hygrométrie de la serre.

Il peut utiliser le bus I2C ou le bus SPI. Dans notre cas, nous utiliserons le bus I2C.

Voici un tableau des caractéristiques du BME280 que nous avons retenu.

Poids	1g
Dimensions	19 x 18 x 3 mm
Tension d'alimentation	3.3 - 5V
Tension de fonctionnement	3.3V ou 5V
Plage de température	-40°C à 85°C
Plage d'humidité	0 – 100 %
Plage de pression	300 – 1100 hPa
Type de sortie	Numérique
Prix	23€



BME280 retenu

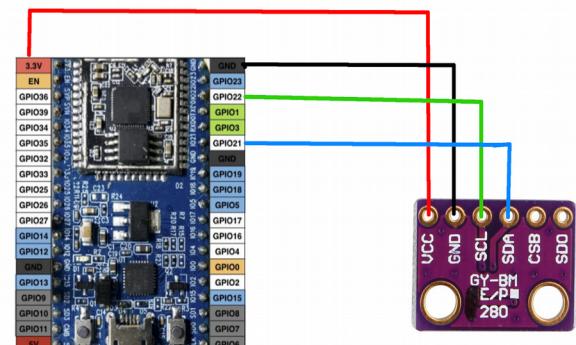


Schéma cablage ESP32 avec le capteur BME280

À la vue du schéma de câblage, nous pouvons distinguer deux types de broches :

- Les broches d'alimentation

Vin qui correspond à la broche d'alimentation (3-5V)

GND qui correspond au GND

Les 3Vo qui correspondent à la sortie de 3.3V

- Les broches I2C

SDA : Signal de données (21)

SCL : Signal d'horloge (22)

2. Capteur de luminosité : TSL2591

Le capteur TSL2591 permet de mesurer une intensité lumineuse de 188 µLux à 88000 Lux. Sa consommation est extrêmement faible avec 0,4mA en activité et moins de 5µA quand il est en veille.

Voici un tableau des caractéristiques du TSL2591 que nous avons retenu.

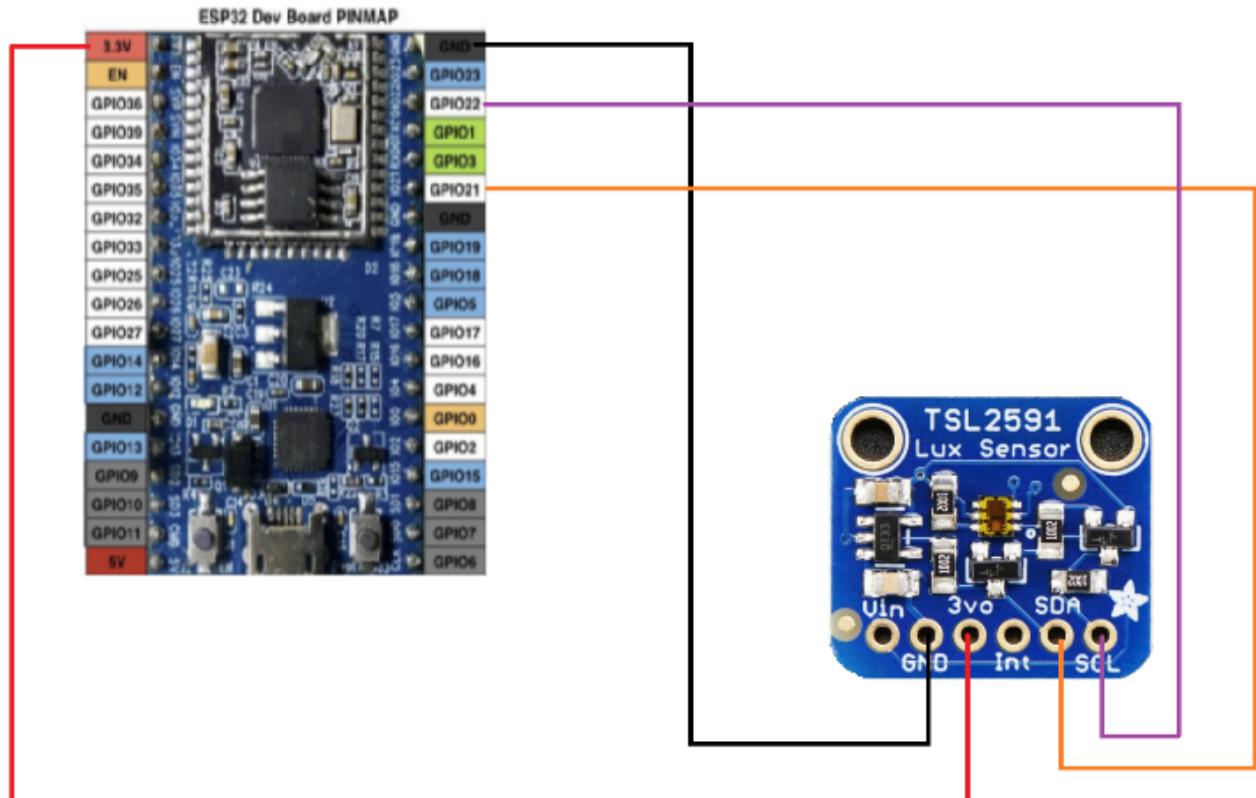
Alimentation	3.3 VCC
Signaux	3.3/5V
Régulateur	Intégré
Interface	I2C – adresse 0x29 fixe -
Spectre de réponse	Proche de l'œil humain
Plage de mesure	188µ - 8800 lux
Dimensions	19 x 17 x 3 mm
Référence fabricant	1980
Signal de sortie	I2C 7-bit (adresse 0x29 fixe)
Sortie	Analogique
Prix	8.70€

Ce capteur convient très bien à notre besoin, car l'éclairement lumineux d'un ciel bleu à midi donc très ensoleillé est de 20 000 lux ; or l'éclairement maximum du capteur est de 88 000 lux.



TSL2591 retenu

3.Schéma de câblage du TSL2591



À la vue du schéma de câblage, nous pouvons distinguer deux types de broches :

- Les broches d'alimentation

Vin qui correspond à la broche d'alimentation (3-5V)

GND qui correspond au GND

Les 3Vo qui correspondent à la sortie de 3.3V

- Les broches I2C

SDA : Signal de données (21)

SCL : Signal d'horloge (22)



4. Principes physiques

1. Température

La sonde électrique est constituée d'une seule tige de métal dont la résistance électrique varie en fonction de la température.

Plus la température du métal augmente, plus sa résistance augmente elle aussi. Il suffit de mesurer la valeur de la résistance électrique pour obtenir la température ambiante.

2. Hygrométrie

On mesure la capacité d'un condensateur hydrophile. La tension varie selon l'humidité du condensateur. La tension ainsi générée nous indique le pourcentage d'humidité.

3. Luminosité

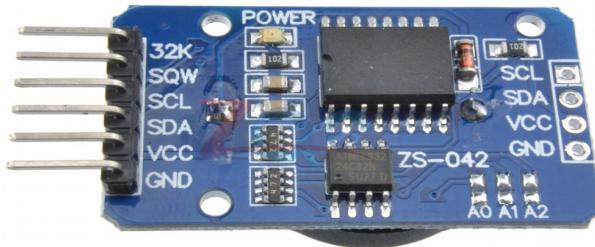
Le soleil envoie des ondes électromagnétiques (lumière) vers la Terre. La lumière est constituée d'un nombre extrêmement grand de photons contenant un peu d'énergie. Lorsqu'un capteur reçoit de l'énergie, il est capable de libérer des électrons. Comme le courant correspond à un déplacement d'électrons, une tension naît donc aux bornes du capteur. Ainsi en fonction de l'intensité lumineuse extérieure, le capteur créera un courant plus ou moins fort.



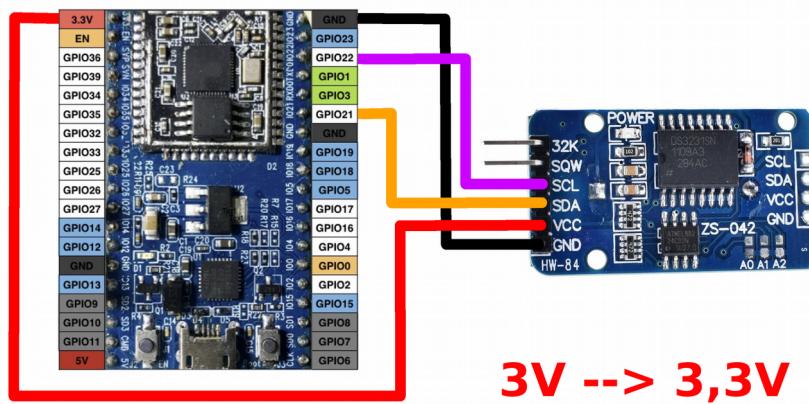
5. Module RTC : DS3231SN

Nous avons besoin d'obtenir la date et l'heure pour savoir quand les données ont été relevées. Afin de pouvoir insérer les données dans la base de données.

Pour cela, nous allons utiliser le module DS3231SN, il utilise le bus I2C pour communiquer avec l'objet connecté.



Module RTC retenu



Câblage du module RTC à l'ESP32

6. Bus I2C

Afin de faciliter la transmission des données, les capteurs utilisent le bus I2C.

Le bus I²C (Inter Integrated Circuit) fait partie des bus série : 3 fils pour faire tout passer. Il a été développé au début des années 1980, par Phillips pour minimiser les liaisons entre les circuits intégrés numériques de ses produits (Téléviseurs, éléments Hi-Fi, magnétoscopes ...).

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils : un signal de données (SDA), un signal d'horloge (SCL), et un signal de référence électrique (masse).

Il s'agit d'une liaison en mode série, ce qui signifie que la vitesse de transfert sera plus faible qu'avec un bus de type parallèle. Le bus I²C permet cependant des échanges à la vitesse de 100 kbit par seconde.

Le premier fil, SDA (Signal DAta), est utilisé pour transmettre les données. L'autre fil, SCL (Signal CLock) est utilisé pour transmettre un signal d'horloge synchrone (signal qui indique le rythme d'évolution de la ligne SDA).

7. Tâche n°2 « Envoyer les données de la serre »

Une fois, les données météorologiques ont été récupérées, nous avons besoin de les envoyer au serveur afin de pouvoir les exploiter et être réutilisées.

Description du cas d'utilisation

Nom du cas d'utilisation	Envoyer les données de la serre
Étudiant en charge	Étudiant numéro 2, Killian AUVÉ
Préconditions	Mesure du débit d'eau, de l'humidité du sol et de l'air, de la température et de l'ensoleillement a été effectué et présente soit sur l'objet connecté, soit dans le fichier local
Scénario nominal	<p>On vérifie que la connexion à la base de données est bien effectuée.</p> <p>Si le serveur BDD n'est pas accessible, les données sont enregistrées dans un fichier local, l'incident est mémorisé.</p> <p>Si le serveur de BDD est accessible, on vérifie s'il y a des données dans le fichier local puis elles sont envoyées à la BDD distante pour être stocké.</p> <p>S'il n'y a pas de données à transmettre , on envoie les données que l'objet connecté relève toutes les 15 minutes.</p>
Post-condition(s)	Les données ont été envoyées au serveur

Diagrammes de séquence

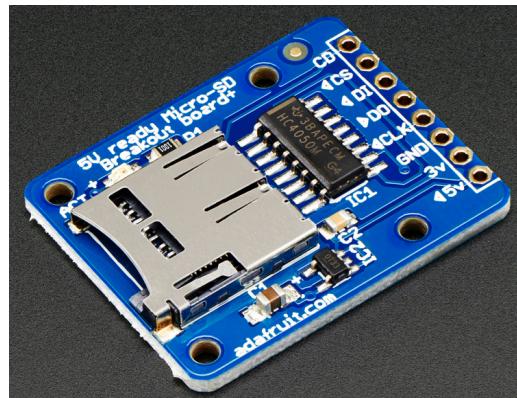




8. Choix des technologies

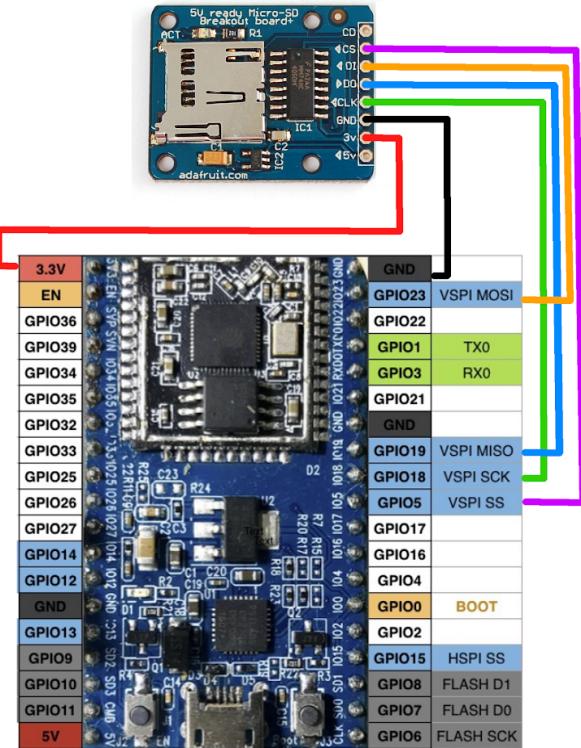
1. Stockage données:Shield SD

Pour pouvoir enregistrer les données dans un fichier local en cas de perte de connexion, il va falloir utiliser une carte SD; parce que la mémoire de l'esp32 est limitée. Mais pour pouvoir lire et écrire sur une carte SD, il faut un Shield SD qui sera relié à l'esp32.



Shield SD retenu

2. Schéma de câblage avec l'ESP32

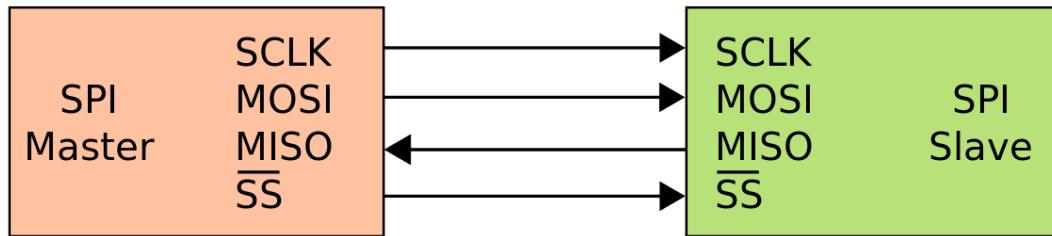


3V → 3,3V
GND → GND
CLK → 18 (SCK)
DO → 19(MISO)
DI → 23 (MOSI)
CS → 5 (SS)

3. Bus SPI

Le Shield SD va utiliser le bus SPI pour fonctionner.

Un bus SPI est un bus de communication série synchrone (avec horloge). Il y a toujours un maître (la carte ESP32) et un ou plusieurs esclaves (une carte SD dans notre cas).



Un bus SPI est composé de quatre fils :

- deux fils de données : MOSI (Master Output) et un MISO (Master Input)
- un fil d'horloge : SCK(Serial Clock)
- un fil d'adressage par périphérique (CS).

Les trois premiers fils sont communs à tous les périphériques d'un même bus SPI. Le quatrième fil (CS) est unique par périphérique et permet au maître de choisir : quel périphérique souhaite-t-il communiquer.

9. Interaction avec un Shield SD

L'interaction avec le Shield SD est nécessaire dans le fonctionnement de notre projet de supervision de serre ; en permettant d'écrire sur la carte SD.

1. Inclusion des bibliothèques

Pour pouvoir interagir avec un Shield SD sur Arduino, il faut inclure la bibliothèque "SD" et "SPI".

```
#include <SPI.h>
#include <SD.h>
```

2. Définition de la broche

il faut définir la broche sur lequel le Shield SD va se connecter.

```
SD.begin(broche_cs)
```

3. Vérification de l'existence d'un fichier

Ce code initialise le port série (pour debug), le port SPI et la carte SD.

Pour vérifier la présence d'un fichier ou d'un dossier sur la carte SD, vous pouvez utiliser la fonction SD.exists().

```
SD.exists(chemin_fichier);
```

4. Ouverture d'un fichier

Pour ouvrir un fichier, il convient d'utiliser la fonction SD.open().

Cette fonction permet d'ouvrir un fichier en lecture (défaut) ou en écriture.

```
File fichier = SD.open(chemin_fichier, FILE_WRITE);
```

Pour pouvoir écrire du texte, il suffit d'utiliser la fonction println sur le fichier.

```
fichier.println(F("Bonjour le monde"));
```

5. Bibliothèques Wi-Fi

Pour pouvoir transmettre les données de l'objet connecté vers la base de données, nous allons devoir utiliser la bibliothèque **MySQL Connector/Arduino**. C'est une bibliothèque créée par Charles A. Bell qui permet de se connecter à un réseau et une base de données, et de lancer des requêtes SQL depuis l'ESP32 afin de pouvoir insérer, sélectionner des données et mettre à jour la base de données.

Pour l'utiliser, il suffit de l'inclure comme ci-dessous :

```
#include <WiFi.h> // Use this for WiFi instead of Ethernet.h
```

Inclusion de la bibliothèque WiFi

```
#include <MySQL_Connection.h>
```

Inclusion de la bibliothèque MySQL connection

```
#include <MySQL_Cursor.h>
```

Inclusion de la bibliothèque MySQL Cursor

```
#include <Ethernet.h>
```

Inclusion de la bibliothèque Ethernet

Ensuite, il faut renseigner l'adresse IP du serveur ainsi que les identifiants de connexion (login/ mot de passe) de la base de données de la manière qui suit :

```
IPAddress server_addr(172,18,58,14);
```

On spécifie l'adresse IP du serveur

```
char user[] = "snir";
```

On spécifie le login de la base de donnée

```
char password[] = "snir";
```

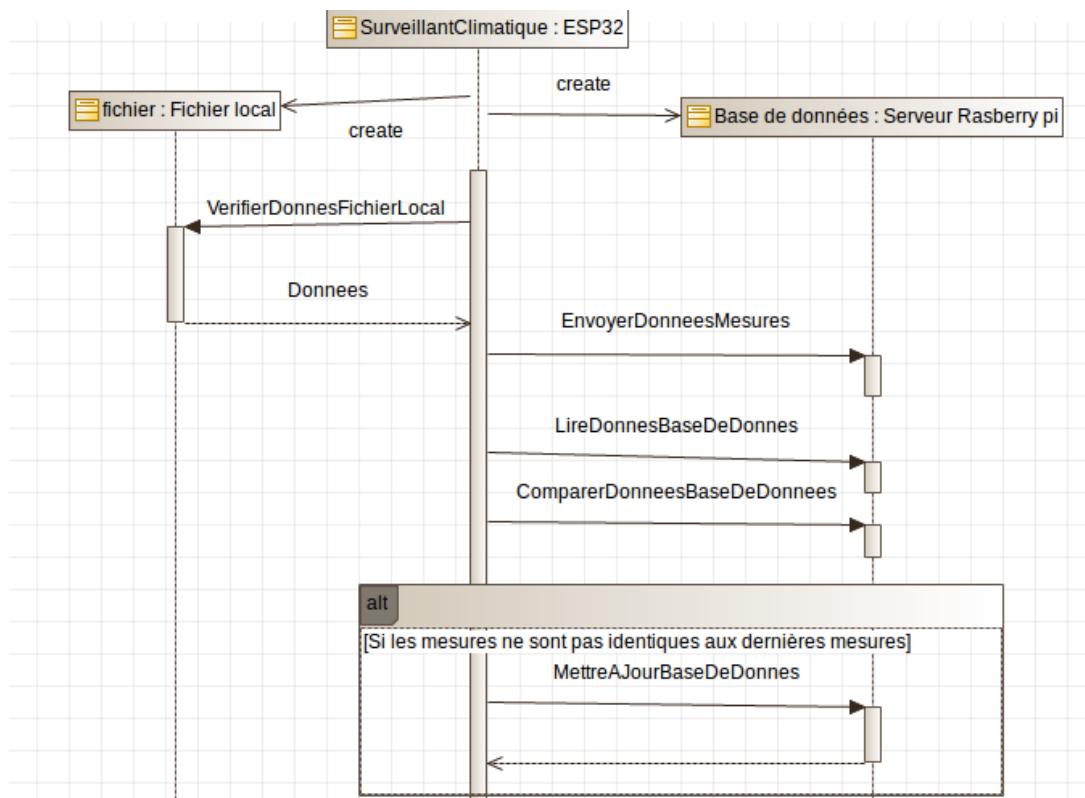
On spécifie le mot de passe de la base de donnée

10. Tâche n°3 « Mettre à jour la base de données »

Description du cas d'utilisation

Nom du cas d'utilisation	Mettre à jour la base de données
Étudiant en charge	Étudiant numéro 2, Killian AUVÉ
Préconditions	Les informations ont été envoyées sur la base de données
Scénario nominal	On lit les valeurs de la base de données et on les compare aux valeurs reçues. Si les valeurs reçues ne sont pas les mêmes que les précédentes, la base de données est mise à jour. Une perte de connexion avec une serre constitue un incident, il est mémorisé dans la base de données.
Post-condition(s)	La base de données a été mise à jour et s'il y a eu un incident, il est mémorisé dans la base de données

Diagramme de séquence



1. Les tables nécessaires dans la base de données

Pour pouvoir mettre à jour la base de données, je dois interagir avec plusieurs tables qui sont les suivantes

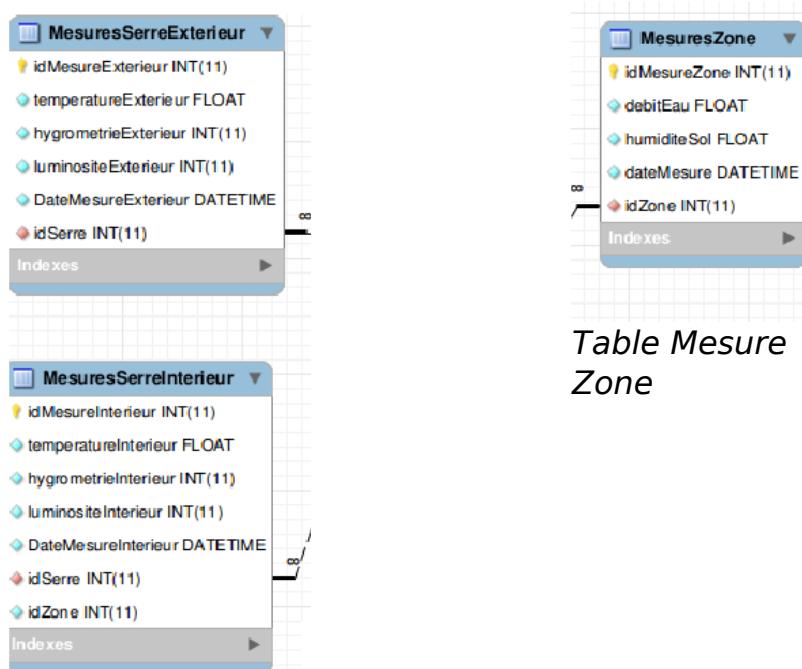


Table Mesure Zone

table des Mesures

2. Description des tables nécessaires

Dans cette table, je vais mettre à jour la température, l'hygrométrie, la luminosité intérieure de la serre et la date à laquelle les données ont été relevées.

<u>MesuresSerreInterieur</u>		
Nom du champ	Type du champ	Description
idMesureInterieur	INT : entier	Identifiant de la mesure
temperatureInterieur	FLOAT : réel	Valeurs de la température intérieure
hygrometrieInterieur	INT : entier	Valeurs de l'hygrométrie intérieure
luminositeInterieur	INT : entier	Valeurs de la luminosité intérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
idSerre	INT : entier	Identifiant de la serre
idZone	INT : entier	Identifiant de la zone

Dans cette table, je vais mettre à jour la température, l'hygrométrie, la luminosité extérieure de la serre et la date auquel les données ont été relevées.

<u>MesuresSerreExterieur</u>		
Nom du champ	Type du champ	Description
idMesureExterieur	INT : entier	Identifiant de la mesure
temperatureExterieur	FLOAT : réel	Valeurs de la température extérieure
hygrometrieExterieur	INT : entier	Valeurs de l'hygrométrie extérieure
luminositeExterieur	INT : entier	Valeurs de la luminosité extérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
IdSerre	INT : entier	Identifiant de la serre

Nous allons insérer dans cette table les données liées aux vannes que l'étudiant n°1 aura relevées.

<u>MesuresZone</u>		
Nom du champ	Type du champ	Description
idMesureZone	INT : entier	Identifiant de la mesure de la zone
debitEau	FLOAT : réel	Mesure du débit d'eau
debitEau	FLOAT : réel	Mesure de l'humidité du sol
dateMesure	DATETIME : Date	Date de la mesure
idZone	INT : entier	Identifiant de la zone

III/- Conception préliminaire : Étudiant 3, VILLERMIN Maxime

1. Préambule

Conformément au cahier des charges, mais aussi au diagramme de classe, l'étudiant n°3 a été affecté aux tâches :

- ✗ Programmer les paramètres d'une serre.
- ✗ Visualiser les données d'une serre.
- ✗ S'authentifier

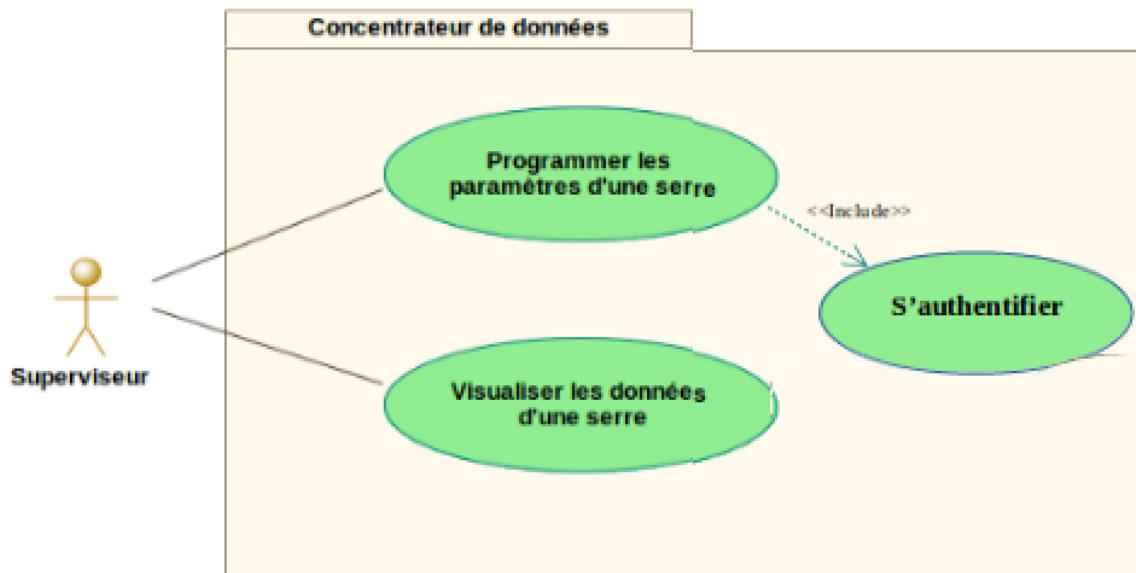


Diagramme des cas d'utilisation : étudiant n°3

2. Le site web

Le site doit permettre à l'un des responsables de serre :

- ▶ De visualiser les grandeurs physiques mesurées par les différents capteurs qui composent la serre (hygrométrie, température, luminosité, débit d'eau, etc.)
- ▶ De programmer des seuils d'alertes qui lui seront envoyés (Seuils de températures, débit d'eau, luminosité, hygrométrie, etc.)
- ▶ D'ajouter un ou plusieurs responsables et lui associer une serre
- ▶ De programmer des périodes d'arrosage et choisir son mode d'arrosage

Lorsque l'utilisateur accède au site web (à l'adresse : 172.18.58.213/ProjetSerre/PHP), il accède à la page d'accueil. L'image d'une serre y est présentée ainsi qu'un message de bienvenue.



IHM de la "page d'accueil"

Description de l'IHM « Page d'accueil »

n°1	Lien vers la page « visualiser » (visualisation des données)
n°2	Lien vers la page « programmer » (programmer les paramètres de la serre)
n°3	Fonction alerte (notification d'une alerte avec sa description)
n°4	Lien vers la page d'accueil (retourner sur la page d'accueil)

3. Les outils de développement

Pour pouvoir faire mes tâches, j'ai dû utiliser plusieurs outils de développement.

Voici les logiciels que j'ai utilisés :

- Modelio : outil de modélisation UML. Il m'a permis de faire mes diagrammes de séquences et d'états-transitions.



- Github : service web d'hébergement et de gestion de développement de logiciels.

Il m'a permis notamment d'héberger mes fichiers en ligne et pouvoir récupérer d'anciennes versions si nécessaire (back-up).



- Netbeans : environnement de développement intégré (EDI), *open source*.

Il permet de coder en C, C++, JavaScript, PHP, HTML, etc.

C'est sur cet outil que j'ai développé le site web.



Pour les langages, j'ai utilisé :

- HTML/CSS : langage informatique qui permet de mettre en forme un fichier HTML et de gérer le design de cette page avec un fichier CSS.



- JavaScript/PHP : Utilisé pour rendre le site web dynamique.



- Apache : Logiciel du serveur web.



- Bootstrap : En plus d'HTML/CSS, nous utilisons bootstrap, c'est un framework d'interface.



- SQL : est un langage informatique normalisé servant à exploiter des bases de données relationnelles



4. Choix et présentations des technologies

1. Raspberry PI

Raspberry PI 3 équipé du système RASPBIAN. Cette Raspberry aura pour rôle d'être notre serveur de base de données, mais aussi notre serveur web. Pour le serveur web, Apache et PHP seront installés.



2. Apache

Nous avons choisi comme logiciel de serveur web : Apache. Il est gratuit, fiable et facile à configurer. Les fichiers de configuration apache utilisent ASCII. Ce qui les rend extrêmement simples à administrer, car il est d'éditer les fichiers à l'aide de n'importe quel texte éditeur. C'est le logiciel de serveur web le plus utilisé (60 % de part du marché). Aussi, l'un de ses avantages est qu'il y a beaucoup de documentation, de cours et de tutoriel.

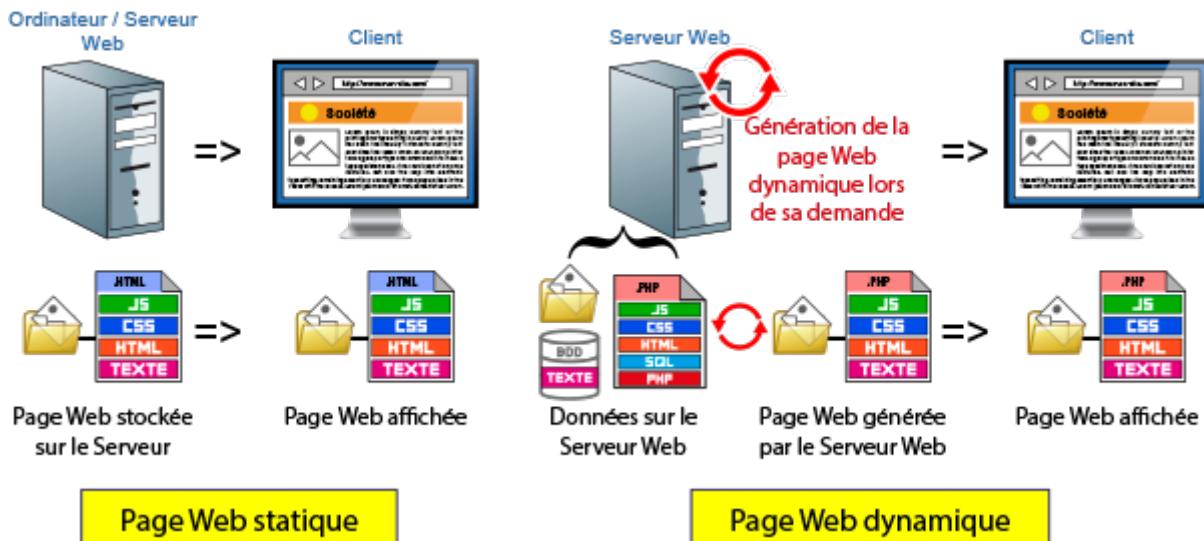


Installation d'Apache

<i>Commandes</i>	<i>Fonctions</i>
<code>sudo apt update</code>	Mets à jour la raspberry
<code>sudo apt install apache2</code>	Installe le serveur apache
<code>sudo chown -R pi:www-data /var/www/html/ sudo chmod -R 770 /var/www/html/</code>	Donne les droits d'accès au dossier d'Apache (/html/)
<code>wget -O verif_apache.html http://127.0.0.1</code>	Vérifie le bon fonctionnement d'Apache

3. PHP

PHP est principalement utilisé, pour rendre un site dynamique, c'est-à-dire que l'utilisateur envoie des informations au serveur qui lui renvoie les résultats modifiés en fonction de ces informations. A contrario, un site statique ne s'adapte pas aux informations fournies par un utilisateur. Il est enregistré sous forme de fichier une fois pour toutes, et livrera toujours le même contenu.



[Différence entre une page Internet statique et une page Internet dynamique](#)

Nous avons choisi d'utilisé PHP, car, comme pour Apache il est gratuit, fiable et facile à configurer. C'est le langage, pour son utilité, le plus utilisé (79 % de part du marché). Aussi, notre choix est dû au fait qu'il y a beaucoup de documentation, de tutoriel et que c'est aussi un langage vu en classe.

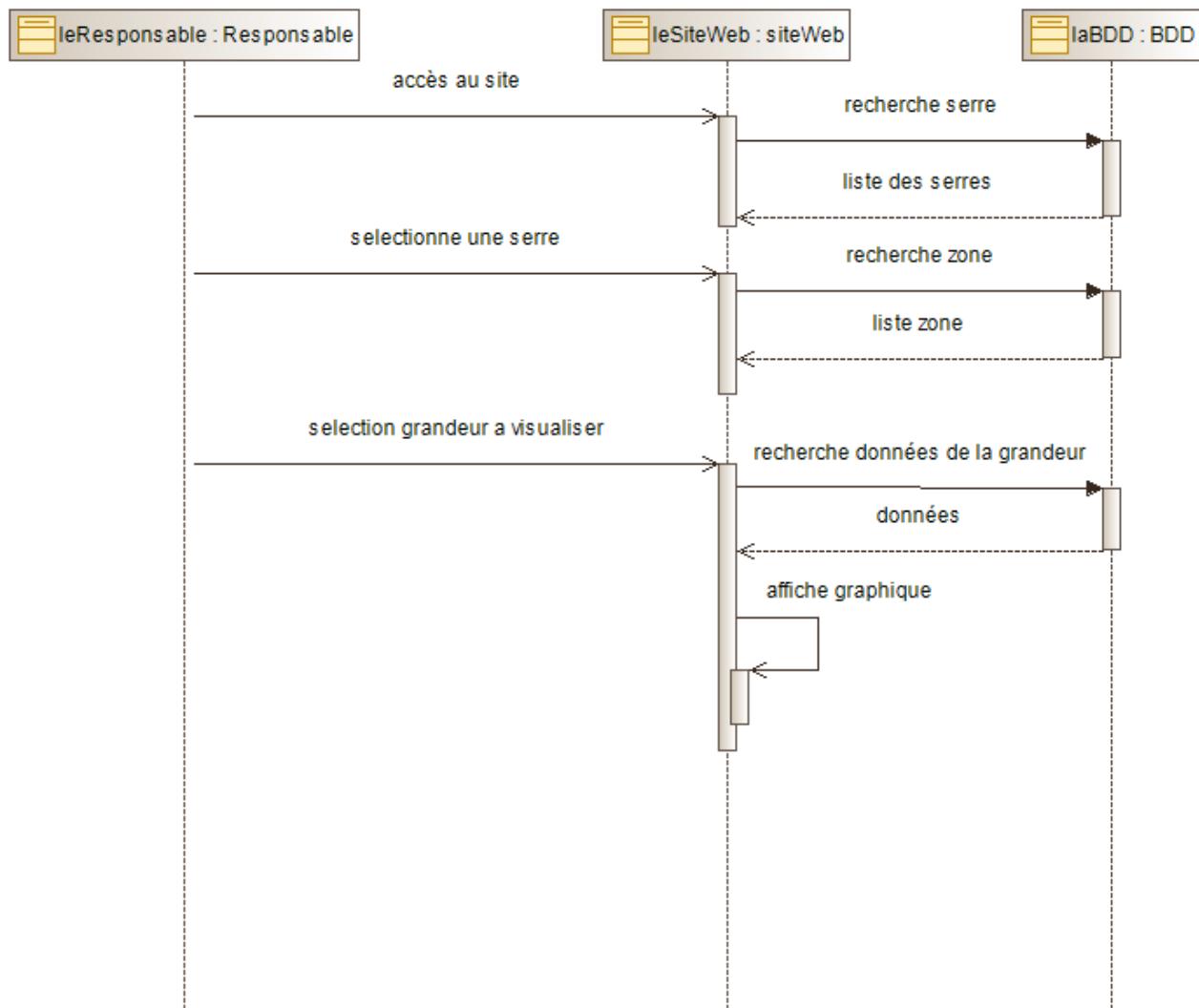
Installation de PHP

Commandes	Fonctions
sudo apt install php php-mbstring	Installation de PHP
sudo rm /var/www/html/index.html	Vérification du fonctionnement
echo "<?PHP phpinfo(); ?>" > /var/www/html/index.php	Création d'un fichier test en PHP à l'adresse (/var/www/html/index.php)

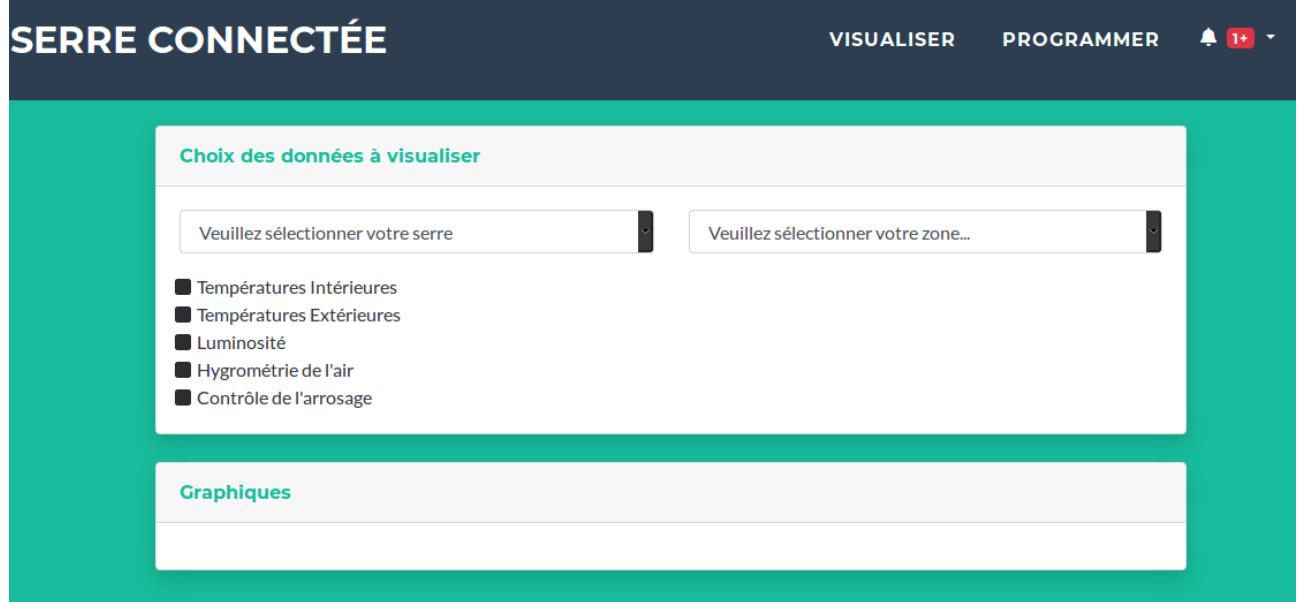
5. Tâche n°1 : Visualiser les données d'une serre

Description de cas d'utilisation

Nom du cas d'utilisation	Visualiser les données d'une serre
Utilisateur concerné	Superviseur
Étudiant en charge	Étudiant numéro 3, Maxime VILLERMIN
précondition(s))	Le responsable est sur le site web dans la partie « Visualiser ». Les données du contrôleur de serre sont dans la base de données.
Scénario nominal	Le responsable sélectionne une serre. Lorsque la serre est sélectionnée, il peut choisir d'afficher la luminosité, la température ou l'hygrométrie de l'air. Lorsqu'il choisit sa serre, la liste des zones est mise à jour. Le responsable peut alors visionner la période d'arrosage de la zone.
Post-condition(s)	Le responsable visualise la/les grandeurs sélectionnées sur des graphiques.

Diagramme de séquence

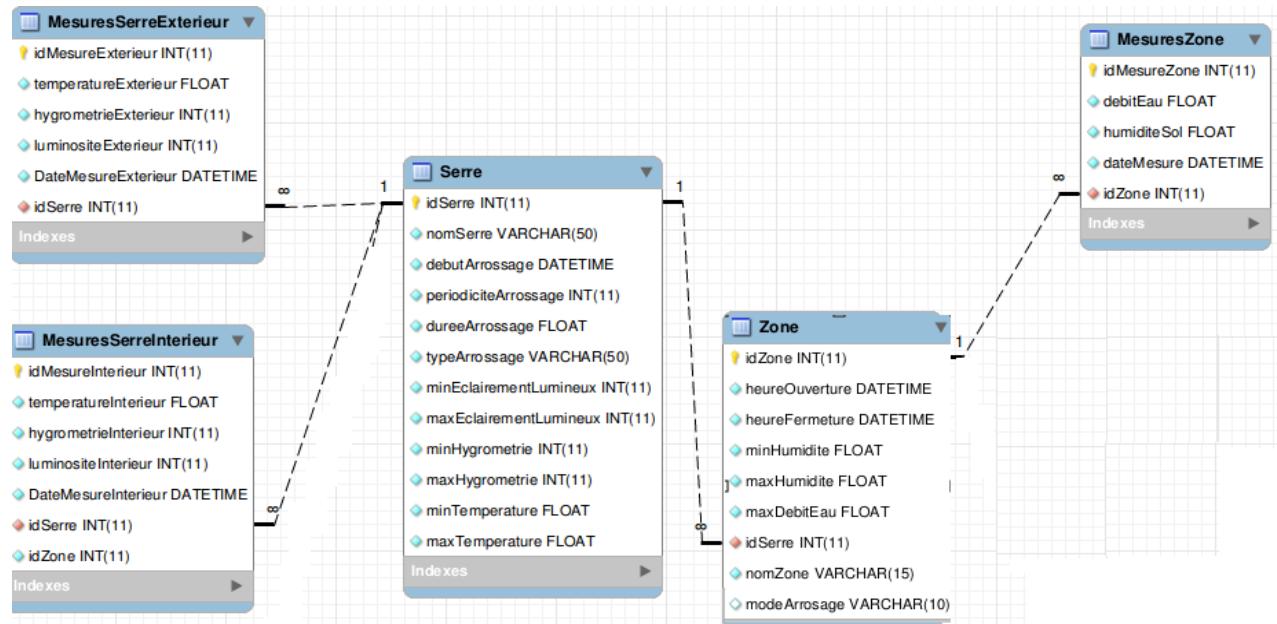
Chacune des grandeurs mesurées par les capteurs sera enregistrée dans la base de données. Le responsable devra accéder à la page « Visualiser ». Il pourra sélectionner la serre dans laquelle il souhaite visualiser des données (visualisation de l'humidité de l'air, températures extérieures et intérieures, luminosité) et ensuite il pourra choisir la zone où il pourra visualiser l'humidité du sol, le débit d'eau et les périodes d'arrosage correspondant à cette zone.



Présentation globale de la page "Visualiser"

1. Les tables nécessaires dans la base de données

Pour cette partie, nous avons besoin des tables suivantes :



2. Description des tables nécessaires

Le rôle de la table MesuresSerreExterieur permet de répertorier l'ensemble des mesures tirées des capteurs situés à l'extérieur de la serre.

<u>MesuresSerreExterieur</u>		
Nom du champ	Type du champ	Description
idMesureExterieur	INT : entier	Identifiant de la mesure
temperatureExterieur	FLOAT : réel	Valeurs de la température extérieure
hygrometrieExterieur	INT : entier	Valeurs de l'hygrométrie extérieure
luminositeExterieur	INT : entier	Valeurs de la luminosité extérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
IdSerre	INT : entier	Identifiant de la serre

Le rôle de la table Serre permet de répertorier l'ensemble des caractéristiques d'une serre.

<u>Serre</u>		
Nom du champ	Type du champ	Description
idSerre	INT : entier	Identifiant de la serre
nomSerre	VARCHAR(50) : chaîne de 50 caractères	Nom de la serre
debutArrosage	DATETIME : Date	Date du début d'arrosage de la serre
periodiciteArrosage	INT : entier	Périodicité de l'arrosage
dureeArrosage	FLOAT : réel	Durée de l'arrosage
typeArrosage	VARCHAR(50) : chaîne de 50 caractères	Type d'arrosage
minEclairementLumineux	INT : entier	Seuil minimum de l'éclairement lumineux
maxEclairementLumineux	INT : entier	Seuil maximum de l'éclairement lumineux
minHygrometrie	INT : entier	Seuil minimum de l'hygrométrie
maxHygrometrie	INT : entier	Seuil maximum de l'hygrométrie
minTemperature	FLOAT : réel	Seuil minimum de la température
maxTemperature	FLOAT : réel	Seuil maximum de la température

Le rôle de la table Zone permet de répertorier l'ensemble des caractéristiques d'une zone (seuils, identifiants, heure_d'ouverture et de fermeture)

<u>Zone</u>		
Nom du champ	Type du champ	Description
idZone	INT : entier	Identifiant de la zone
heureOuverture	DATETIME : Date	Heure d'ouverture de la vanne
heureFermeture	DATETIME : Date	Heure de fermeture de la vanne
minHumidite	FLOAT : réel	Seuil minimum d'humidité
maxHumidite	FLOAT : réel	Seuil maximum d'humidité
maxDebitEau	FLOAT : réel	Seuil maximum de débit d'eau
idSerre	INT : entier	Identifiant de la serre
nomZone	VARCHAR(15) : chaîne de 15 caractères	Nom de la zone
modeArrosage	VARCHAR(10) : chaîne de 10 caractères	Mode sélectionner des zones

Le rôle de la table MesuresZone permet de répertorier l'ensemble des mesures liées à une zone

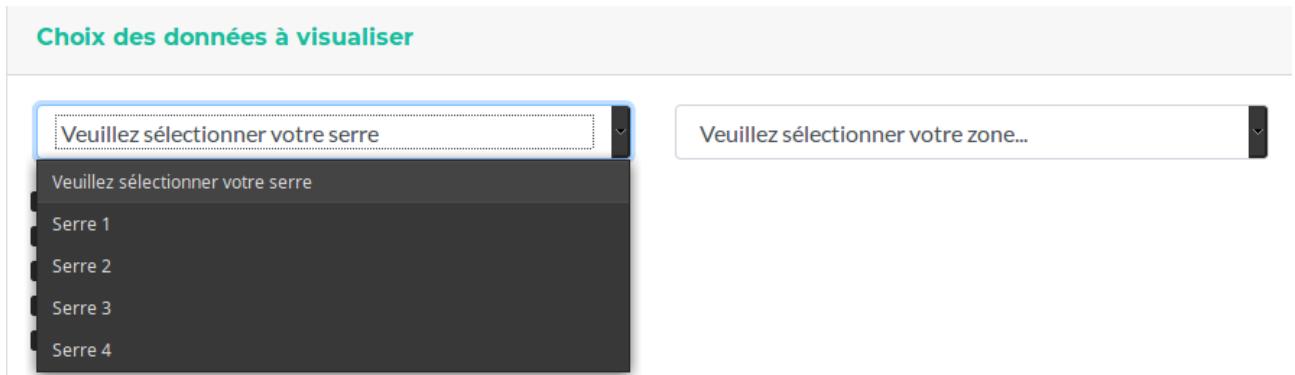
<u>MesuresZone</u>		
Nom du champ	Type du champ	Description
idMesureZone	INT : entier	Identifiant de la mesure de la zone
debitEau	FLOAT : réel	Mesure du débit d'eau
debitEau	FLOAT : réel	Mesure de l'humidité du sol
dateMesure	DATETIME : Date	Date de la mesure
idZone	INT : entier	Identifiant de la zone

Le rôle de la table MesuresSerreInterieur permet de répertorier l'ensemble des mesures tirées des capteurs situés à l'intérieur de la serre.

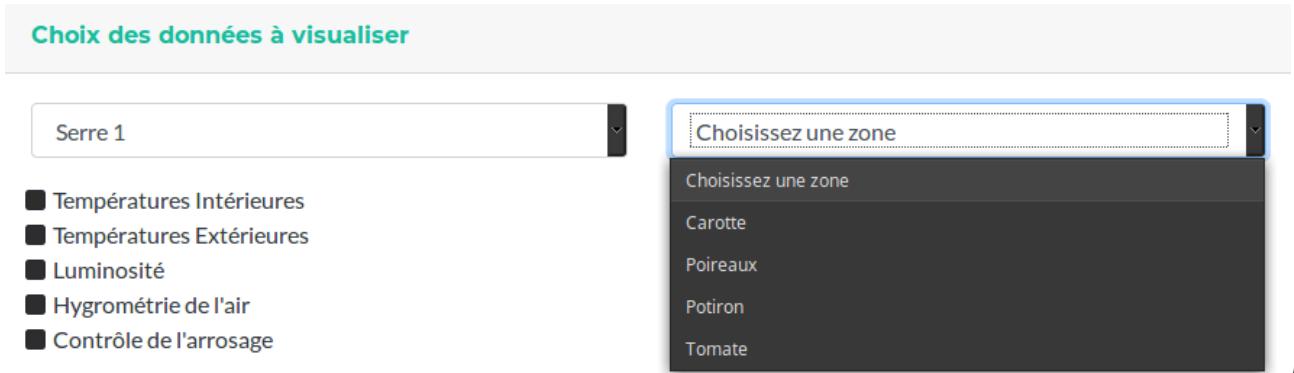
<u>MesuresSerreInterieur</u>		
Nom du champ	Type du champ	Description
idMesureInterieur	INT : entier	Identifiant de la mesure
temperatureInterieur	FLOAT : réel	Valeurs de la température intérieure
hygrometrieInterieur	INT : entier	Valeurs de l'hygrométrie intérieure
luminositeInterieur	INT : entier	Valeurs de la luminosité intérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
idSerre	INT : entier	Identifiant de la serre
idZone	INT : entier	Identifiant de la zone

3. Visualisation des données

Après avoir accédé à la page « Visualiser », l'utilisateur doit sélectionner une serre. Lorsqu'une serre est sélectionnée, la liste déroulante contenant le nom des zones associé à la serre se met à jour.



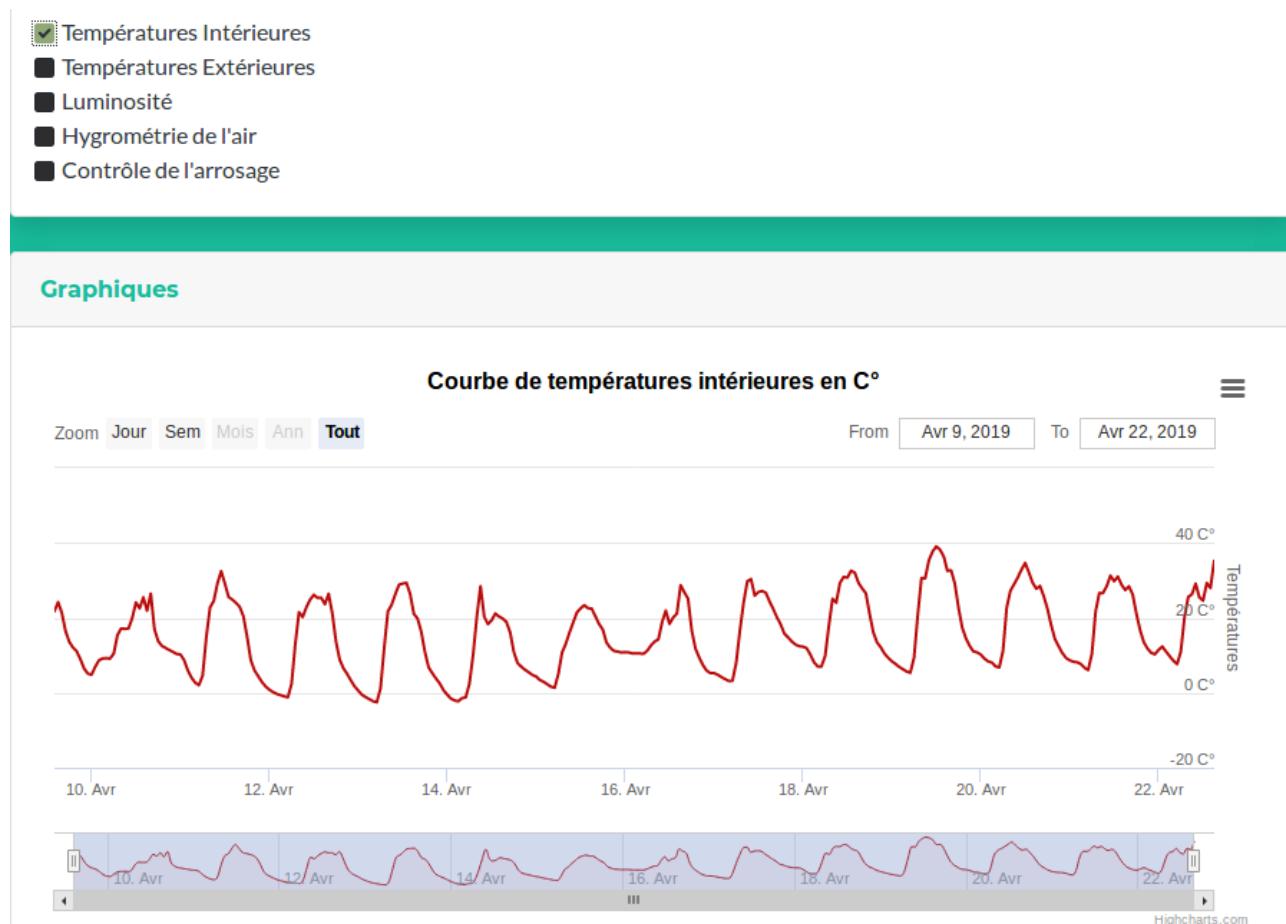
Pas de serre sélectionnée, la liste des zones est vide



La serre « Serre 1 » est sélectionnée, la liste des zones est mise à jour

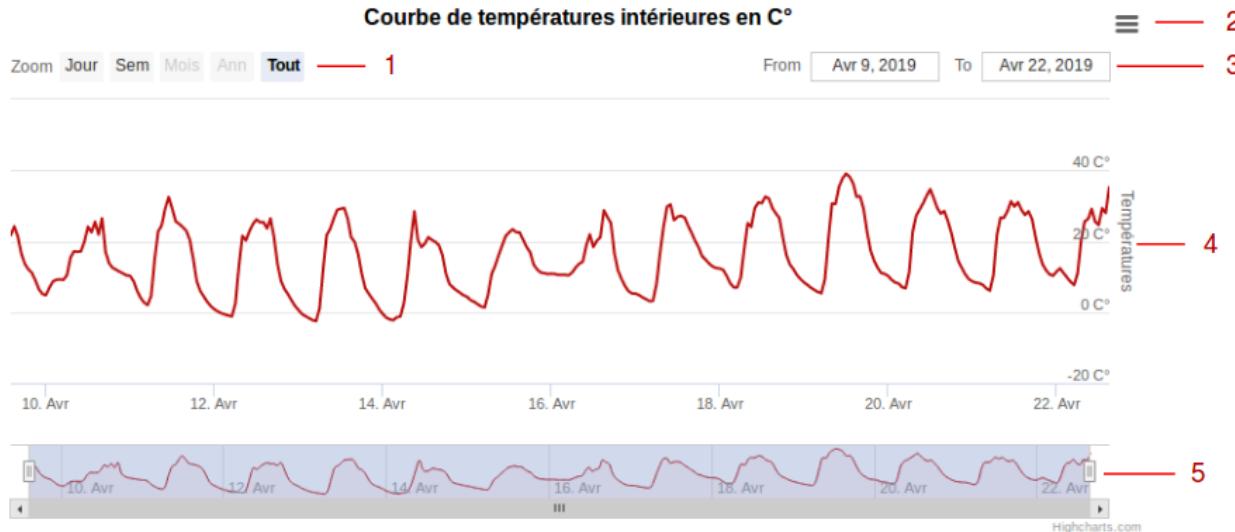
La

Quand le choix de la serre est fait, l'utilisateur peut visualiser les données correspondant à cette serre.



Température intérieure sélectionnée

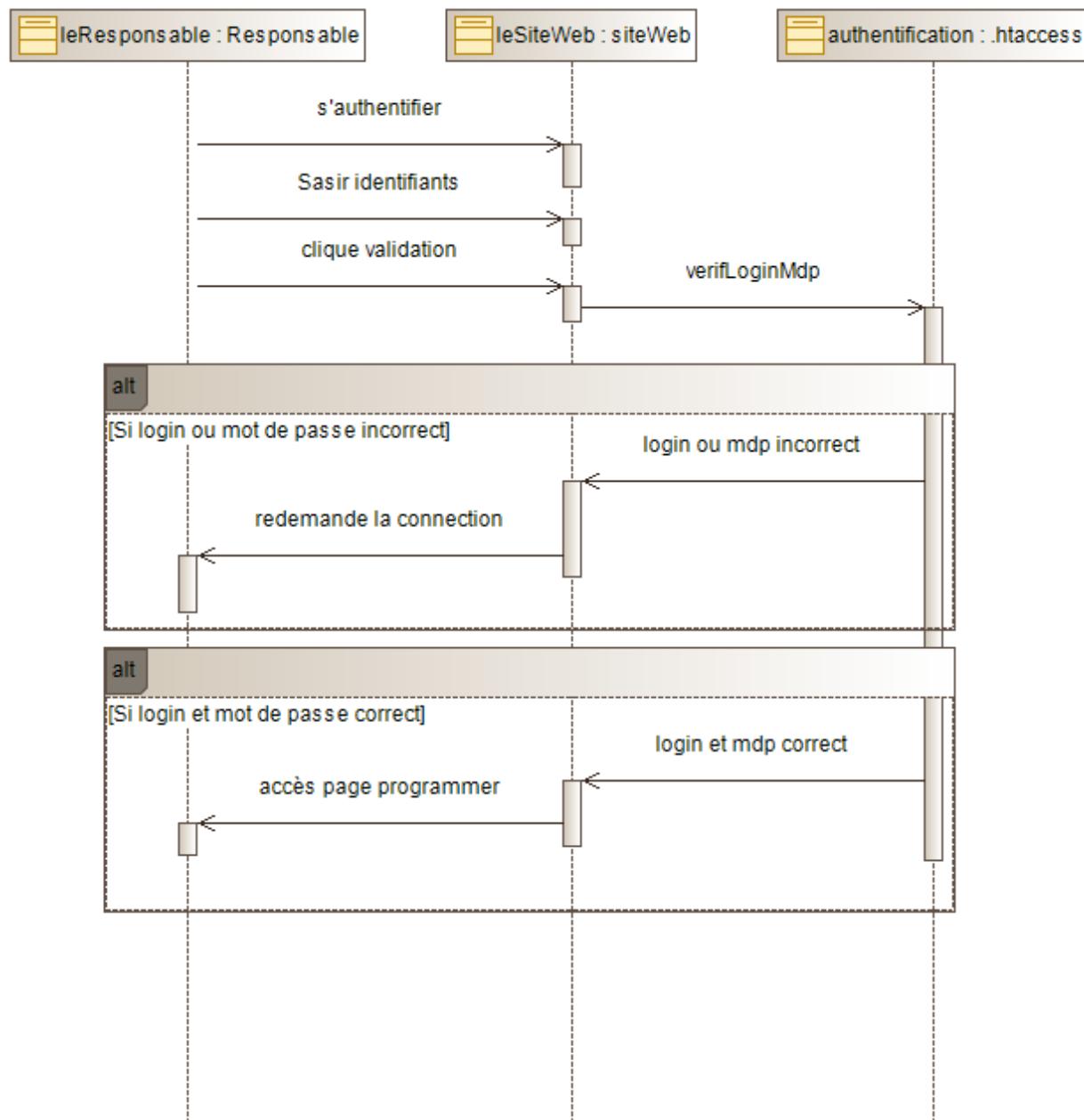
Dans cet exemple, l'utilisateur a sélectionné la température intérieure. Un graphique apparaît, la courbe correspond aux données des capteurs enregistrés dans la base de données. Si l'utilisateur coche une autre grandeur, le nouveau graphique sera positionné sous celui qui est existant. Si l'utilisateur décoche une grandeur, le graphique disparaît de la zone d'affichage.

Graphiques*Détails d'un graphique***Description de l'IHM d'un graphique**

n°1	Sélection de la période de visualisation (par défaut : 1j, 1sem, 1mois, 1an,tout)
n°2	Donne accès à des outils : voir en plein écran le graphique, imprimer le graphique, le télécharger au format '.PNG', '.pdf' ...
n°3	Permet de sélectionner la période de visualisation avec une date précise définie
n°4	Visualisation de la courbe
n°5	Permet de sélectionner la période de visualisation avec le curseur de la souris

4. Tâche n°2 : S'authentifier

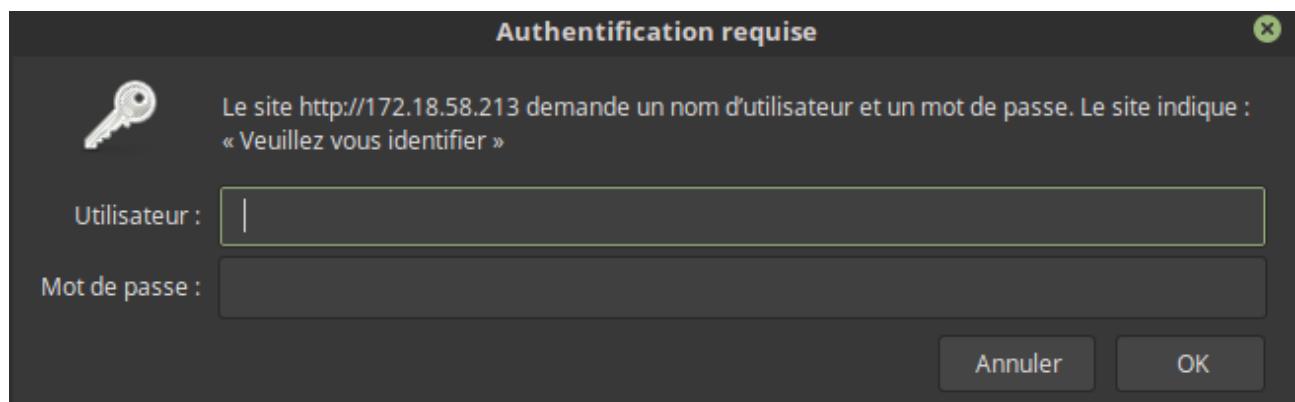
Diagramme de séquence



Pour accéder à la partie « Programmer », l'utilisateur devra s'authentifier. Pour cette partie, nous avons choisi de gérer l'authentification grâce à un fichier « .htaccess ».

(Nous verrons dans la partie Conception détaillée de l'étudiant n°3 Villermin Maxime comment l'htaccess fonctionne)

Quand l'utilisateur clique sur « Programmer », une boîte de dialogue s'ouvre :



Boîte de dialogue de configuration

On voit dans cette boîte de dialogue l'adresse du site qui demande l'authentification (<http://172.18.58.213>) avec le message « Veuillez vous identifier ». Si l'authentification correspond à ce que le fichier attend, on accède à la page « Programmer » sinon la boîte de dialogue redemandera l'authentification.

A screenshot of the 'SERRE CONNECTÉE' web application interface. The top navigation bar includes 'VISUALISER', 'PROGRAMMER', and a notification bell icon showing 16 notifications. The main content area has three cards: 'Configuration Utilisateur' (User Configuration) with a user icon and '+', 'Configuration Météo' (Weather Configuration) with a sun and rain icon, and 'Configuration Arrosage' (Irrigation Configuration) with an irrigation洒水 icon. Each card has a 'Cliquez-ici' (Click here) button at the bottom right.

Page « Programmer »

Si l'utilisateur clique sur « annuler » de la boîte de dialogue alors il sera redirigé vers :

Unauthorized

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.

Apache/2.4.25 (Raspbian) Server at 172.18.58.213 Port 80

Message d'erreur

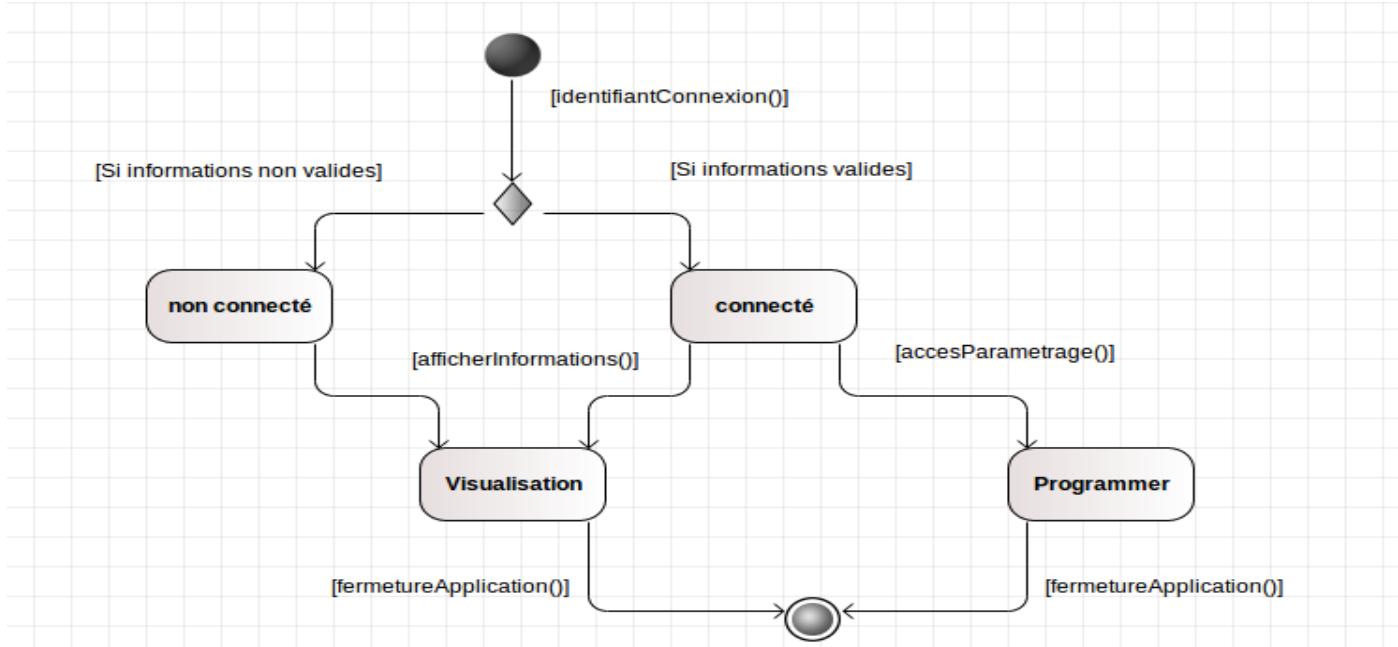
Traduis par :

'Non autorisé'

Ce serveur n'a pas pu vérifier que vous êtes autorisé à accéder au document demandé. Vous avez fourni des informations d'identification incorrectes (mot de passe incorrect, par exemple) ou votre navigateur ne comprend pas comment fournir les informations d'identification requises.'

(Nous verrons aussi que dans la partie Conception détaillée de l'étudiant n°3 que ces pages de redirections sont paramétrables)

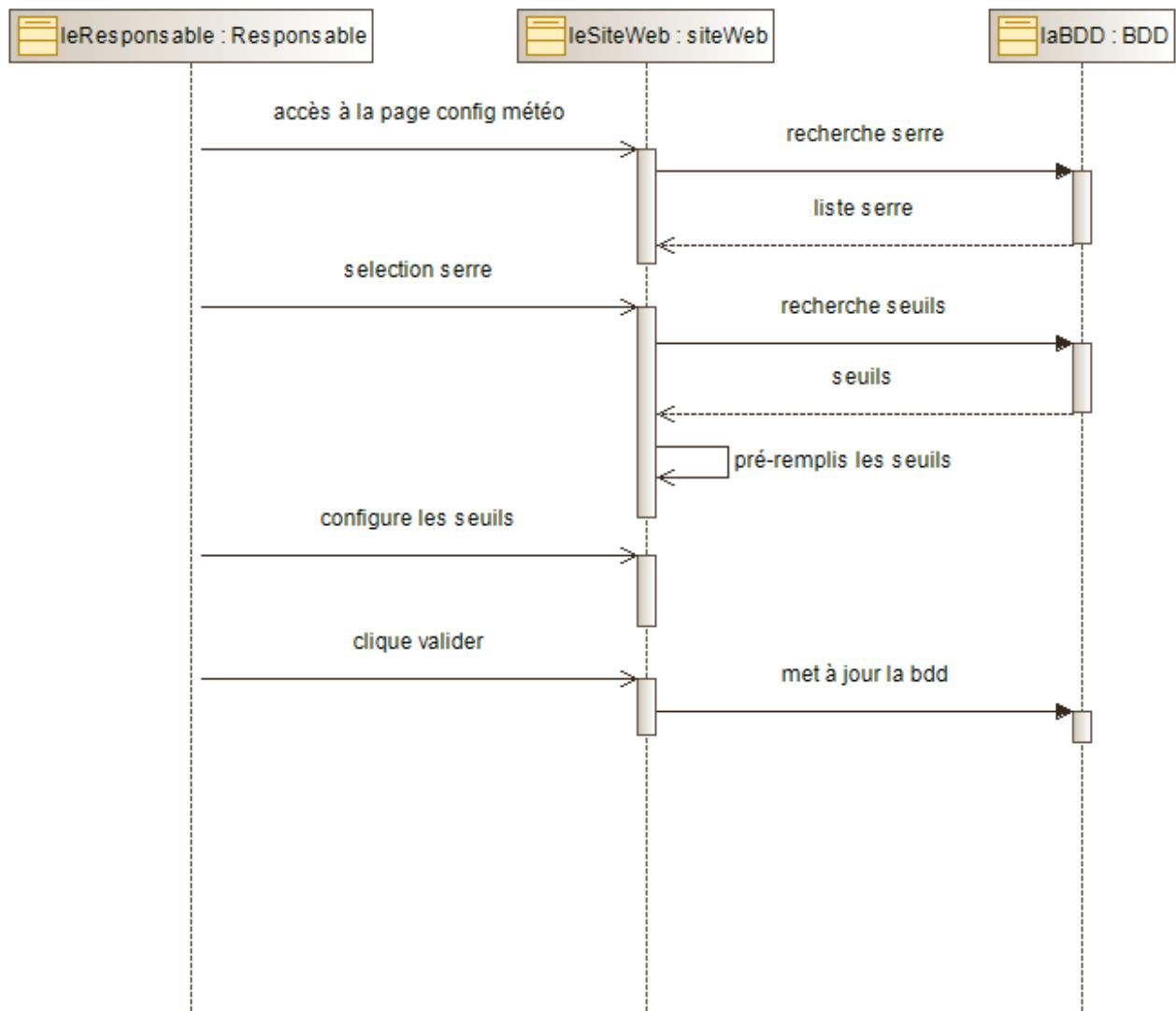
Diagramme d'état-transition « Connexion »



5. Tâche n°3 : Programmer les paramètres d'une serre

Description de cas d'utilisation n°1 : Programmation des seuils d'alertes

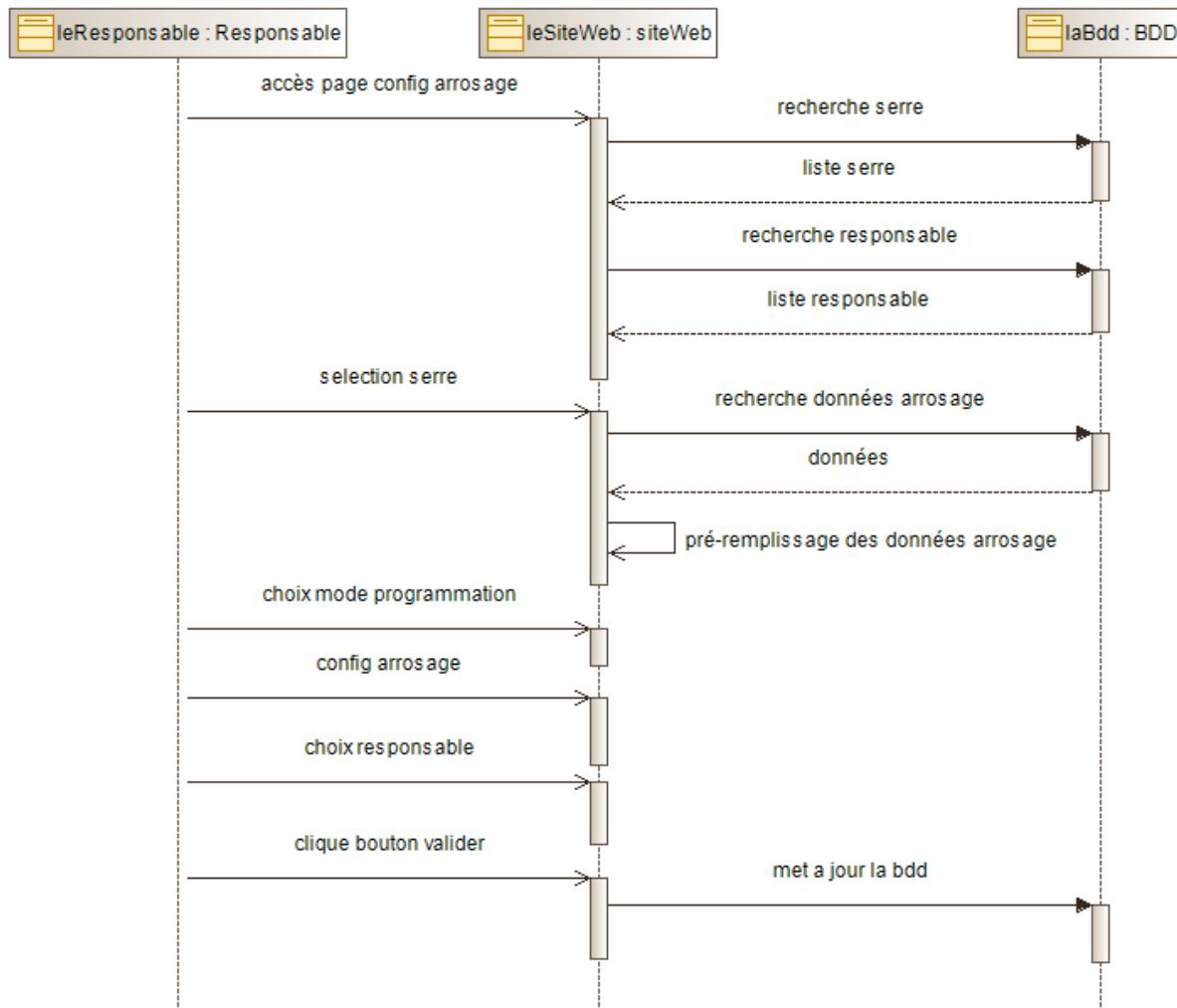
Nom du cas d'utilisation	Programmer les paramètres d'une serre
Utilisateur concerné	Superviseur
Étudiant en charge	Étudiant numéro 3, Maxime VILLERMIN
Préconditions	Le responsable est authentifié. Il est dans la partie «Configuration Météo» de l'onglet «Programmer» du site web.
Scénario nominal	Le responsable sélectionne une serre et configure, depuis cette page, les seuils de luminosité, d'hygrométrie de l'air, la température intérieure de la serre, ainsi que le débit d'eau et l'hygrométrie du sol de chaque zone appartenant à la serre.
Post-condition(s)	Les paramètres de la serre sont envoyés dans la base de données et mis à jour dans le contrôleur de serre.

Diagramme de séquence du cas d'utilisation n°1 : programmation des seuils d'alertes

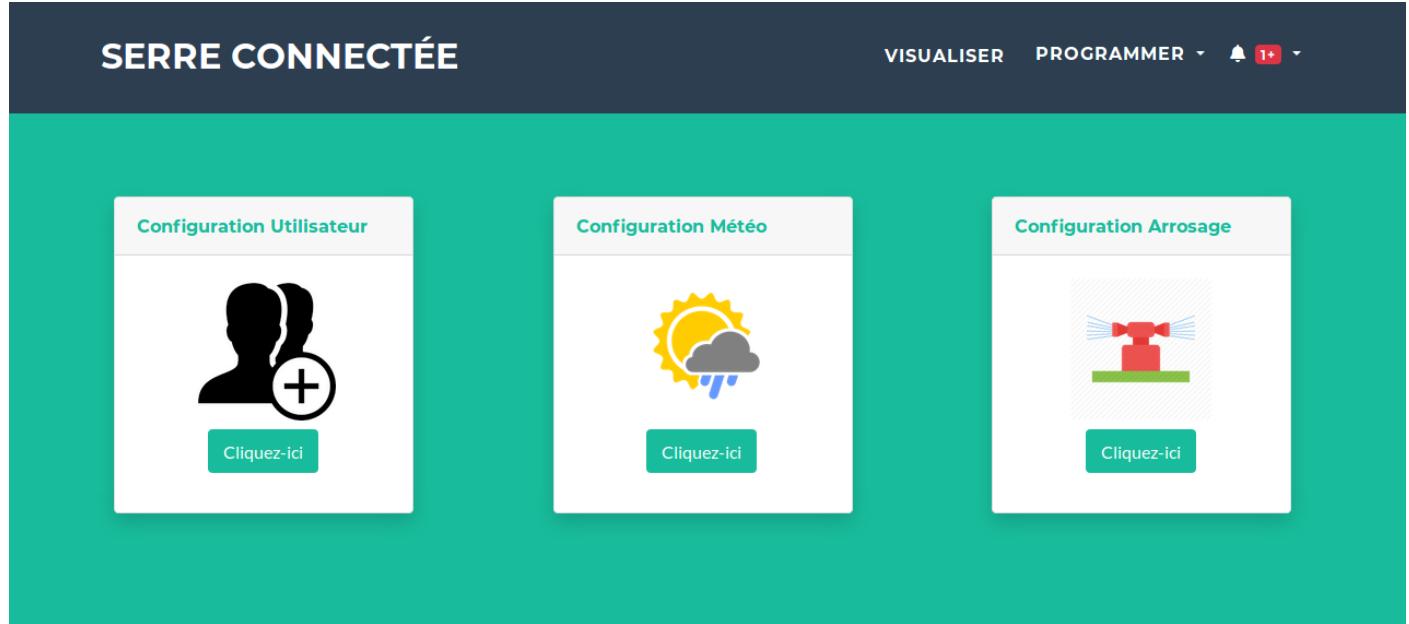
Description de cas d'utilisation n°2 : Programmation de l'arrosage

Nom du cas d'utilisation	Programmer les paramètres d'une serre
Utilisateur concerné	Superviseur
Étudiant en charge	Étudiant numéro 3, Maxime VILLERMIN
Préconditions	Le responsable est authentifié. Il est dans la partie «Configuration Arrosage » de l'onglet « Programmer » du site web.
Scénario nominal	Le responsable sélectionne une serre et configure, depuis cette page, le mode de paramétrage (automatique, manuel ou programmé) et définit la date de début d'arrosage, la périodicité et la durée qu'il souhaite paramétrier. Il sélectionne ensuite le responsable de la serre.
Post-condition(s)	Les paramètres de la serre sont envoyés dans la base de données et mis à jour dans le contrôleur de serre.

Diagramme de séquence du cas d'utilisation n°2 : programmation de l'arrosage



Lorsque l'utilisateur souhaite ajouter un utilisateur, programmer des seuils ou bien programmer l'arrosage, etc., il doit accéder à la page « Programmer ». Pour y accéder, il doit au préalable être connecté. Lorsqu'il est connecté, il arrive sur cette interface :



À partir de là, il peut accéder soit à la page de « configuration d'utilisateur », soit à la « configuration de la météo » ou soit à la « configuration de l'arrosage ». Il peut aussi accéder à ces 3 pages en passant par là :



1. Paramétrage des seuils

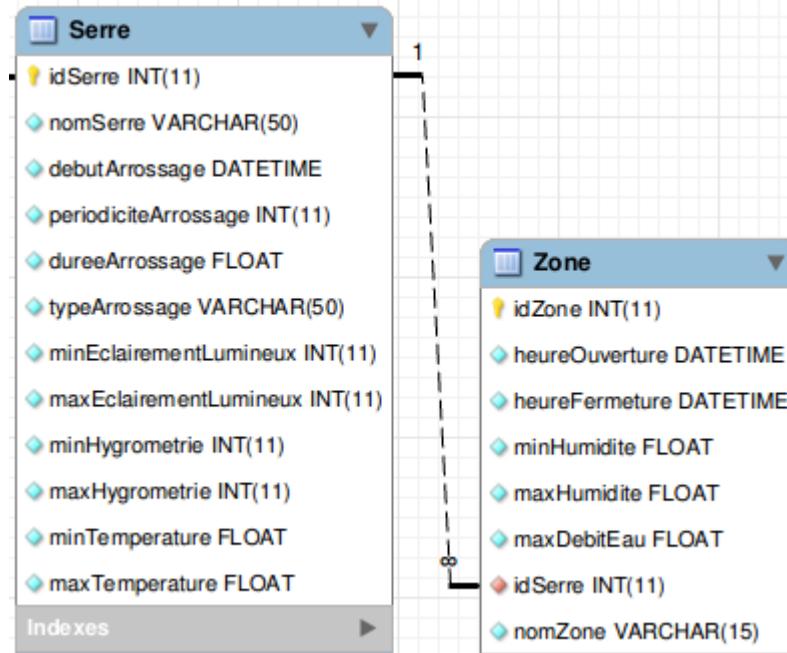
Pour paramétriser les différents seuils, l'utilisateur doit se diriger dans la partie « Programmer » puis dans « Configuration Météo ». Cette partie du projet permet à l'utilisateur (le responsable de serre) de paramétriser les différents seuils d'alertes associés aux capteurs qui composent le système. L'utilisateur peut alors définir : les seuils (minimum/maximum) de, températures intérieures, luminosité, l'hygrométrie de l'air en ce qui concerne la serre et le débit d'eau, hygrométrie du sol.

The screenshot shows the 'SERRE CONNECTÉE' web application. At the top, there are navigation links: 'VISUALISER', 'PROGRAMMER', and a notification icon with a red badge showing '10'. Below the header, the main content area is titled 'Choix de la serre' (Please select your greenhouse). The configuration section is titled 'Configuration de la serre' and contains three sets of sliders for temperature, light, and air humidity. The bottom section is titled 'Configuration de la Météo' and is divided into four columns (Zone 1, Zone 2, Zone 3, Zone 4), each with sliders for water flow and soil humidity thresholds.

Présentation globale de la page « Configuration Météo ».

2. Les tables nécessaires dans la base de données pour paramétriser les seuils

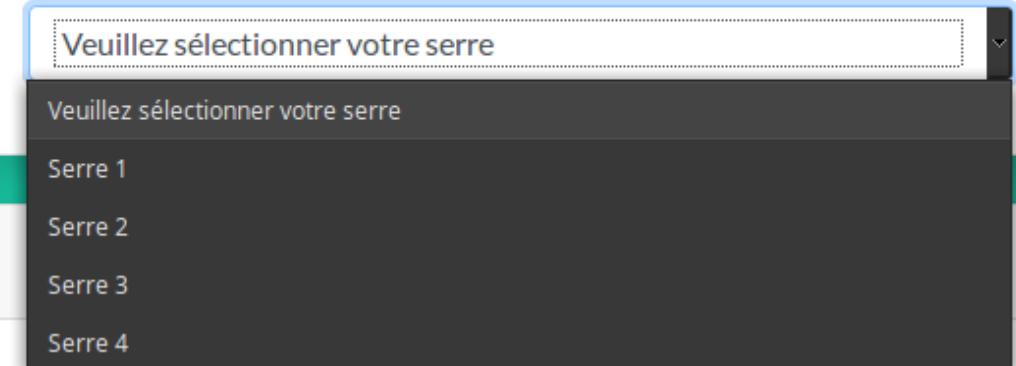
Pour cette partie, nous avons besoin des tables suivantes :



La table Serre sera utilisée pour son IdSerre et le nomSerre qui permettra d'afficher la liste des serres disponibles dans la liste de choix de serre, on utilisera aussi minEclairageLumineux,maxEclairageLumineux, minHygrometrie, maxHygrometrie, minTemperature, maxTemperature. Dans la table Zone, on utilisera minHumidite, maxDebitEau, maxHumidite.

3. Configuration des seuils

Choix de la serre



Veuillez sélectionner votre serre

- Serre 1
- Serre 2
- Serre 3
- Serre 4

Liste de serre : affiche la liste des serres disponibles

Lorsqu'aucune serre n'est sélectionnée, les champs qui permettent à l'utilisateur de remplir les seuils sont vides.

Configuration de la serre

Seuil de température intérieure

Mini : °C Maxi : °C

Seuil de luminosité

Mini : Lux Maxi : Lux

Seuil d'hygrométrie de l'air

Mini : % Maxi : %

Partie configuration de la serre : Aucune serre n'est sélectionnée, les champs sont vides

Configuration de la Météo**Zone 1****Seuil de débit d'eau**Maxi : ▲ ▼**Seuil d'hygrométrie du sol**Mini : ▲ ▼ Maxi : ▲ ▼**Zone 3****Seuil de débit d'eau**Maxi : ▲ ▼**Seuil d'hygrométrie du sol**Mini : ▲ ▼ Maxi : ▲ ▼**Zone 2****Seuil de débit d'eau**Maxi : ▲ ▼**Seuil d'hygrométrie du sol**Mini : ▲ ▼ Maxi : ▲ ▼**Zone 4****Seuil de débit d'eau**Maxi : ▲ ▼**Seuil d'hygrométrie du sol**Mini : ▲ ▼ Maxi : ▲ ▼**Enregistrer**

Partie configuration des zones (qui correspondent à la serre sélectionnée) : Aucune serre n'est sélectionnée, les champs sont vides

Pour configurer les seuils d'une serre, l'utilisateur doit alors choisir une serre.

Choix de la serre**Serre 1**

Une serre est choisie : Serre 1

Lorsque la serre est choisie, s'il en existe dans la BDD, le champ des seuils affiche les seuils existants.

Configuration de la Météo**Zone 1****Seuil de débit d'eau**

Maxi : 30

Seuil d'hygrométrie du sol

Mini : 70

Maxi : 80

Zone 2**Seuil de débit d'eau**

Maxi : 25

Seuil d'hygrométrie du sol

Mini : 80

Maxi : 90

Zone 3**Seuil de débit d'eau**

Maxi : 22

Seuil d'hygrométrie du sol

Mini : 60

Maxi : 70

Zone 4**Seuil de débit d'eau**

Maxi : 28

Seuil d'hygrométrie du sol

Mini : 75

Maxi : 85

Enregistrer

La serre « Serre 1 » est sélectionnée, les seuils des zones correspondantes à la serre sont récupérés et affichés.

Dans la base de données, on a bien 4 zones associées à la serre qui a pour idSerre « 1 » (elle correspond à la serre « Serre 1 »). Si l'on regarde les données insérées dans les champs des zones ci-dessus, ces données correspondent bien avec ceux qui sont dans la BDD.

idZone	he	he	minHumidite	maxHumidite	maxDebitEau	idSerre
1	00	00	70	80	30	1
2	00	00	80	90	25	1
3	00	00	60	70	22	1
4	00	00	75	85	28	1

Que se passe-t-il si aucune donnée (aucun seuil) n'existe dans la BDD ? Les champs restent vides.

Choix de la serre

Serre 2

L'utilisateur sélectionne la serre « Serre 2 »

Configuration de la Météo

Zone 1

Seuil de débit d'eau
Maxi: 12

Seuil d'hygrométrie du sol
Mini: 0 Maxi: 0

Zone 2

Seuil de débit d'eau
Maxi: 10

Seuil d'hygrométrie du sol
Mini: 0 Maxi: 0

Zone 3

Seuil de débit d'eau
Maxi: l/min

Seuil d'hygrométrie du sol
Mini: % Maxi: %

Zone 4

Seuil de débit d'eau
Maxi: l/min

Seuil d'hygrométrie du sol
Mini: % Maxi: %

Enregistrer

Zone 1 et Zone 2 ont des champs préremplis, Zone 3 et Zone 4 sont vides

Les Zones 3 et 4 sont vides, car dans la BDD, il n'y a pas de champs pour ces zones existants.

5 00 00	0	0	12	2 Choux
6 00 00	0	0	10	2 Patate

Toutes les configurations vues précédemment pour les zones dans « Configuration de la météo » seront valables pour la partie « Configuration de la serre », mais ne sont pas encore codées.

4. Paramétrage de l'arrosage

Pour paramétrier des cycles d'arrosage, l'utilisateur doit se diriger dans la partie « Programmer » puis dans « Configuration arrosage ». Cette partie du projet permet à l'utilisateur de paramétrier les arrosages des différentes zones qui composent la serre. Il pourra paramétrier le mode, la date de début, la périodicité et la durée. Il pourra ensuite choisir un responsable de serre qui recevra les alertes en cas de nécessité.

SERRE CONNECTÉE

VISUALISER PROGRAMMER 1

Choix de la serre

Veuillez sélectionner votre serre

Configuration de l'arrosage

Zone 1

Mode de programmation: Choisissez le mode ...

Date de l'arrosage

Début: jj / mm /aaaa Heure Minutes

Périodicité: Heures Durée: Minutes

Zone 2

Mode de programmation: Choisissez le mode ...

Date de l'arrosage

Début: jj / mm /aaaa Heures Minutes

Périodicité: Heures Durée: Minutes

Zone 3

Mode de programmation: Choisissez le mode ...

Date de l'arrosage

Début: jj / mm /aaaa Heures Minutes

Périodicité: Heures Durée: Minutes

Zone 4

Mode de programmation: Choisissez le mode ...

Date de l'arrosage

Début: jj / mm /aaaa Heures Minutes

Périodicité: Heures Durée: Minutes

Choix du responsable de la serre

Responsable:

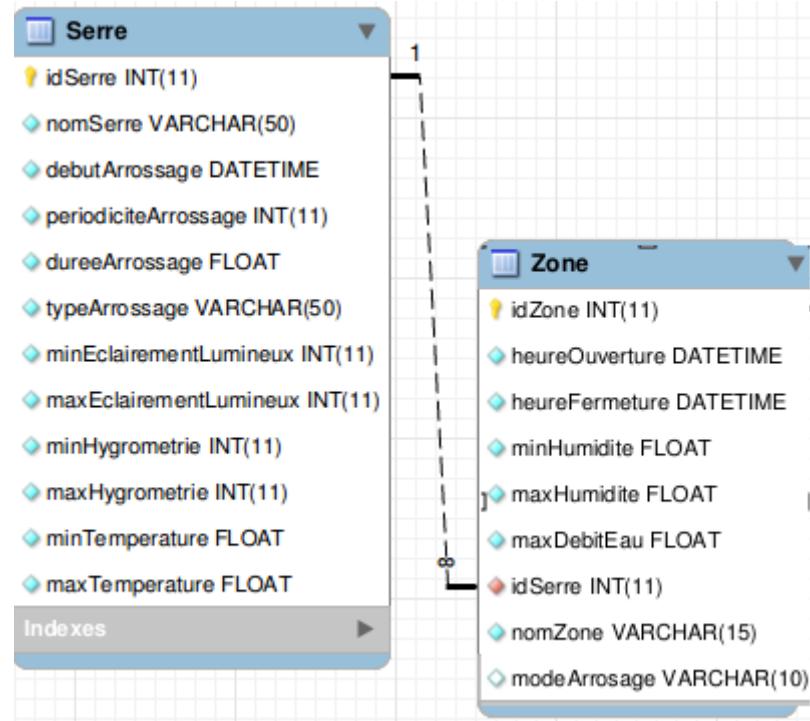
Veuillez sélectionner un responsable...

Valider

Présentation globale de la page « Configuration Arrosage ».

1. Les tables nécessaires pour configurer l'arrosage

Pour cette partie, nous avons besoin des tables suivantes :



La table Serre sera utilisée pour son IdSerre et le nomSerre qui permettra d'afficher la liste des serres disponibles dans la liste de choix de serre. Pour la table Zone on utilisera idZone qui correspondra avec l'idSerre.

2. La configuration de l'arrosage

Choix de la serre

Veuillez sélectionner votre serre

Liste de serre : pas de serre sélectionnée

Choix de la serre

Veuillez sélectionner votre serre

- Serre 1
- Serre 2
- Serre 3
- Serre 4

Liste de serre : affiche la liste des serres disponibles

Configuration de l'arrosage

Zone 1

Mode de programmation : Choisissez le mode ...

Date de l'arrosage

Début : jj / mm / aaaa Heure Minutes
Périodicité : Heures Durée : Minutes

Zone 2

Mode de programmation : Choisissez le mode ...

Date de l'arrosage

Début : jj / mm / aaaa Heures Minutes
Périodicité : Heures Durée : Minutes

Zone 3

Mode de programmation : Choisissez le mode ...

Date de l'arrosage

Début : jj / mm / aaaa Heures Minutes
Périodicité : Heures Durée : Minutes

Zone 4

Mode de programmation : Choisissez le mode ...

Date de l'arrosage

Début : jj / mm / aaaa Heures Minutes
Périodicité : Heures Durée : Minutes

Lorsqu'aucune serre n'est sélectionnée, les champs qui permettent à l'utilisateur de paramétrier l'arrosage sont vides. L'utilisateur peut paramétrier chaque zone indépendamment.

Comme pour la partie « Configuration Météo » vue précédemment, lors de la sélection de la serre, si les paramètres existent dans la base, les champs de configuration de l'arrosage seront préremplis. L'utilisateur a le choix entre 3 modes de programmation d'arrosage : Mode automatique, mode manuel, mode programmé.

- En mode automatique :

La serre prend en paramètre les seuils configurés dans « Configuration météo ». Par exemple : si l'humidité dépasse le seuil d'hygrométrie du sol minimum, l'arrosage sera déclenché jusqu'à ce que l'hygrométrie du sol remonte.

- En mode manuel :

C'est le responsable qui gère l'arrosage manuellement avec son smartphone (il ouvre ou ferme les vannes).

- En mode programmé :

Le responsable doit alors configurer la date de début d'arrosage, la périodicité, et la durée de chaque zone.

Comme chaque zone est indépendante, il peut mixer les configurations des zones de la serre.

Zone 1

Mode de programmation : Mode automatique

Date de l'arrosage

Début : jj / mm /aaaa Heure Minutes

Périodicité : Heures Durée : Minutes

Zone 2

Mode de programmation : Mode programmé

Date de l'arrosage

Début : jj / mm /aaaa Heures Minutes

Périodicité : Heures Durée : Minutes

À gauche mode automatique : champs grisés / à droite mode programmé : champs configurables

En mode manuel et automatique, les champs de configurations seront grisés, car inutiles dans le cas de ces deux modes.

Une fois que l'utilisateur a paramétré l'arrosage comme il le souhaite, il doit sélectionner un responsable de serre.

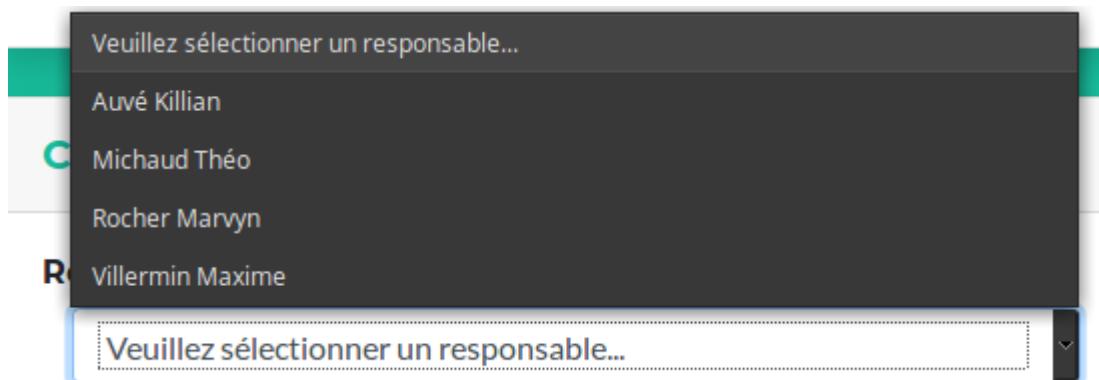
Choix du responsable de la serre**Responsable :**

Veuillez sélectionner un responsable...

Valider

Choix du responsable de la serre

La liste des responsables est préremplie d'utilisateur contenu dans la base de données. Lorsqu'il clique sur la liste déroulante, il peut alors en choisir un.

*Liste déroulante des responsables*

Quand le responsable est choisi, les informations le concernant s'affichent à l'écran (Nom, Prénom, Numéro de téléphone et Adresse mail) ainsi qu'un bouton « modifier ».

Responsable :

Villermin Maxime

Informations du Responsable :

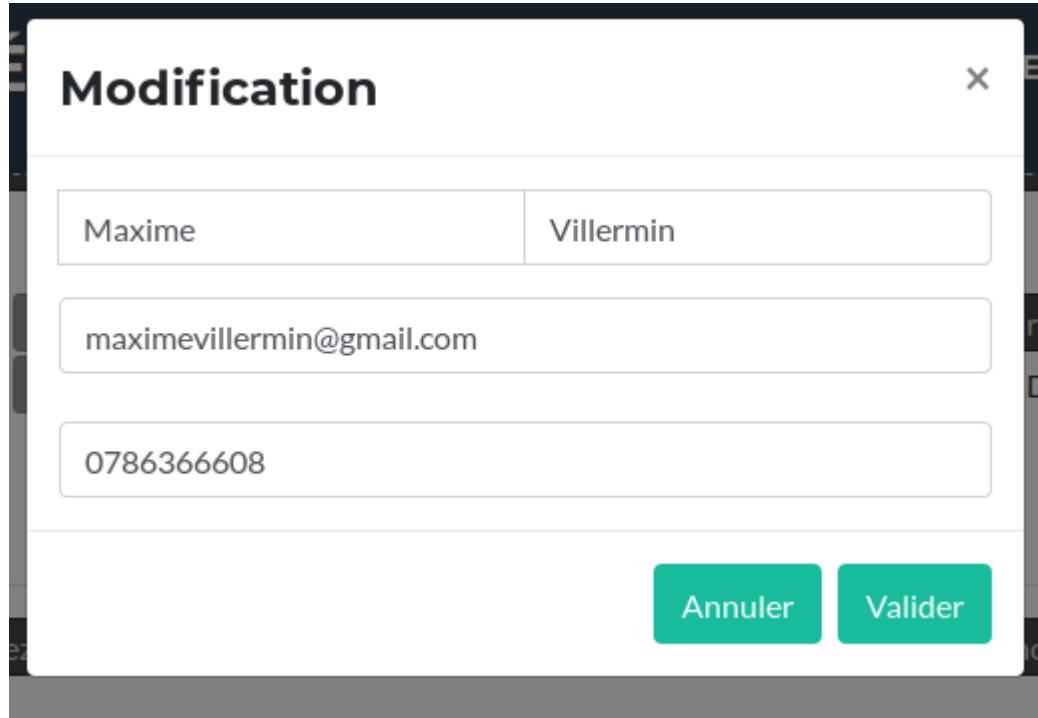
Adresse mail : maximevillermin@gmail.com

Numéro de téléphone : 0786366608

Modifier

Le responsable « VILLERMIN Maxime » a été choisi, on affiche ses informations

Si le responsable souhaite modifier un responsable (pour modifier son numéro de téléphone par exemple), il clique sur le bouton « modifier ». Alors s'affiche une boîte de dialogue de modification.



Modal de modification de responsable

Les champs de la boîte de dialogue (modal) sont préremplis pour éviter à l'utilisateur de ressaisir toutes les informations concernant le responsable et de modifier que ce qu'il faut réellement modifier. Deux boutons se situent sous les différents champs : « annuler » qui annule la modification et donc n'enregistre pas la modification, et un bouton « valider » qui lui va mettre à jour l'utilisateur et ses informations dans la base de données.

3.Configuration d'un responsable

Pour configurer un responsable, l'utilisateur doit se diriger dans la partie « Programmer » puis dans « Configuration utilisateur ». Cette partie du projet permet à l'utilisateur de paramétriser un nouveau responsable, de le modifier, de le supprimer et d'avoir la liste des utilisateurs dans un tableau. Il devra renseigner le nom, prénom, e-mail, numéro de téléphone du responsable.

The screenshot shows the 'SERRE CONNECTÉE' application interface. At the top, there are navigation tabs: 'VISUALISER', 'PROGRAMMER', and a notification icon with '1+' notifications. The main area has two sections: 'Ajouter un nouvel utilisateur' (Add a new user) and 'Liste des responsables' (List of responsible persons). The 'Ajouter un nouvel utilisateur' section contains four input fields: 'Nom' (Name), 'Prénom' (First name), 'Email', and 'Numéro de téléphone' (Phone number), followed by a green 'OK' button. The 'Liste des responsables' section displays a table with five rows of data:

Prénom	Nom	Email	Numéro de téléphone
Killian	Auvé	killianAuve@gmail.com	0604050607
Théo	Michaud	theomichaud@gmail.com	0606060606
Marvyn	Rocher	marvynrocher@gmail.com	0601010105
Maxime	Villermin	maximevillermin@gmail.com	0786366607

Below the table are two buttons: 'Modifier' (Modify) and 'Supprimer' (Delete).

Présentation globale de la page « Configuration Utilisateur ».

1. Les tables nécessaires pour configurer un responsable

Utilisateur	
idUtilisateur	INT(11)
nom	VARCHAR(255)
prenom	VARCHAR(255)
numTel	VARCHAR(10)
mail	VARCHAR(255)
Indexes	

Serre	
idSerre	INT(11)
nomSerre	VARCHAR(50)
debutArrossage	DATETIME
periodiciteArrossage	INT(11)
dureeArrossage	FLOAT
typeArrossage	VARCHAR(50)
minEclairementLumineux	INT(11)
maxEclairementLumineux	INT(11)
minHygrometrie	INT(11)
maxHygrometrie	INT(11)
minTemperature	FLOAT
maxTemperature	FLOAT
Indexes	

La table Utilisateur sera la table principale de cette partie. Elle est utilisée pour son IdUtilisateur, le nom, prénom, numTel, mail qui correspond aux données d'un utilisateur. La table Serre sera utilisée pour son idSerre pour associer un utilisateur à une serre.

2. Ajout d'un nouvel utilisateur

Ajouter un nouvel utilisateur

Nom
Prénom
Email
Numéro de téléphone
OK

Ajout d'un utilisateur

Lors de l'ajout d'un utilisateur, les différents champs sont vides. Il faut donc les remplir. On saisit alors le nom, le prénom, l'e-mail et le numéro de téléphone. Le nom et prénom sont de type 'texte', l'e-mail de type 'mail' et le numéro de téléphone de type 'tel'.

(Nous verrons dans la partie Conception détaillée de l'étudiant n°3 Villermin Maxime ce qui différencie les types de champs et les différents paramètres qu'on leur a attribués)

Ajouter un nouvel utilisateur

Nom

Veuillez compléter ce champ.

Email

Numéro de téléphone

OK

Ajout d'un utilisateur lorsque l'utilisateur clique sur 'OK', mais que les champs sont vides

On peut voir que si l'utilisateur ne remplit pas tous les champs, une bordure rouge entoure le champ et un message indique 'Veuillez compléter ce champ'.

Liste des responsables

Prénom	Nom	Email	Numéro de téléphone
Killian	Auvé	killianAuve@gmail.com	0604050607
Théo	Michaud	theomichaud@gmail.com	0606060606
Marvyn	Rocher	marvynrocher@gmail.com	0601010105
Maxime	Villermin	maximevillermin@gmail.com	0786366607

Modifier

Supprimer

Liste des utilisateurs

idUtilisateur	nom	prenom	numTel	mail
6	Villermin	Maxime	0786366607	maximevillermin@gmail.com
8	Michaud	Théo	0606060606	theomichaud@gmail.com
10	Rocher	Marvyn	0601010105	marvynrocher@gmail.com
11	Auvé	Killian	0604050607	killianAuve@gmail.com

Liste des utilisateurs contenus dans la base de données

Les utilisateurs enregistrés dans la base sont bien affichés dans le tableau des utilisateurs.

(Les boutons que nous allons aborder ne sont pas encore codés)

Le bouton modifier reprend le principe d'ouverture d'un modal (*comme dans la partie « Configurer l'arrosage »*) lors du clique, avec les champs préremplis pour une modification plus rapide. Le bouton supprimer servira à supprimer la ligne du tableau qui aura été cocher par un ‘checkButton’ (case à cocher).



Exemple d'Alertes

Lorsqu'un seuil est dépassé (*voir réalisation partie personnelle de l'étudiant 4 : ROCHER Maryn*), une notification est envoyée. Cette notification, l'alerte, contient la date et l'heure d'émission de l'alerte ainsi qu'un message détaillant la nature de l'incident. Un checkbox sera disponible pour confirmer la lecture de l'alerte.

ALERTE

26 FÉVRIER 2019

 **L'HUMIDITÉ DU SOL EST FAIBLE !
VEUILLEZ ARROSER LE SOL**

[VOIR TOUTES LES ALERTES](#)

IHM de l'alerte

Cette IHM est non fonctionnelle, rien n'a encore été codé.

Quand elle le sera, le responsable sera en mesure de visualer les alertes depuis son bureau au cas où il n'aurait pas son téléphone proche de lui. La confirmation de lecture serait utile pour éviter que l'alerte soit renvoyée pour le même incident dans un temps proche (éviter de renvoyer l'alerte toutes les 15 min).

IV/- Conception préliminaire : Étudiant 4, ROCHER Marvyn

1. Préambule

Conformément au cahier des charges, mais aussi au diagramme de classe, l'étudiant n°4 a été affecté aux tâches :

- ✗Contrôler l'état d'une serre.
- ✗Envoyer une alerte (SMS ou mail) au responsable en cas de dépassement de seuil.

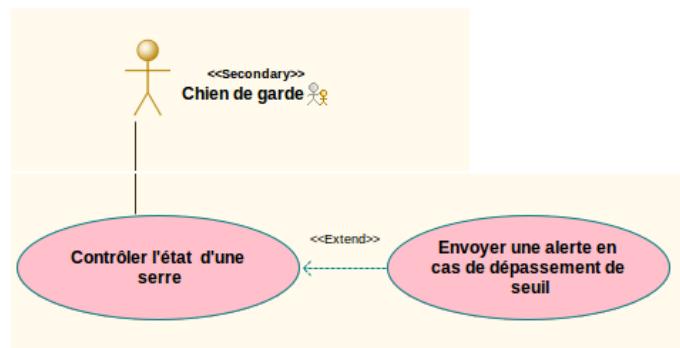


Diagramme des cas d'utilisation : étudiant n°4

2. Les outils de développement

Pour pouvoir faire mes tâches, j'ai dû utiliser plusieurs outils de développement.

Voici les logiciels que j'ai utilisés :

- Modelio : outil de modélisation UML. Il m'a permis de faire mes diagrammes de séquences et d'états-transitions.



- Github : service web d'hébergement et de gestion de développement de logiciels.

Il m'a permis notamment d'héberger mes fichiers en ligne et pouvoir récupérer d'anciennes versions si nécessaire (back-up).



Pour les langages, j'ai utilisé :

- Bash Linux : langage de programmation sous Linux



- SQL : est un langage informatique normalisé servant à exploiter des bases de données relationnelles



3. Les installations diverses

Dans la liste des utilitaires à installer sur notre serveur web, qui est un Raspberry PI 3, je devais particulièrement m'attarder sur l'installation de MariaDB (base de données choisie dans le cahier des charges), mais également PhpMyAdmin, Apache, car la base de données a besoin d'un serveur HTTP sur son serveur d'installation.

1. Apache

Nous devions choisir un serveur HTTP afin de mener à bien notre projet, nous avons donc opté pour Apache qui est le plus connu. Afin de pouvoir l'installer sur le Raspberry PI 3, j'ai effectué toutes les installations des utilitaires en ligne de commande.

```
pi@raspberrypi:~ $ sudo apt-get install apache2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
apache2 is already the newest version (2.4.25-3+deb9u6).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

La base de données a besoin d'un serveur HTTP sur son serveur d'installation

2. MariaDB

MariaDB étant un fork de MySQL, nous l'avons choisi afin de créer notre base de données. En adéquation donc avec le cahier des charges.

```
pi@raspberrypi:~ $ sudo apt-get install mariadb-server
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
mariadb-server is already the newest version (10.1.37-0+deb9u1).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

La base de données choisie dans le cahier des charges étant MariaDB

Après avoir installé MariaDB, il est nécessaire de redémarrer le service afin de pouvoir compléter l'installation et voir si celle-ci est bien installée.

```
root@raspberrypi:/etc/mysql/mariadb.conf.d# service mariadb restart
```

On redémarre le service MariaDB

3. PhpMyAdmin

Toujours dans le respect de notre projet, il a été nécessaire de penser à installer PhpMyAdmin afin de pouvoir gérer la base de données sur un ordinateur par interface graphique.

```
pi@raspberrypi:~ $ sudo apt-get install phpmyadmin
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
phpmyadmin est déjà la version la plus récente (4:4.6.6-4).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

PhpMyAdmin permet de gérer la base de données via interface graphique

4. Configuration générale

Dans la configuration générale que j'ai dû effectuer il y a en d'autres : la création de la base de données, la création des utilisateurs, l'affectation des droits aux utilisateurs, mais aussi permettre de pouvoir accéder à la base de données à distance.

1. Crédation de la base de données

La base de données créée se nomme « laSerre », elle y répertorie l'ensemble des données nécessaires au bon fonctionnement du projet : les données des capteurs, les informations des utilisateurs, les informations concernant les seuils.

```
MariaDB [(none)]> create database laSerre;
Query OK, 1 row affected (0.00 sec)
```

Commande afin de créer la base de données "laSerre"

2. Crédation des utilisateurs

Afin de pouvoir gérer la base de données, il a été nécessaire de créer divers utilisateurs, dont un qui possède l'ensemble des droits sur la base de données pour pouvoir la gérer entièrement. Nous avons donc opté également pour créer un autre utilisateur qui n'a accès qu'aux tables de la base « laSerre ».

```
MariaDB [(none)]> create user 'serreUtilisateur'@'%' identified by 'Touchard72';
Query OK, 0 rows affected (0.00 sec)
```

Création du super Utilisateur : serreUtilisateur

```
MariaDB [(none)]> select user from mysql.user;
+-----+
| user |
+-----+
| serre |
| serreUtilisateur |
| phpmyadmin |
| root |
+-----+
4 rows in set (0.00 sec)
```

Vérification de la création de l'utilisateur "serreUtilisateur"

3. Affectation des droits

Dès la création des utilisateurs, il a fallu administrer leurs droits respectifs ; ainsi nous avons choisi que le super Utilisateur comme son nom l'indique ait tous les droits, le second utilisateur n'aura pas le droit que d'accéder à la table **Serre**.

```
MariaDB [(none)]> grant all privileges on laSerre.* to 'serreUtilisateur'@'%';
Query OK, 0 rows affected (0.01 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Affectation des droits au super Utilisateur "serreUtilisateur"

```
MariaDB [(none)]> show grants for 'serreUtilisateur'@'%';
+-----+
| Grants for serreUtilisateur@% |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'serreUtilisateur'@'%' IDENTIFIED BY PASSWORD '*7AA6C59F59CD0CE364F10B272F25D5E70DF15A9B' |
| GRANT ALL PRIVILEGES ON `laSerre`.* TO 'serreUtilisateur'@'%' |
+-----+
2 rows in set (0.00 sec)
```

Affichage des droits pour le super Utilisateur

4. Accès à la base de données en distant

Afin de compléter la configuration générale du Raspberry PI3, il est nécessaire de pouvoir accéder à distance à la base de données dans le cadre de notre projet. Afin que les utilisateurs puissent se connecter au serveur depuis une machine distante.

```
root@172.18.58.213's password:

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

root@raspberrypi:/etc/mysql/mariadb.conf.d# nano 50-server.cnf
```

Il est nécessaire de modifier le fichier "50-server.cnf" afin de pouvoir accéder à la base de données en distant

5. Gammu

Afin de mieux décrire le second cas d'utilisation auquel j'ai été affecté, il a été nécessaire d'installer Gammu, effectivement le second cas étant l'envoi d'une alerte en cas de dépassement de seuil. Cela passe principalement par le modem GSM fourni dès le début du projet.

Gammu étant un utilitaire permettant d'envoyer des SMS via Raspberry PI, en ligne de commande il a été judicieux de le mettre en avant, mais également de s'en servir afin de répondre à une exigence du cahier des charges. Afin de l'installer, j'ai dû l'installer par ligne de commande.

```
sudo apt-get install gammu
```

Installation de l'utilitaire Gammu permettant l'envoi de SMS via Raspberry PI équipé d'un modem GSM

6. Serveur mail

Dans le cadre de notre projet, il est indiqué que le responsable sera prévenu en cas de dépassement de seuils, en cas d'incidents, par courriel envoyé de manière automatique. Dans cette démarche, il est nécessaire de savoir quelle technologie nous allons utiliser dans le cadre de l'envoi de courriels.

Pour pouvoir envoyer des courriels, nous devons donc créer un serveur mail propre à nos besoins : nous allons utiliser PostFix sur le Raspberry.

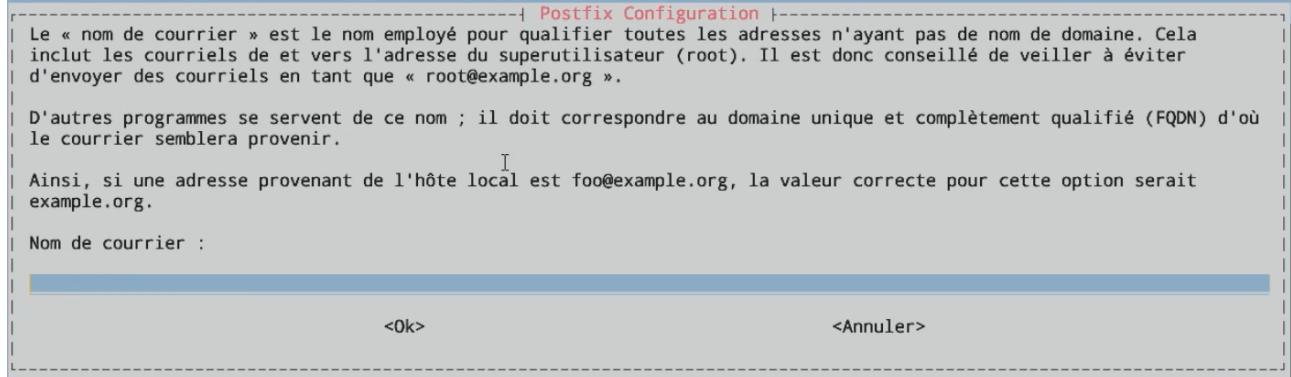
PostFix étant un utilitaire permettant d'envoyer des e-mails via Raspberry PI, en ligne de commande il a été judicieux de le mettre en avant, mais également de s'en servir afin de répondre à une exigence du cahier des charges. Afin de l'installer, j'ai dû l'installer par ligne de commande.

```
sudo apt-get install postfix
```

Installation de postfix

Une fois installé, il a fallu que je sache comment configurer PostFix afin de pouvoir envoyer des emails.

Afin de configurer PostFix il a fallu que je choisisse un nom de courrier, que j'édite le fichier de configuration de PostFix afin d'enlever le nom de courrier sur la ligne destination ; que je puisse configurer également le fait que le serveur puisse écouter en « local ».



Insertion du nom de courrier

sudo vim / etc / postfix / main.cf

Édition du fichier de configuration de postfix

```
+ /e/p/main.cf
# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = debian
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = debian, localhost.localdomain, localhost
relayhost =
mynetworks = [127.0.0.0/8 ::ffff:127.0.0.0]/104 [::1]/128
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
```

On enlève sur la ligne destination le nom de courrier

```
+ /e/p/main.cf
# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = debian
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = debian, localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = loopback-only
/etc/postfix/main.cf[+] CWD: /home/dev Line: 41
```

Afin que le serveur écoute en local, on supprime sur la ligne inet_interfaces; le "all" en le remplaçant par "loop-back"

Une fois cette configuration terminée, il a fallu redémarrer le service PostFix, et installer les utilitaires de mail afin de pouvoir en envoyer.

sudo service restart postfix

Redémarrage de postfix avec les changements de paramètres effectués.

7. Tâche n°1 : Contrôler l'état d'une serre

Dans le cadre de la supervision de la serre, il est nécessaire de pouvoir la surveiller et donc de la contrôler. Ainsi, l'étudiant n°4 étant en charge de cette partie du projet, il sera détaillé dans ce dossier les points sur lesquels il a été affecté, comment a-t-il réalisé sa partie personnelle tout en corrélation avec les autres membres du projet afin d'imbriquer sa partie dans le projet commun. Le contrôle de la serre se fera de manière automatique toutes les 15 minutes, c'est-à-dire que toutes les 15 minutes, de manière automatique, le Raspberry PI interrogera la base de données et lira les seuils et les dernières valeurs relevées afin de pouvoir les comparer.

Description du cas d'utilisation

Nom du cas d'utilisation	Contrôler l'état d'une serre
Utilisateur concerné	Chien de garde
Étudiant en charge	Étudiant numéro 4, Marvyn ROCHER
Préconditions	Les informations relevées des différents capteurs ont été relevées, et sont présentes dans la base de données
Scénario nominal	Toutes les 15 minutes, les données présentes dans la base de données sont comparées aux seuils qui ont été fixés (en termes d'hygrométrie, de température , de luminosité pour la Serre, en termes d'humidité et de débit d'eau pour chaque zone présente dans la Serre)
Post-condition(s)	L'état de la serre a été contrôlé

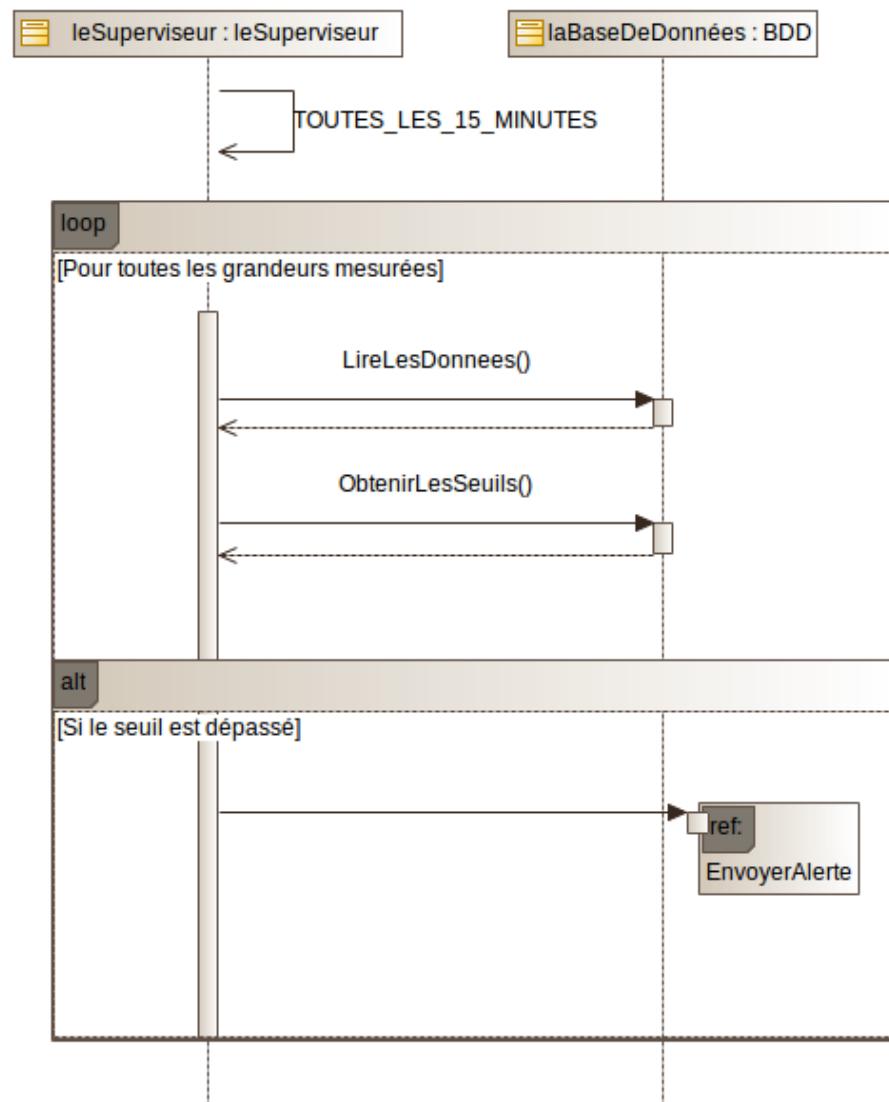
Diagramme de séquence

Diagramme de séquence du cas d'utilisation n°1 :
Contrôler l'état d'une serre

8. Tâche n°2 : Envoyer une alerte en cas de dépassement de seuil

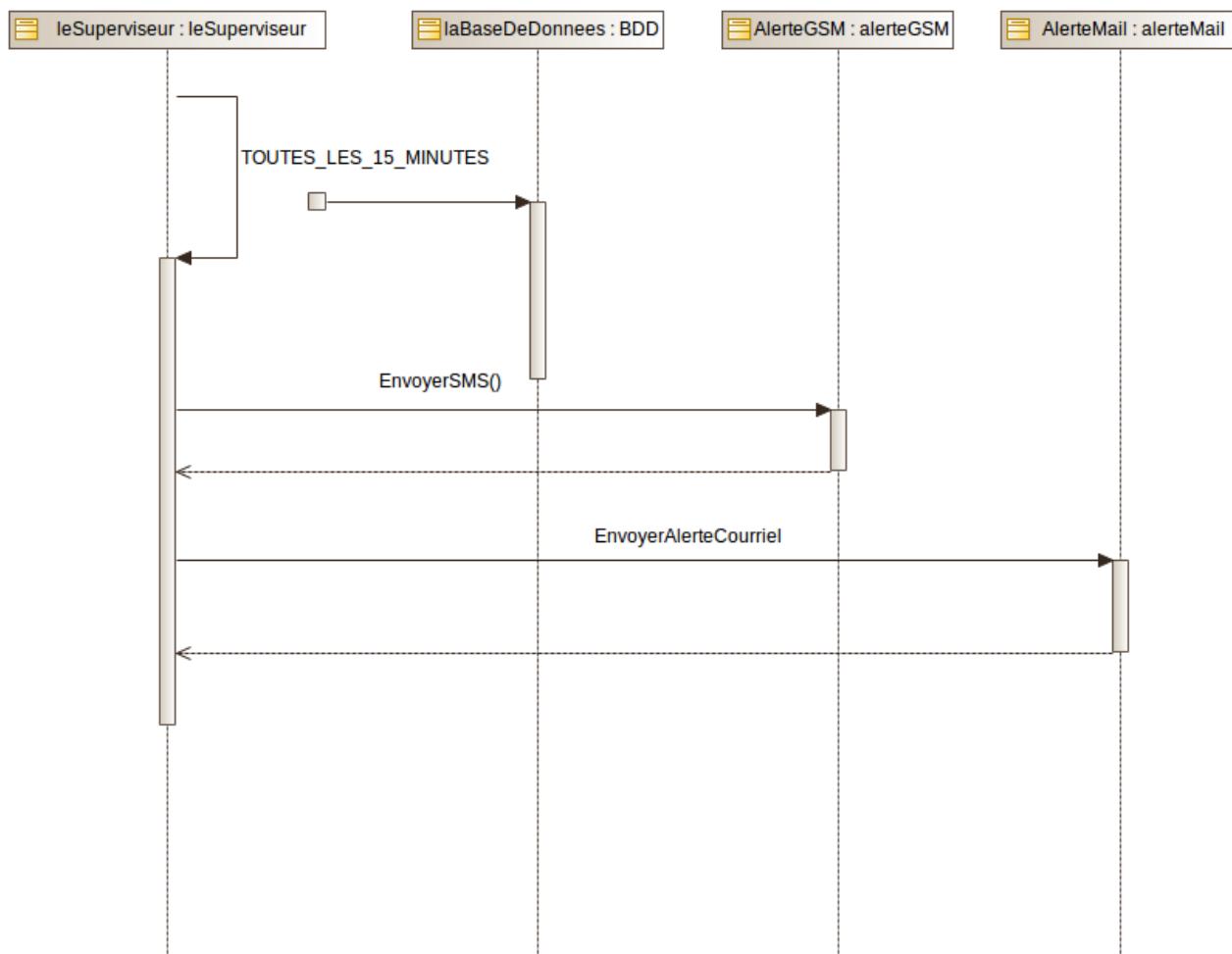
Afin de compléter la surveillance de la serre, il sera également détaillé au cœur de ce dossier le complément de la surveillance par l'envoi d'alertes sous diverses formes : SMS/courriel.

Ces alertes seront alors envoyées de manière autonome : les valeurs relevées, à partir des capteurs présents dans la serre, et présentes dans la base de données seront comparées avec les seuils qui, eux, ont été établis sur le site Internet dédié, ou bien sur la base de données directement.

Le type d'alerte est alors choisi par l'utilisateur, soit sous forme de SMS, ou bien de courriel. Dans ces alertes sont présents les relevés des capteurs concernés avec le seuil franchi.

Description du cas d'utilisation

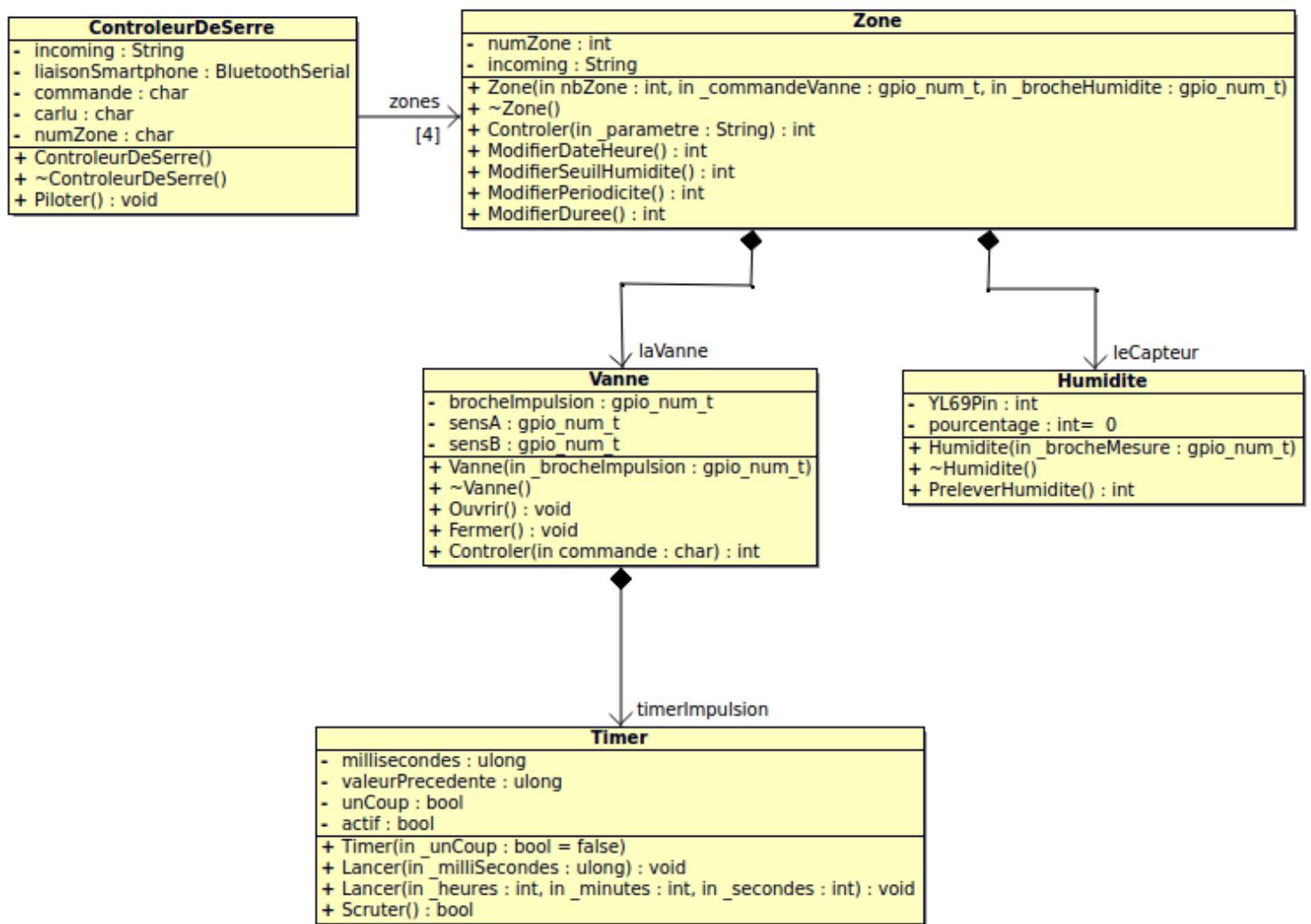
Nom du cas d'utilisation	Envoyer une alerte en cas de dépassement de seuil
Utilisateur concerné	Chien de garde
Étudiant en charge	Étudiant numéro 4, Marvyn ROCHER
Préconditions	Les informations relevées des différents capteurs ont été relevées, et sont présentes dans la base de données.
Scénario nominal	Lecture du numéro de téléphone, de l'e-mail du responsable à contacter en cas d'accident, ces informations sont présentes dans la base de données. Un SMS ou un e-mail sont envoyés au responsable en fonction de son choix d'alerte.
Post-condition(s)	Les valeurs sont comparées. Un SMS, ou courriel sont envoyés en cas de dépassement des seuils, ou en cas d'incident.

Diagramme de séquence

D- Conception détaillée

I/ Conception détaillée: Étudiant 1, MICHAUD Théo

1. Diagramme de classe



Nous pouvons noter que la classe débit d'eau n'est pas représentée, car elle n'est pas encore réalisée, la classe humidité quant à elle, est présente bien qu'elle puisse être encore modifiée, car incomplète pour l'instant.

2. Description des classes

1. Classe : contrôleurDeSerre

Cette classe, grâce au Bluetooth va nous permettre de récupérer la chaîne, transmise par l'application lors du choix du mode fait par le maraîcher, on va comparer la chaîne pour trouver quelle est la zone sélectionnée et le choix du mode effectué.

1. Piloter

Pour cela, la classe fait appel à la méthode [Piloter\(\)](#).

Piloter()			
Paramètre d'entrée	Chaîne reçue de l'application		
Paramètre de retour	XX		
Algorithme	Variables	Types	
DÉBUT TANT QUE liaisonSmartphone → OK carlu ← lire liaisonSmartphone incoming += carlu FINTANTQUE SI incoming Compare « 0 » !=0 ALORS commande ← prendre caractère 0 de la chaîne SWITCH commande CAS 1 = « z » numZone ← prendre caractère 1 de la chaîne prendre le caractère 2 de la chaîne et le passer en parametre de Controler() FINSWITCH incoming ← « » FINSI POUR indice=0 indice <4 indice ++ zone[indice] → Controler(« ») FINPOUR FIN	♦ commande ♦ numZone ♦ incoming ♦ calru ♦ zones	♦ char ♦ char ♦ String ♦ char ♦ Zone	

2. Classe : Zone

Cette classe est appelée par la classe ContrôleurDeSerre pour comparer le reste de la chaîne reçue par le Bluetooth. Elle permet de comprendre dans quel mode est positionnée la vanne : Manuel, programmé ou automatique.

1. Contrôler

Pour cela, la classe fait appel à la méthode `Contrôler(String _parametre)`, dans ce programme pour le mode auto et programmé le code n'est pas fini alors les modes ne fonctionnent pas encore.

<code>Contrôler(String _parametre)</code>			
Paramètre d'entrée	Chaîne reçue de l'application		
Paramètre de retour	Mode choisi... et action à effectuer		
Algorithme		Variables	Types
DÉBUT	<pre> parametre ← prendre caractère 0 de la chaîne SWITCH parametre CAS 1 = « m » SI caractère 1 de la chaîne _parametre = 'o' laVanne ← Contrôler('O') FINSI SI caractère 1 de la chaîne _parametre = 'f' laVanne ← Contrôler('F') FINSI CAS 2 = « p » CAS 3 = « a » FINSWITCH laVanne ← Contrôler('P') FIN </pre>	<ul style="list-style-type: none"> ◆ <code>_parametre</code> ◆ <code>parametre</code> ◆ <code>laVanne</code> 	<ul style="list-style-type: none"> ◆ <code>String</code> ◆ <code>char</code> ◆ <code>Vanne</code>

3. Classe : Vanne

Cette classe s'occupe d'ouvrir ou de fermer une vanne en fonction de la lettre reçue. Si la commande ‘O’ est reçue on ouvre, si ‘F’ on ferme. Mais elle gère aussi, le temps de l’impulsion envoyé sur l’électrovanne qui doit être de 250ms, grâce à un timer.

J’ai eu recours à l’utilisation du driver GpiO pour reconnaître les broches de l’ESP32.

J’ai également dû utiliser un mutex pour gérer l’ouverture ou la fermeture.

1. Ouverture d’une vanne

Pour ouvrir une vanne, la classe fait appel à la méthode `Ouvrir()`

Ouvrir()			
Paramètre d’entrée	X		
Paramètre de retour	X		
Algorithme	Variables	Types	
DÉBUT SI !mutex ALORS mutex ← true broche sens A ← 1 broche sens B ← 0 broche impulsion ← 1 Lancer timerImpulsion ← 250ms etat ← EN_OUVERTURE FINSI SINON etat ← EN_ATTENTE_OUVERTURE FIN	◆ etat ◆ brocheImpulsion ◆ sensA ◆ sensB ◆ mutex	◆ ETAT_VANNE ◆ const gpio_num_t ◆ const gpio_num_t ◆ const gpio_num_t ◆ bool	

2. Fermeture d'une vanne

Pour fermer la vanne, la classe fais appel à la méthode [Fermer\(\)](#)

Fermer()			
Paramètre d'entrée	X	Variables	Types
Paramètre de retour	X		
Algorithme		Variables	Types
DÉBUT			
SI !mutex			
ALORS mutex ← true	◆ etat		◆ ETAT_VANNE
broche sens A ← 0	◆ brocheImpulsion		◆ const gpio_num_t
broche sens B ← 1	◆ sensA		◆ const gpio_num_t
broche impulsion ← 1	◆ sensB		◆ const gpio_num_t
Lancer timerImpulsion ← 250ms	◆ mutex		◆ bool
etat ← EN_FERMETURE			
FINSI			
SINON			
etat ← EN_ATTENTE_FERMETURE			
FIN			

3. Contrôler une vanne

Cette classe est aussi composée de la méthode ‘Controler()’, qui nous permet en fonction de l’état des vannes d’appeler la méthode Ouvrir ou Fermer()

Controler(char commande)			
Paramètre d’entrée	Char commande		
Paramètre de retour	XX		
Algorithme		Variables	Types
DÉBUT			
SWITCH			
CAS 1 ← en_ouverture		♦ etat	♦ ETAT_VANNE
SI timerImpulsion.Scruter()		♦ brocheImpulsion	♦ const gpio_num_t
broche impulsion ← 0		♦ commande	♦ char
etat ← REPOS		♦ timerImpulsion	♦ Timer
mutex ← false		♦ mutex	♦ bool
FINSI			
CAS 2 ← en_fermeture			
SI timerImpulsion.Scruter()			
broche impulsion ← 0			
etat ← REPOS			
mutex ← false			
FINSI			
CAS 3 ← repos			
SI commande = ‘O’			
ouvrir ()			
FINSI			
SI commande = ‘F’			
fermer ()			
FINSI			
CAS 4 ← en_attente_ouverture			
SI !mutex			
ouvrir ()			
FINSI			
CAS 5 ← en_attente_ouverture			
SI !mutex			
fermer ()			
FINSI			
FINSWITCH			
FIN			

4. Classe : Timer

Cette classe nous permet de créer et de lancer différents Timer.

Elle est composée de trois méthodes, Lancer(), une autre méthode Lancer() et Scruter().

1. Lancer

La méthode [Lancer](#)(unsigned long _millisecondes), va permettre de lancer le timer avec le temps en millisecondes

Lancer(unsigned long _millisecondes)		
Paramètre d'entrée	_millisecondes	
Paramètre de retour	XX	
	Algorithme	Variables
DÉBUT	<pre>_millisecondes ← millisecondes valeurPrecedente ← millis() actif ← true</pre>	<ul style="list-style-type: none"> ◆ _millisecondes ◆ millisecondes ◆ actif ◆ valeurPrecedente
FIN		<ul style="list-style-type: none"> ◆ unsigned long ◆ unsigned long ◆ bool ◆ unsigned long

La méthode [Lancer](#)(int _heures, int _minutes, int _secondes), va permettre de lancer le timer avec le temps exprimé en heures, minutes, secondes.

Lancer(int _heures, int _minutes, int _secondes)		
Paramètre d'entrée	_heures, _minutes, _secondes	
Paramètre de retour	XX	
	Algorithme	Variables
DÉBUT	<pre>millisecondes ← 60 * (60 * _heures + _minutes) + _secondes) * 1000 valeurPrecedente ← millis() actif ← true</pre>	<ul style="list-style-type: none"> ◆ _millisecondes ◆ millisecondes ◆ actif ◆ valeurPrecedente
FIN		<ul style="list-style-type: none"> ◆ unsigned long ◆ unsigned long ◆ bool ◆ unsigned long

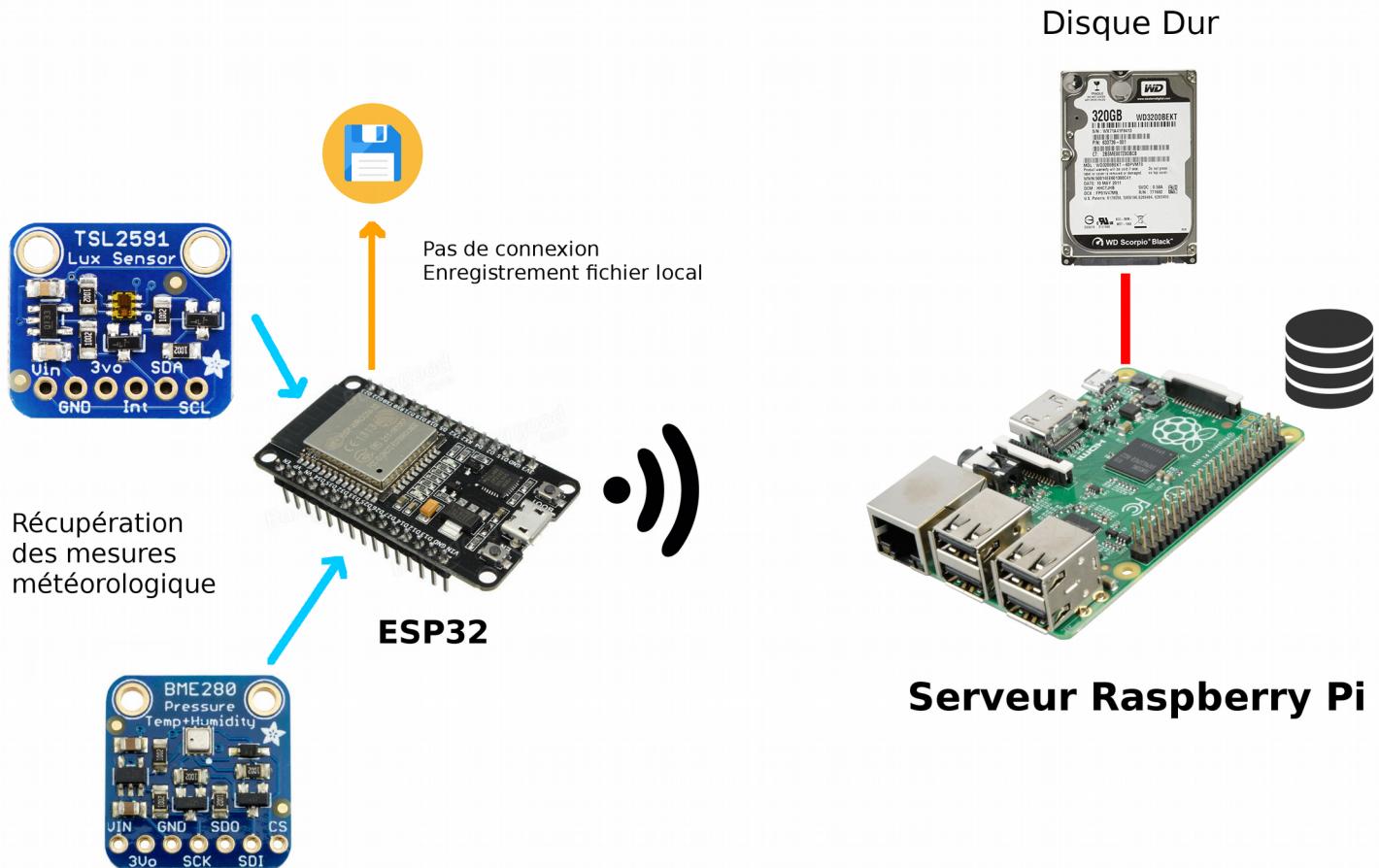
2. Scruter

La méthode `Scruter()`, va permettre de lancer le timer avec le temps exprimé en heures, minutes, secondes.

Scruter()			
Paramètre d'entrée	XX		
Paramètre de retour	retour		
	Algorithme	Variables	Types
DÉBUT	<pre> retour ← false SI actif et millis()-valeurPrecedente≥ millisondes valeurPrecedente ← millis() retour ← true SI unCoup actif ← false FINSI FINSI retourner retour FIN </pre>	<ul style="list-style-type: none"> ◆ unCoup ◆ millisondes ◆ actif ◆ valeurPrecedente ◆ retour 	<ul style="list-style-type: none"> ◆ bool ◆ unsigned long ◆ bool ◆ unsigned long ◆ bool

II/ Conception détaillée: Étudiant 2, AUVÉ Killian

1. Synoptique personnel



L'objet connecté ESP32 va pouvoir effectuer la mesure auprès des capteurs. Les capteurs vont pouvoir lire la température, l'hygrométrie et la luminosité. L'ESP32 va pouvoir écrire et lire dans un fichier local pour pouvoir enregistrer les données et mémoriser des incidents (perte de connexion) lorsque la connexion à la base de données est perdue. L'objet connecté va pouvoir se connecter à la base de données et envoyer les données de la serre et les données présentes dans le fichier local.

Le serveur Raspberry PI, quant à lui va pouvoir recevoir les données, les lire, puis il va pouvoir mettre à jour la base de données.

2. Diagramme de classe

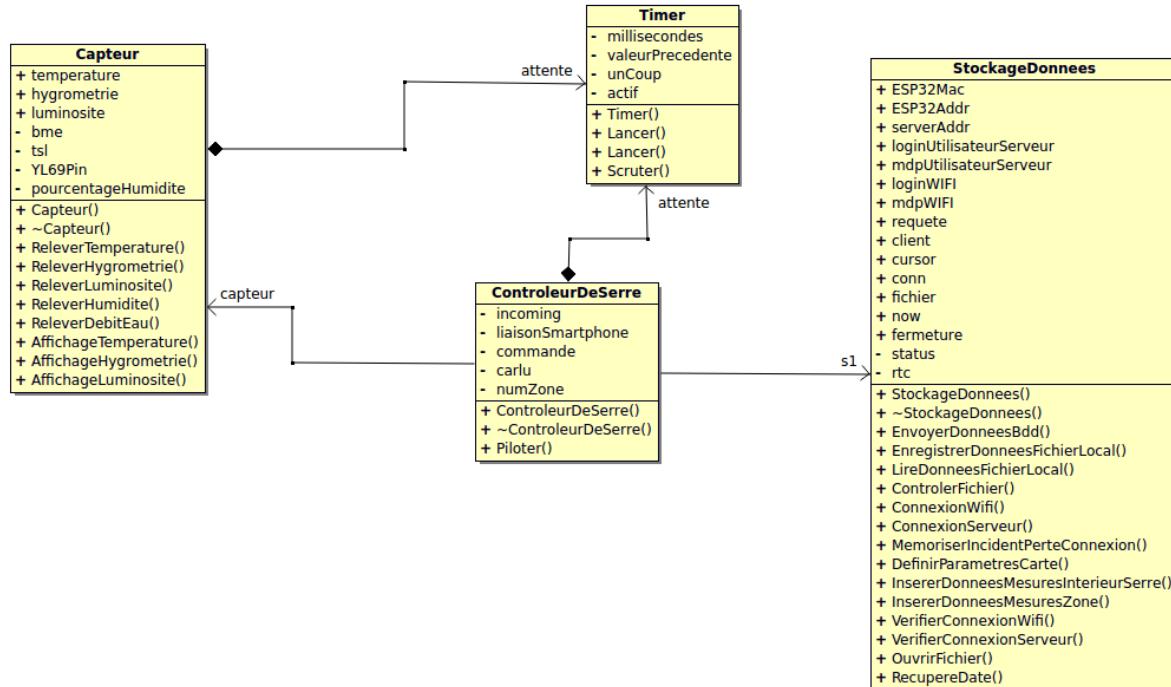


Diagramme de classe : étudiant n°2

Notre programme situé sur l'ESP32 contiendra 4 classes :

- La classe **controleurDeSerre** est la classe où toutes nos opérations se dérouleront.
- La classe **Capteur** va nous permettre de récupérer toutes les données liées aux capteurs.
- La classe **Stockage de données** va permettre d'insérer les données des capteurs dans une requête SQL afin de l'envoyer au serveur pour pouvoir être exploitée.
- La classe **Timer** va nous permettre de créer un Timer pour que toutes les opérations soient répétées toutes les 15 minutes.

3. Description des classes

1. Classe : ControleurDeSerre

1. Initialisation

La méthode **Initialisation()** permet d'initialiser le système, de définir les paramètres de la carte, de créer une connexion au Wi-Fi et au serveur.

Initialisation()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
DÉBUT MySQL_Connection conn(&s1->client) Creation timer à 15 minutes DefinirParametresCarte(); s1->ConnexionWifi(); conn ← s1->ConnexionServeur(); FIN	◆ conn	◆ MySQL_Connection

2. Piloter

Cette classe va nous permettre de lancer toutes les méthodes des classes à partir de la méthode **Piloter()**. C'est ici que l'on va appeler les méthodes de relever de données météorologiques et d'envoi de données au serveur.

Piloter()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
DÉBUT TOUTES LES 15 MINUTES FAIRE Vérifier la connexion Wi-Fi Vérifier la connexion au serveur Relever la Temperature Relever l'Hygrometrie Relever la Luminosite Insérer les données dans la requête Envoyer les Donnees à la base de données FIN		

2. Classe : Capteur

Cette classe va se charger de relever les données auprès des différents capteurs du système. Pour lire les données des différents capteurs, nous allons utiliser 3 méthodes.

1 ReleverTemperature

La méthode **ReleverTemperature()** qui permet de relever la température (en degrés) auprès du capteur BME280.

ReleverTemperature()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
DÉBUT temperature ← bme.readTemperature(); FIN	<ul style="list-style-type: none"> ◆ temperature ◆ bme 	<ul style="list-style-type: none"> ◆ Float ◆ Adafruit_BM_E280

2. ReleverHygrometrie

La méthode **ReleverHygrometrie()** qui permet de récupérer l'hygrométrie (en %) auprès du capteur BME280.

ReleverHygrometrie()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
DÉBUT hygrometrie ← bme.readHumidity(); FIN	<ul style="list-style-type: none"> ◆ Hygrometrie ◆ bme 	<ul style="list-style-type: none"> ◆ Float ◆ Adafruit_BME280

3. ReleverLuminosite

La méthode **ReleverLuminosite()** qui permet de relever la luminosité (en lux) auprès du capteur TSL2591.

ReleverLuminosite()			
Paramètre d'entrée	X		
Paramètre de retour	X		
Algorithme	Variables	Types	
DÉBUT luminosite ← tsl.getLuminosity(TSL2591_VISIBLE); FIN	<ul style="list-style-type: none"> ◆ Luminosite ◆ tsl ◆ TSL2591_VISIBLE 	<ul style="list-style-type: none"> ◆ Float ◆ Adafruit_TS_L2591 	

3. Classe : StockageDonnes

La classe StockageDonnees va nous permettre d'effectuer toutes nos interactions avec la base de données. On va pouvoir insérer, sélectionner des données. Gérer les données présente sur la carte SD (écriture et lecture).

1. ConnexionWifi

La méthode **ConnexionWifi()** va nous permettre de nous connecter à un réseau Wi-Fi afin de pouvoir envoyer nos données.

ConnexionWifi()			
Paramètre d'entrée	X		
Paramètre de retour	X		
Algorithme	Variables	Types	
DÉBUT status ← WiFi.begin(loginWIFI,mdpWIFI); Si status == PasConnecté ALORS Afficher pas connecter WiFi.begin(loginWIFI, mdpWIFI); SINON Afficher Connecté FIN SI FIN	<ul style="list-style-type: none"> ◆ Statuts ◆ loginWIFI ◆ mdpWIFI 	<ul style="list-style-type: none"> ◆ Bool ◆ char [] ◆ char [] 	

2. VerifierConnexionWifi

La méthode **VerifierConnexionWifi()** permet de vérifier si la connexion au réseau Wi-Fi est encore effectuée, si ce n'est pas le cas, elle va tenter de reconnecter l'objet connecté au réseau Wi-Fi.

VerifierConnexionWifi()			
Paramètre d'entrée	X		
Paramètre de retour	X		
Algorithme	Variables		Types
DÉBUT Si status == PasConnecté ALORS Afficher pas connecter WiFi.begin(loginWIFI, mdpWIFI); SINON Afficher Connecté FIN SI FIN	<ul style="list-style-type: none"> ◆ Status ◆ loginWIFI ◆ mdpWIFI 	<ul style="list-style-type: none"> ◆ Bool ◆ char [] ◆ char [] 	

3. ConnexionServeur

La méthode **ConnexionServeur()** va nous permettre de nous connecter au serveur Raspberry pi. L'objet connecté se connecte au serveur à l'aide de son adresse, de son identifiant et de son mot de passe.

ConnexionServeur()			
Paramètre d'entrée	X		
Paramètre de retour	MySQL_Connection conn		
Algorithme	Variables		Types
DÉBUT MySQL_Connection conn(&client); conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur) FIN	<ul style="list-style-type: none"> ◆ ServerAddr ◆ loginUtilisateurServeur ◆ mdpUtilisateurServeur 	<ul style="list-style-type: none"> ◆ IPAddress ◆ char[] ◆ char[] 	

4. VerifierConnexionServeur

La méthode **VerifierConnexionServeur**(MySQL_Connection conn) permet de vérifier si la connexion au serveur est encore effectuée, tant que la connexion n'est pas rétablie, on essaye de se reconnecter au serveur.

VerifierConnexionServeur(MySQL_Connection conn)		
Paramètre d'entrée	MySQL_Connection conn	
Paramètre de retour		
Algorithme	Variables	Types
DÉBUT Si la connexion n'est pas établie ALORS <code>conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur);</code> <code>VerifierConnexionWifi();</code> FIN SI FIN	<ul style="list-style-type: none"> ◆ ServerAddr ◆ loginUtilisateurSe rveur ◆ mdpUtilisateurSer veur 	<ul style="list-style-type: none"> ◆ IPAddress ◆ char[] ◆ char[]

5. DefinirParametresCarte

La méthode **DefinirParametresCarte()** va nous permettre d'attribuer à la carte ESP32 une adresse IP et une adresse MAC.

DefinirParametresCarte()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
DÉBUT <code>Ethernet.begin(ESP32Mac, ESP32Addr);</code> FIN	<ul style="list-style-type: none"> ◆ ESP32Mac ◆ ESP32Addr 	<ul style="list-style-type: none"> ◆ Byte ◆ IPAddress

6. InsererDonneesMesuresInterieurSerre

Cette méthode va nous permettre d'insérer les variables passées en paramètre dans une chaîne de caractère qui sera envoyée au serveur.

InsererDonneesMesuresInterieurSerre(float temperature, float hygrometrie, float luminosite)			
Paramètre d'entrée	temperature, hygrometrie, luminosite		
Paramètre de retour	X		
Algorithme	Variables	Types	
<p>DÉBUT</p> <pre>String chaineTemperature, chaineHygrometrie, chaineLuminosite, date; chaineTemperature ← temperature chaineHygrometrie ← hygrometrie chaineLuminosite ← luminosite date ← RecupereDate() requeteMesureInterieur ← "INSERT INTO laSerre.MesuresSerreInterieur(temperatureInterieur,hygrometrieInterieur,luminositeInterieur,DateMesureInterieur,idSerre,idZone) VALUES (" + chaineTemperature + "," + chaineHygrometrie + "," + chaineLuminosite + "," + date + " , " + IDSERRE + ",1LaSerre.MesuresSerreInterieur(temperatureInterieur,hygrometrieInterieur,luminositeInterieur,DateMesureInterieur,idSerre,idZone) VALUES (" + chaineTemperature + "," + chaineHygrometrie + "," + chaineLuminosite + "," + date + " , " + IDSERRE + ",1)"; FIN</pre>	<ul style="list-style-type: none"> ◆ ChaineTemperatur e ◆ chaineHygrometrie ◆ chaineLuminosite ◆ date ◆ requeteMesureInterieur 	<ul style="list-style-type: none"> ◆ String ◆ String ◆ String ◆ String ◆ String 	

7. InsererDonneesMesuresZone

Cette méthode va nous permettre d'insérer les données du capteur de débit d'eau et d'humidité du sol passées en paramètre dans une chaîne de caractère qui sera envoyée au serveur.

InsererDonneesMesuresZone(float debitEau, float humiditeSol)		
Paramètre d'entrée	float debitEau, float humiditeSol	
Paramètre de retour	X	
Algorithme	Variables	Types
<p>DÉBUT</p> <pre>String chainedebitEau, chainehumiditeSol, date; chainedebitEau ← debitEau chainedebitEau ← humiditeSol date ← RecupereDate() requeteMesureZone ← "INSERT INTO laSerre.MesuresZone(debitEau,humiditeSol,dateMesu re,idZone) VALUES (" + chainedebitEau + "," + chainehumiditeSol + "," + date + " , " + IDZONE + ")" FIN</pre>	<ul style="list-style-type: none"> ◆ ChainedebitEau ◆ chainehumiditeSol ◆ date ◆ requeteMesureZone 	<ul style="list-style-type: none"> ◆ String ◆ String ◆ String ◆ String

8. OuvrirFichier

La méthode **OuvrirFichier()** va nous permettre d'ouvrir un fichier présent sur la carte SD.

OuvrirFichier()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
<pre> DÉBUT pinmode ← 5 Si Shield SD n'est pas branchée ALORS Afficher erreur initialisation SINON fichier = SD.open("/donnees.txt", FILE_WRITE); SI le fichier n'est pas ouvert ALORS Afficher erreur ouverture fichier FIN SI FIN SI FIN </pre>	<ul style="list-style-type: none"> ◆ Fichier 	<ul style="list-style-type: none"> ◆ File

9. EnregisterDonneesFichierLocal

Cette méthode va nous permettre de récupérer la date et l'heure pour l'insertion des données.

EnregisterDonneesFichierLocal(float temperature, float hygrometrie, float luminosite)		
Paramètre d'entrée	temperature, hygrometrie, luminosite	
Paramètre de retour	X	
Algorithme	Variables	Types
<pre> DÉBUT SI le fichier n'est pas ouverte ALORS OuvrirFichier() FIN SI date ← RecupereDate() chaine ← temperature chaine ← hygrometrie chaine ← luminosite chaine ← date fichier ← -chaîne SI la fermeture du fichier est confirmée ALORS fichier.close(); FIN SI FIN </pre>	<ul style="list-style-type: none"> ◆ Chaîne ◆ temperature ◆ hygrometrie ◆ luminosite ◆ fichier 	<ul style="list-style-type: none"> ◆ String ◆ float ◆ float ◆ float ◆ String ◆ File

10. RecupererDate

RecupereDate()		
Paramètre d'entrée	X	
Paramètre de retour	String date	
Algorithme	Variables	Types
<p>DÉBUT</p> <pre>now ← rtc.now(); annee ← String(now.year()); mois ← String(now.month()); jour ← String(now.day()); heure ← String(now.hour()); minute ← String(now.minute()); seconde ← String(now.second()); date ← annee + mois +jour + heure + minute +seconde retourner date FIN</pre>	<ul style="list-style-type: none"> ◆ Now ◆ annee ◆ mois ◆ jour ◆ heure ◆ minute ◆ date 	<ul style="list-style-type: none"> ◆ DateTime ◆ String ◆ String ◆ String ◆ String ◆ String ◆ String

11. LireDonneesFichierLocal

LireDonneesFichierLocal()		
Paramètre d'entrée	X	
Paramètre de retour	X	
Algorithme	Variables	Types
<p>DÉBUT</p> <pre>fichier.read TANT QUE tous les caractères du fichier n'ont pas été lus FAIRE chaîne ← Lire caractère du fichier FIN TANT QUE FIN</pre>	<ul style="list-style-type: none"> ◆ Fichier ◆ chaîne 	<ul style="list-style-type: none"> ◆ File ◆ String

4. Classe : Timer

Cette classe va nous permettre de créer un timer afin de pouvoir effectuer les opérations toutes les 15 minutes.

1. Lancer

Les méthodes **Lancer**(unsigned long _millisecondes) vont nous permettre de définir la durée du timer.

Lancer(unsigned long _millisecondes)			
Paramètre d'entrée	_millisecondes		
Paramètre de retour	X		
Algorithme	Variables		Types
DÉBUT $_millisecondes \leftarrow millisecondes$ $valeurPrecedente \leftarrow millis()$ $actif \leftarrow true$	<ul style="list-style-type: none"> ◆ $_millisecondes$ ◆ $millisecondes$ ◆ $actif$ ◆ $valeurPrecedente$ 	<ul style="list-style-type: none"> ◆ $unsigned\ long$ ◆ $unsigned\ long$ ◆ $bool$ ◆ $unsigned\ long$ 	
FIN			

La méthode ci-dessous est celle que nous allons utiliser pour définir le timer, en effet, c'est plus simple d'utiliser celle du dessous que celle du dessus où il faudrait définir 15 minutes en milliseconde.

Lancer(int _heures, int _minutes, int _secondes)		
Paramètre d'entrée	_heures, _minutes, _secondes	
Paramètre de retour	X	
Algorithme	Variables	Types
<p>DÉBUT</p> <pre>millisecondes ← 60 * (60 * _heures + _minutes) + _secondes) * 1000 valeurPrecedente ← millis() actif ← true</pre> <p>FIN</p>	<ul style="list-style-type: none"> ◆ _millisecondes ◆ millisecondes ◆ actif ◆ valeurPrecedente 	<ul style="list-style-type: none"> ◆ unsigned long ◆ unsigned long ◆ bool ◆ unsigned long

2. Scruter

La méthode **Scruter()** va nous permettre de lancer le timer.

Scruter()		
Paramètre d'entrée	X	
Paramètre de retour	bool	
Algorithme	Variables	Types
<pre> DÉBUT retour ← false SI actif et millis()-valeurPrecedente≥= millisecondes valeurPrecedente ← millis() retour ← true SI unCoup actif ← false FINSI FINSI retourner retour FIN </pre>	<ul style="list-style-type: none"> ◆ unCoup ◆ millisecondes ◆ actif ◆ valeurPrecedente ◆ retour 	<ul style="list-style-type: none"> ◆ bool ◆ unsigned long ◆ bool ◆ unsigned long ◆ bool

III/ Conception détaillée: Étudiant 3, VILLERMIN Maxime

1. Préambule

Le serveur web étant prêt nous hébergeons le site à l'adresse '172.18.58.213/var/www/html/ProjetSerre' de la raspberry.

Prérequis pour le développement : Comme le site web est hébergé sur la raspberry, je ne pouvais pas ouvrir mon projet depuis netbeans qui lui est sur mon ordinateur local. J'ai utilisé le « SSH Filesystem » qui permet de monter sur son système de fichier, un autre système de fichier distant à travers une connexion SSH le tout avec de divers droits utilisateurs. L'avantage est de manipuler les données distantes avec n'importe quel gestionnaire de fichiers (en ligne de commande par exemple).

Pour réaliser cette manipulation, on ouvre le terminal de l'ordinateur et l'on écrit :

→ ‘sshfs root@172.18.58.213:/var/www remote’

On peut donc accéder au dossier ‘www’ situé dans ‘var’ de la raspberry à l'adresse ‘172.18.58.213’ dans le dossier local ‘remote’.

2. Le site web

1. Bootstrap



Qu'est-ce que Bootstrap ?

Il fournit des classes HTML/CSS de composant préfait, il gère aussi le responsive (gestion des différents appareils : smartphone, tablette, ordinateur).

C'est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. L'intérêt est de pouvoir profiter d'une large bibliothèque de choix de design en HTML et en CSS gratuitement, c'est donc un gain de temps pour le développement du site.

Pour la partie graphique du site web, l'étudiant n°3 a téléchargé un template (une base de site déjà faite) sur Internet. Le template a été téléchargé sur '<https://startbootstrap.com/themes/>' s'appelle « Freelancer ».



J'ai donc modifié toutes les parties qui ne m'intéressent pas et gardé celle qui pouvait être intéressante (barre de navigation, background et page d'accueil).

2 W3Schools

Pour m'aider au développement du site web, j'ai utilisé le site '<https://www.w3schools.com/>'. On y retrouve plein d'exemples de code en JavaScript, HTML, CSS... La partie qui m'a surtout aidé est 'Learn Bootstrap', qui donne des exemples de code intégré à Boostrap (le modal que j'ai utilisé par exemple).

3 Organisations des fichiers du projet

J'ai choisi d'organiser mes fichiers en plusieurs parties :

- Document JS : qui contient mes fichiers JS
- Document PHP : qui contient mes fichiers PHP
- Document Programmer : qui contient mes fichiers PHP de la partie programmer, mon .htaccess et mon .htpassword (on y reviendra plus bas)

Les autres documents du projet sont les documents Boostrap qui accueillent les styles de mise en page, les polices d'écritures...



2. Jquery

Nous allons-nous intéressé plus particulièrement à l'appel de la fonction JavaScript et comment nous appellerons cette fonction. Pour cela, nous utiliserons une bibliothèque JavaScript qui est Jquery.

1. Qu'est-ce que jQuery?

JQuery est une bibliothèque JavaScript qui est principalement utilisée sur les pages WEB interactives. Cette bibliothèque permet de manipuler plus facilement le DOM (Document ObjectModel), d'utiliser l'AJAX (Asynchronous JavaScript And XML), ainsi que de créer des animations. L'origine de cette bibliothèque est de permettre de développer des applications plus rapidement, donc de permettre de gagner du temps.

Comment l'implémenter à notre site?

Il l'est déjà, car on a utilisé un template bootstrap qui nous l'inclut directement.

2. Fonctionnement

Nous utilisons la bibliothèque jQuery lors d'un changement de sélection dans les listes déroulantes. Cela va permettre lorsqu'un changement de sélection se fait, qu'il y ait un appel de la fonction JavaScript prévu à cet effet. Donc quand on va sélectionner une Serre, la liste déroulante des zones sera mise à jour par rapport au choix de la serre.

3. Où utilisons-nous la bibliothèque ?

Nous utilisons la bibliothèque jQuery dans le formulaire lors de la création de nos éléments. En effet, la bibliothèque jQuery nous permettre de faire le lien entre notre élément et la fonction auquel on souhaite associer l'élément par l'intermédiaire d'un événement JavaScript.



3. AJAX

Ensuite, nous voulons que l'internaute ne soit pas dérangé par un rechargement intempestif de la page à chaque fois qu'il sélectionne quelque chose dans une liste déroulante. Pour cela, nous allons utiliser de l'AJAX qui nous permettra d'actualiser les listes qui doivent être actualisées suivant ce qui a été sélectionné sans occasionner un rechargement complet de la page.

1. Qu'est-ce que AJAX?

L'AJAX (Asynchronous Javascript And XML) n'est ni une technologie ni un langage de programmation. Il s'agit plutôt d'un concept de programmation WEB reposant sur une combinaison de multiples technologies comme le JavaScript et le XML – d'où le nom AJAX.

À l'heure actuelle, le XML tend à être délaissé au profit du JSON. JSON est une forme de notation qui permet de représenter l'information de façon structurée. Ce concept va permettre d'avoir une meilleure maniabilité du site WEB. Lorsque ces technologies sont combinées dans le concept AJAX, les applications web sont capables de réaliser des mises à jour rapides de l'interface utilisateur sans devoir recharger la page entière du navigateur. Les applications fonctionnent plus rapidement et sont plus réactives aux actions de l'utilisateur.

Les diverses technologies qui peuvent être combinées dans l'AJAX sont:

- DOM et JavaScript sont utilisés pour modifier l'information présentée dans le navigateur par programmation.
- L'objet XMLHttpRequest est utilisé pour dialoguer de manière asynchrone avec le serveur Web.
- La notation XML est utilisée pour structurer les informations transmises entre Le serveur Web et le navigateur. L'AJAX est associé le plus souvent avec la librairie JavaScript jQuery, et du format de données JSON.

Nous allons donc utiliser ces différentes technologies pour faire de l'ajax

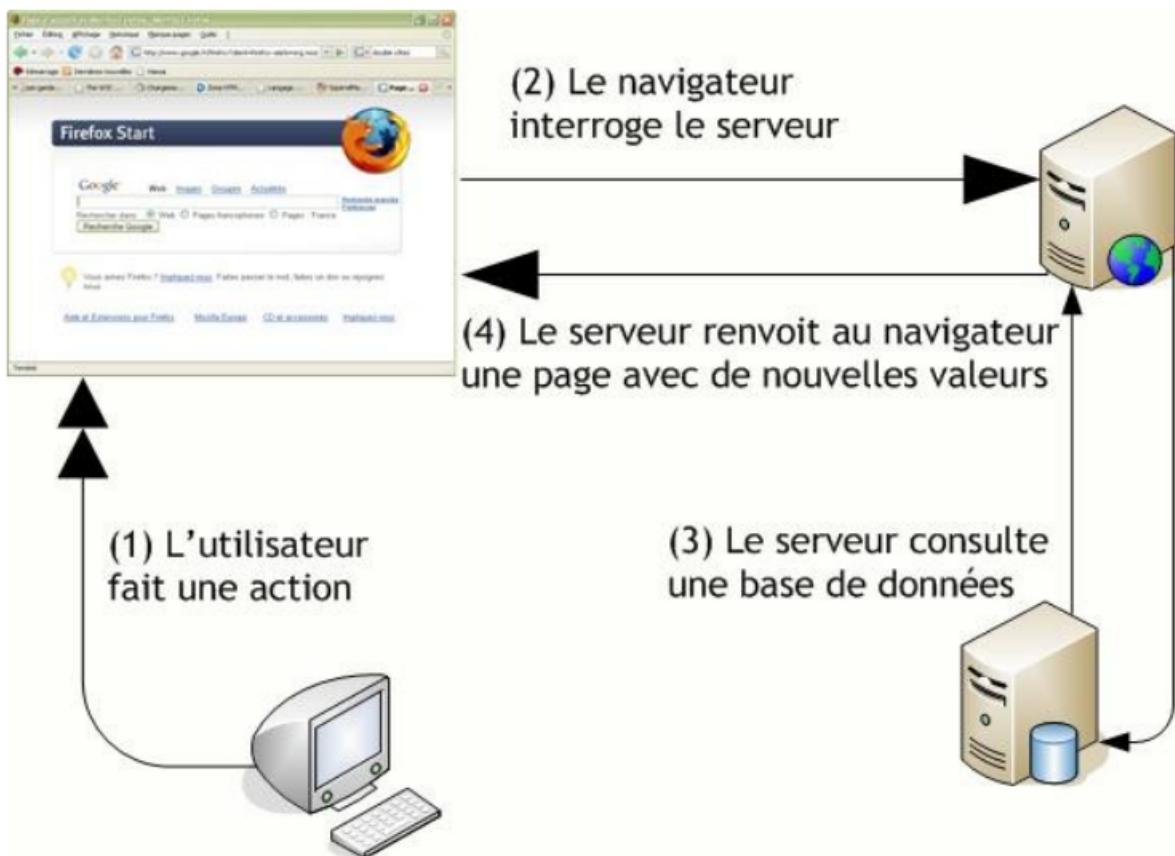
- DOM et JavaScript
- L'objet XMLHttpRequest
- La notation JSON

2. Fonctionnement

En simplifiant, on peut dire qu'AJAX se déroule suivant trois étapes:

- Une action utilisateur côté client (par exemple le remplissage et la soumission d'un formulaire)
- Le traitement de cette action côté serveur, et le renvoi éventuel, par le serveur, d'une réponse, généralement au format XML ou JSON
- Le traitement de la réponse en JavaScript et la modification conséquente de la page côté client.

Schéma de fonctionnement:



Le principe de la méthode est le suivant: enclencher le processus AJAX (étape1) et traiter la réponse (étape3) en utilisant les fonctions jQuery consacrées à AJAX. Quant à la réponse du serveur, elle sera retournée au format JSON (étape2).

3. De quoi avons-nous besoin?

Côté client, nous ferons appel à la bibliothèque JavaScript jQuery, qui devra donc être incluse à notre page.

Côté serveur, il nous faudra un langage capable de créer et de retourner une réponse au format de données JSON. Dans notre cas, nous utiliserons du PHP.

Pour la partie AJAX, nous avons trois parties distinctes. Dans un premier temps, on va voir le fichier JavaScript qui lance la requête AJAX et fait les mises à jour de la page. Dans un second temps, le contrôleur AJAX qui va faire le lien entre ma fonction et mon modèle. Et dans un troisième temps, le modèle qui lui contient toutes les requêtes qui vont être exécutées suivant dans quels cas on est. Pour mieux comprendre le fonctionnement d'AJAX, nous allons procéder de façon séquentielle.

4. JSON

1. Qu'est-ce que JSON?

Javascript Object Notation d'où son anagramme JSON est une forme de notation de données en JavaScript. Il s'agit d'une dérivée de la notation des objets du langage ECMAScript. Le format JSON permet de représenter de l'information structurée.

2. Structure d'un document JSON

Un document JSON ne comprend que deux éléments structurels:

- Des ensembles de paires nom/valeur;
- Des listes de valeurs;

Ces mêmes éléments représentent 3 types de données:

- Des objets;
- Des tableaux;
- Des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou nulle.

3. Avantage

Ces types de données sont assez génériques pour pouvoir représenter dans n'importe quel langage de programmation, et puis de pouvoir représenter n'importe quelle donnée concrète. Le principal avantage de ce langage est qu'il est simple à mettre en œuvre par un programmeur tout en étant complet. On peut citer d'autres avantages comme le fait que le langage JSON est peu verbeux, ce qui le rend lisible pour une machine, comme pour un être humain. De plus, ces types de données sont connus et simples à décrire.

4. Utilisation de JSON

Comme dit précédemment, JSON utilise une notation JavaScript, mais il est indépendant du langage de programmation. Il sert à faire communiquer différentes applications dans un environnement hétérogène. Il est utilisé comme langage de transport de données par AJAX et les services WEB.

5. JSON dans notre projet

Nous utilisons le format JSON lorsque l'on transmet notre tableau de données vers notre requête pour pouvoir remplacer les données de nos listes déroulantes.

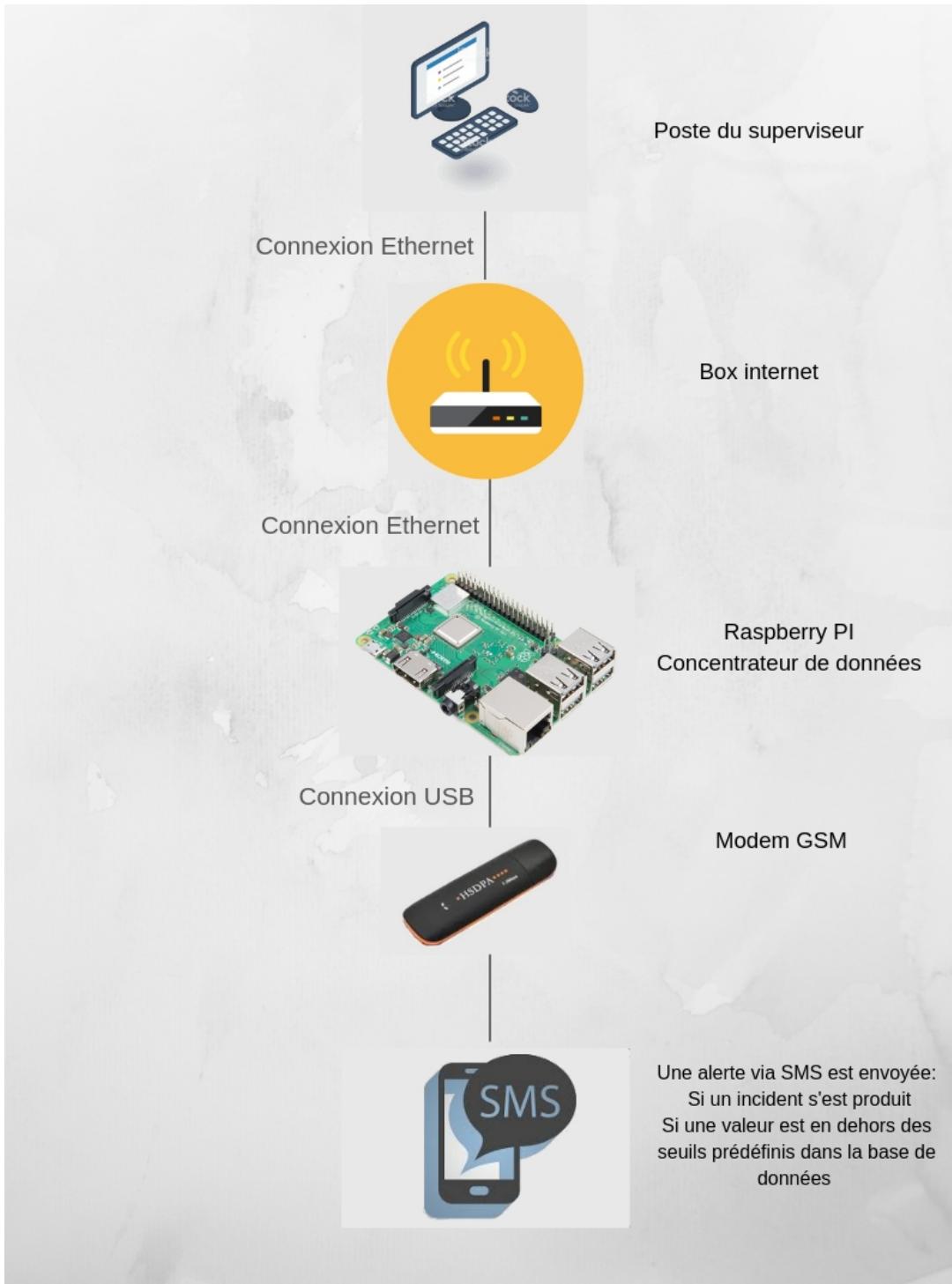
Exemple de fichier au format JSON :

Un exemple simple, définition d'un menu:
il s'agit d'un objet composé de membres qui sont un attribut et un tableau lequel contient d'autres objets, les lignes du menu.

```
{  
    "menu": "Fichier",  
    "commandes": [  
        {  
            "titre": "Nouveau",  
            "action": "CreateDoc"  
        },  
        {  
            "titre": "Ouvrir",  
            "action": "OpenDoc"  
        },  
        {  
            "titre": "Fermer",  
            "action": "CloseDoc"  
        }  
    ]  
}
```

IV/ Conception détaillée: Étudiant 4, ROCHER Marvyn

1.Synoptique personnel

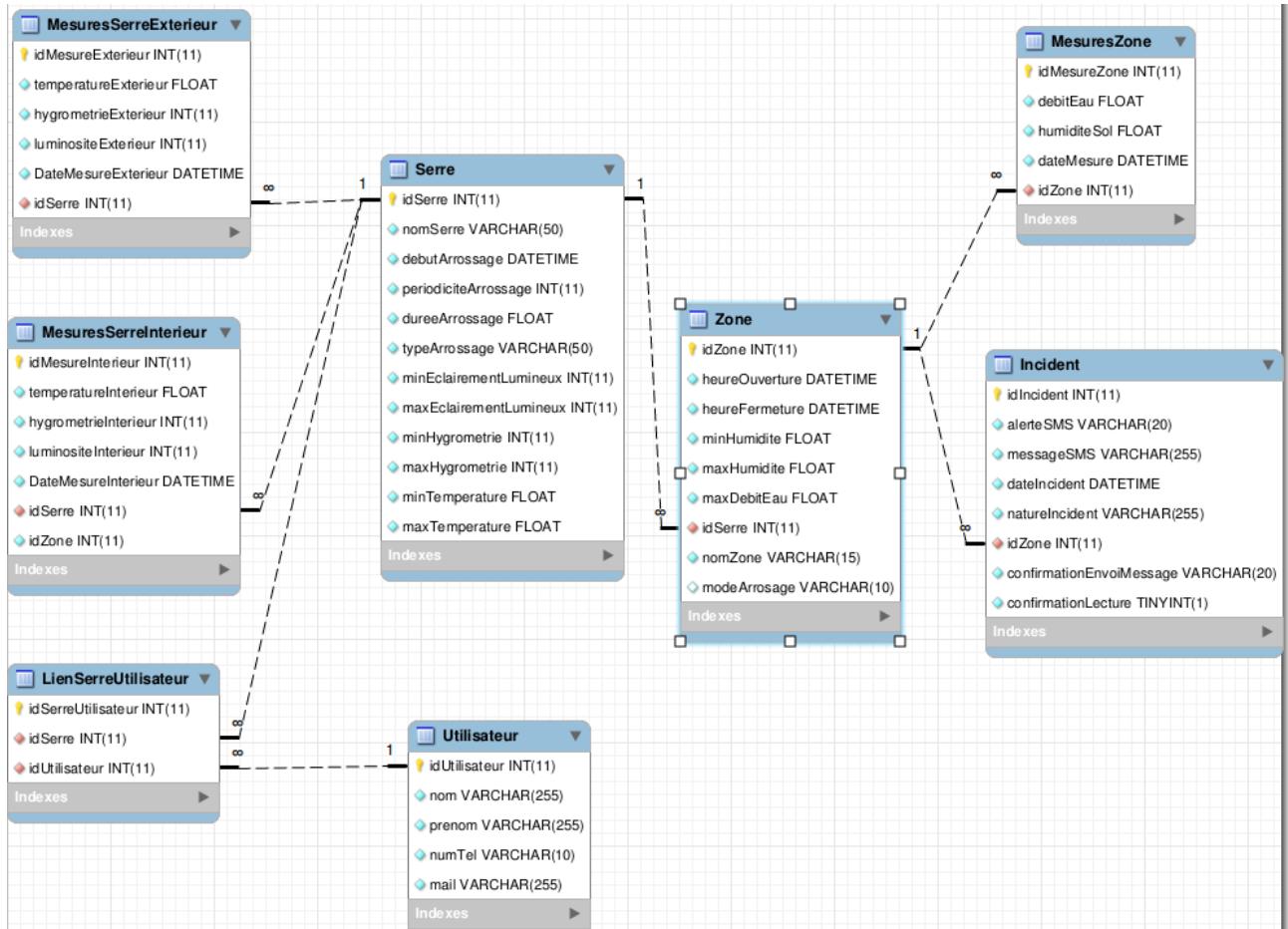


2. Les tables utilisées dans la base de données

Afin de pouvoir réaliser la partie affectée à l'étudiant n°4, il est nécessaire de savoir quelles sont les tables qui sont utilisées par l'étudiant afin de rendre compte de la bonne compréhension du projet.

1. Les tables utilisées dans la base de données

Afin de réaliser sa partie, l'étudiant n°4 utilise les tables présentes ci-dessus



Tables de la base de données concernant l'étudiant n°4

2. Détails des tables

1. MesuresSerreInterieur

Le rôle de la table MesuresSerreInterieur permet de répertorier l'ensemble des mesures tirées des capteurs situés à l'intérieur de la serre.

Nom du champ	Type du champ	Description
idMesureInterieur	INT : entier	Identifiant de la mesure
temperatureInterieur	FLOAT : réel	Valeurs de la température intérieure
hygrometrieInterieur	INT : entier	Valeurs de l'hygrométrie intérieure
luminositeInterieur	INT : entier	Valeurs de la luminosité intérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
idSerre	INT : entier	Identifiant de la serre
idZone	INT : entier	Identifiant de la zone

2. MesuresSerreExterieur

Le rôle de la table MesuresSerreExterieur permet de répertorier l'ensemble des mesures tirées des capteurs situés à l'extérieur de la serre

Nom du champ	Type du champ	Description
idMesureExterieur	INT : entier	Identifiant de la mesure
temperatureExterieur	FLOAT : réel	Valeurs de la température extérieure
hygrometrieExterieur	INT : entier	Valeurs de l'hygrométrie extérieure
luminositeExterieur	INT : entier	Valeurs de la luminosité extérieure
DateMesureInterieur	DATETIME : date	Date de la mesure
idSerre	INT : entier	Identifiant de la serre

3. LienSerreUtilisateur

Le rôle de la table LienSerreUtilisateur permet d'associer les utilisateurs aux serres

Nom du champ	Type du champ	Description
idSerreUtilisateur	INT : entier	Identifiant du lien
idSerre	INT : entier	Identifiant de la serre
idUtilisateur	INT : entier	Identifiant de l'utilisateur



4. Utilisateur

Le rôle de la table Utilisateur permet de répertorier l'ensemble des utilisateurs de la serre

Nom du champ	Type du champ	Description
idUtilisateur	INT : entier	Identifiant de la mesure
nom	VARCHAR(255) : chaîne de 255 caractères	Valeurs de la température extérieure
prénom	VARCHAR(255) : chaîne de 255 caractères	Valeurs de l'hygrométrie extérieure
numTel	INT : entier	Valeurs de la luminosité extérieure
mail	DATETIME : date	Date de la mesure
idSerre	INT : entier	Identifiant de la serre

5. Incident

Le rôle de la table Incident permet de répertorier l'ensemble des incidents qui se sont produits

Nom du champ	Type du champ	Description
idIncident	INT : entier	Identifiant de l'incident
alerteSMS	VARCHAR(20) : chaîne de 20 caractères	Confirmation du choix de l'alerte SMS
messageSMS	VARCHAR(255) : chaîne de 255 caractères	Message envoyé auprès du responsable
dateIncident	DATETIME : Date	Date de l'incident
natureIncident	VARCHAR(255) : chaîne de 255 caractères	Nature de l'incident
IdZone	INT : entier	Identifiant de la zone
confirmationEnvoiMessage	VARCHAR(20) : chaîne de 20 caractères	Confirmation de l'envoi du SMS
confirmationLecture	BOOL : booléen	Confirmation de lecture de l'alerte

6. MesuresZone

Le rôle de la table MesuresZone permet de répertorier l'ensemble des mesures liées à une zone

Nom du champ	Type du champ	Description
idMesureZone	INT : entier	Identifiant de la mesure de la zone
debitEau	FLOAT : réel	Mesure du débit d'eau
humiditeSol	FLOAT : réel	Mesure de l'humidité du sol
dateMesure	DATETIME : Date	Date de la mesure
idZone	INT : entier	Identifiant de la zone

7. Zone

Le rôle de la table Zone permet de répertorier l'ensemble des caractéristiques d'une zone (seuils, identifiants, heure d'ouverture et de fermeture)

Nom du champ	Type du champ	Description
idZone	INT : entier	Identifiant de la zone
heureOuverture	DATETIME : Date	Heure d'ouverture de la vanne
heureFermeture	DATETIME : Date	Heure de fermeture de la vanne
minHumidite	FLOAT : réel	Seuil minimum d'humidité
maxHumidite	FLOAT : réel	Seuil maximum d'humidité
maxDebitEau	FLOAT : réel	Seuil maximum de débit d'eau
idSerre	INT : entier	Identifiant de la serre
nomZone	VARCHAR(15) : chaîne de 15 caractères	Nom de la zone
modeArrosage	VARCHAR(10) : chaîne de 10 caractères	Mode d'arrosage (automatique, programmé, manuel)

8. Serre

Le rôle de la table Serre permet de répertorier l'ensemble des caractéristiques d'une serre

Nom du champ	Type du champ	Description
idSerre	INT : entier	Identifiant de la serre
nomSerre	VARCHAR(50) : chaîne de 50 caractères	Nom de la serre
debutArrosage	DATETIME : Date	Date du début d'arrosage de la serre
periodiciteArrosage	INT : entier	Périodicité de l'arrosage
dureeArrosage	FLOAT : réel	Durée de l'arrosage
typeArrosage	VARCHAR(50) : chaîne de 50 caractères	Type d'arrosage
minEclairementLumineux	INT : entier	Seuil minimum de l'éclairement lumineux
maxEclairementLumineux	INT : entier	Seuil maximum de l'éclairement lumineux
minHygrometrie	INT : entier	Seuil minimum de l'hygrométrie
maxHygrometrie	INT : entier	Seuil maximum de l'hygrométrie
minTemperature	FLOAT : réel	Seuil minimum de la température
maxTemperature	FLOAT : réel	Seuil maximum de la température

3. Présentation des tables

1. Le besoin

Le besoin est recevoir une alerte en cas de dépassement de seuil, or la Serre qui devient connectée, possède des capteurs intérieurs et extérieurs, nous devons donc récupérer ces valeurs intérieures, et extérieures présentes dans les tables MesuresSerreInterieur, MesuresSerreExterieur, MesuresZone. Mais également, nous avons besoin des coordonnées du responsable présentes dans la table Utilisateur.

Ensuite afin de prévenir le responsable des incidents, il serait préférable de pouvoir identifier dans quelle serre, et dans quelle zone se déroule l'incident. Ainsi nous trouverons l'ensemble de ses informations dans les tables : Serre, Zone, Incident.

2. Déroulement

Le responsable va sur le site Internet dédié à la gestion de la Serre, il renseigne ses coordonnées (e-mail, numéro de téléphone), dès lors ses informations sont stockées dans la base de données dans la table Utilisateur. Toutes les 15 minutes, la base de données est interrogée, et utilise donc les tables MesuresSerreExterieur, MesuresSerreInterieur, MesuresZone afin d'obtenir l'ensemble des mesures.

Afin de pouvoir faire le contrôle de l'état de la Serre, la base de données est également interrogée pour connaître les seuils qui ont été fixés, et savoir si une alerte a besoin d'être envoyée. Ces informations de seuil se situent dans les tables : Zone, Serre.

En effet, une zone n'a pas les mêmes seuils qu'une Serre.

Une zone a pour caractéristique d'avoir :

- Un seuil minimum et maximum d'humidité
- Un seuil maximum de débit d'eau

Tandis qu'une Serre a pour caractéristiques d'avoir :

- Un seuil minimum et maximum d'hygrométrie
- Un seuil minimum et maximum de température
- Un seuil minimum et maximum de luminosité

3. Réalisation des tâches

Le projet de supervision de serre étant le nôtre, la réussite de celui-ci tient notamment par l'incorporation de divers capteurs (température, hygrométrie, luminosité) afin de rendre compte de l'état de celle-ci.

Afin d'assurer la bonne réussite de notre projet, il a été nécessaire d'installer divers utilitaires, tels que :

- Apache
- MariaDB : la base de données qui a été choisie
- PhpMyAdmin : pouvoir gérer la base de données depuis une interface graphique
- Gammu : pouvoir envoyer des SMS depuis le Modem GSM depuis le Raspberry PI
- PostFix : pouvoir envoyer des mails directement depuis la Raspberry

L'étudiant n°4 a donc été affecté en tant que gestionnaire de contrôle de la serre, avec la possibilité d'envoyer des alertes en cas de dépassement de seuil.

4. Problèmes techniques

Comme dans tout projet, nous avons rencontré des difficultés techniques notamment lors de la réalisation du script en Bash, car nous ne savions quasiment rien de ce langage il a fallu dès lors apprendre la syntaxe du Bash.

Nous avons rencontré également une autre difficulté dans le sens où au départ le projet indiquait que le programme concernant les alertes devait s'effectuer en C++, ainsi nous avons configuré, installé QtCreator (prévu à cet effet afin de pouvoir coder le programme) sur le Raspberry pour pouvoir faire une compilation croisée. Or cela ne nous convenait pas, nous ne parvenions pas à réaliser ce programme ainsi notre choix s'est tourné sur le fait de coder le programme directement depuis la racine du Raspberry PI dans une question d'harmonie, de cohérence et surtout de meilleure compréhension.

E– Réalisation

I/ Réalisation personnelle : étudiant n°1, MICHAUD Théo

1. ControleurDeSerre

La classe ControleurDeSerre va nous permettre de nous connecter dans un premier temps, en Bluetooth, sur notre application.

```
#include "BluetoothSerial.h"
```

```
BluetoothSerial ESP_BT; //Object for Bluetooth
```

```
while (liaisonSmartphone.available() ) //Check if we receive anything from Bluetooth
{
    carlu = liaisonSmartphone.read();
    incoming += carlu;
}
```

Ensuite, on va comparer la chaîne de caractère pour pouvoir trouver sur quelle zone on doit agir :

```
switch (commande)
{
    case 'z':
        numZone = incoming.charAt(1) ;
        Serial.print("Zone ");
        Serial.print(numZone);
        Serial.print(" -> ");
        Serial.println(numZone - '1');
        zones[(int)(numZone - '1')]->Controler(incoming.substring(2));
        break;
    default:
        break;
}
```

La ligne suivante nous permet de faire appel à la méthode Controler(), pour pouvoir trouver dans quel mode on se trouve en ne prenant qu'à partir du 2e caractère de la chaîne de caractère.

```
zones[(int)(numZone - '1')]->Controler(incoming.substring(2));
```

On vide ensuite la chaîne de caractère.

Puis on rappelle la méthode Controler() ;

```
for(int indice=0; indice<4; indice++)
    zones[indice]->Controler(" " );
}
```

2. Classe Timer

La classe Timer va nous permettre de lancer des timers qui vont être l'équivalent des delay mais qui ont l'avantage de ne pas arrêter tout le système.

```
void Timer::Lancer(unsigned long _millisecondes)
{
    millisecondes = _millisecondes;
    valeurPrecedente = millis();
    actif = true;
    Serial.print("Temps avant impulsion : ");
    Serial.println(valeurPrecedente);
}

void Timer::Lancer(int _heures, int _minutes, int _secondes)
{
    millisecondes = (60 * ( 60 * _heures + _minutes) + _secondes) * 1000 ;
    valeurPrecedente = millis();
    actif = true;
}

bool Timer::Scruter()
{
    bool retour = false;
    if(actif && (millis() - valeurPrecedente >= millisecondes))
    {
        valeurPrecedente = millis();
        retour = true;
        if(unCoup)
            actif = false;
        Serial.print("Temps après impulsion : ");
        Serial.println(valeurPrecedente);
    }
    return retour;
}
```

3. Classe Vanne

La classe vanne permet de faire faire appel à deux méthodes Ouvrir() et Fermer() qui nous permettent de pouvoir envoyer une impulsion pour l'ouverture ou la fermeture d'une vanne.

Dans le .h de cette classe on va déjà définir les broches du Sens A et du Sens B :

```
#define SENS_A GPIO_NUM_16  
#define SENS_B GPIO_NUM_17
```

Ce qui nous permet ensuite dans le .cpp de cette classe de définir le mode de configuration des broches, en entrée ou en sortie.

```
gpio_set_direction(brocheImpulsion, GPIO_MODE_OUTPUT);  
gpio_set_pull_mode(brocheImpulsion, GPIO_PULLUP_ONLY);  
gpio_set_direction(sensA, GPIO_MODE_OUTPUT);  
gpio_set_direction(sensB, GPIO_MODE_OUTPUT);
```

Lorsque la méthode Ouvrir() est appelée dans le programme, c'est dans la classe Vanne qu'elle va s'exécuter. Elle va permettre d'envoyer une impulsion à la broche de sensA mais pas au sensB pour définir l'ouverture :

```
gpio_set_level(sensA, 1);  
gpio_set_level(sensB, 0);
```

Et ensuite, une impulsion sur la broche impulsion qui doit durer 250ms :

```
gpio_set_level(brocheImpulsion, 1);  
timerImpulsion.Lancer(250);
```

1. Ouvrir

Ce qui nous donne une méthode Ouvrir() comme celle-ci avec nos états qui varient :

```
void Vanne::Ouvrir()
{
    if(!mutex)
    {
        mutex = true;

        gpio_set_level(sensA, 1);
        gpio_set_level(sensB, 0);
        gpio_set_level(brocheImpulsion, 1);
        timerImpulsion.Lancer(250);
        etat = EN_OUVERTURE;
    }
    else
    {
        Serial.print("Impossible - ");
        Serial.println(brocheImpulsion);
        etat = EN_ATTENTE_OUVERTURE;
    }

}
```

2. Fermer

La méthode Fermer(), c'est le même principe sauf pour deux lignes du code, en effet pour changer de l'ouverture en fermeture l'impulsion de 250ms reste la même, il nous suffit d'inverser notre impulsion entre la broche SensA et SensB :

```
gpio_set_level(sensA, 0);
gpio_set_level(sensB, 1);
```

Et puis nos états changent aussi, si l'impulsion se fait alors notre état se transforme en en_fermeture :

```
etat = EN_FERMETURE;
```

Et si la fermeture n'est pas encore commencée alors :

```
etat = EN_ATTENTE_FERMETURE;
```

3. Contrôler

La méthode **Controler(char commande)** nous permet, grâce à l'état de notre vanne que l'on définit dans Ouvrir() et dans Fermer(), de contrôler l'état de nos vannes grâce à un switch case de l'état :

```
switch(etat)
{
    case EN_OUVERTURE:
        if(timerImpulsion.Scruter())
        {
            gpio_set_level(brocheImpulsion, 0);
            etat = REPOS;
            mutex= false;
            Serial.println("Ouvert");
        }
        break;
```

Pour le cas EN_FERMETURE, c'est la même chose que pour le cas EN_OUVERTURE :

```
case EN_FERMETURE:
if(timerImpulsion.Scruter())
{
    gpio_set_level(brocheImpulsion, 0);
    etat = REPOS;
    mutex=false;
    Serial.println("Fermé");
}
break;
```

Pour l'état REPOS, il va permettre d'ouvrir ou de fermer la vanne en fonction de la commande :

```
case REPOS:
if(commande=='O')
{
    Ouvrir();
}
if(commande=='F')
{
    Fermer();
}
break;
```

Et pour les cas EN_ATTENTE_OUVERTURE ou FERMETURE, cela va dépendre du mutex :

```
case EN_ATTENTE_OUVERTURE:
if(!mutex)
{
    Ouvrir();

}
else{
    Serial.println("En attente d'ouverture");
}
break;
case EN_ATTENTE_FERMETURE:
if(!mutex)
{
    Fermer();

}
else{
    Serial.println("En attente de fermeture");
}
break;
}
```

4. Classe Zone

La classe zone permet d'activer les différents modes selon le choix sur l'application.

Si la chaîne contient un ‘m’, alors le mode manuel s’enclenche :

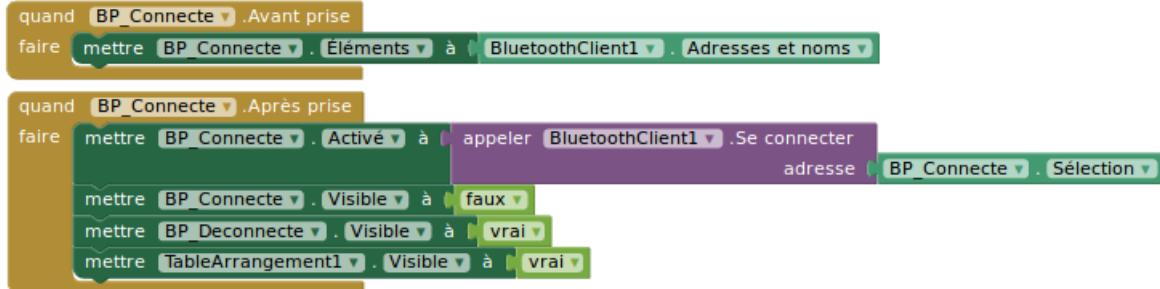
```
case 'm':  
    if(_parametre.charAt(1) == 'o')  
    {  
        laVanne.Controler('O');  
    }  
  
    if(_parametre.charAt(1) == 'f')  
    {  
        laVanne.Controler('F');  
    }
```

Si la chaîne contient un ‘p’, alors le mode programmé s’enclenche, si la chaîne contient un ‘a’, alors le mode automatique s’enclenche.

Le code n’étant pas fini, ces deux modes ne fonctionnent pas encore.

5. Réalisation de l'application

Grâce à App Inventor, seulement des blocs sont à déplacer pour programmer. Nous avons tout d'abord programmé la liaison Bluetooth :

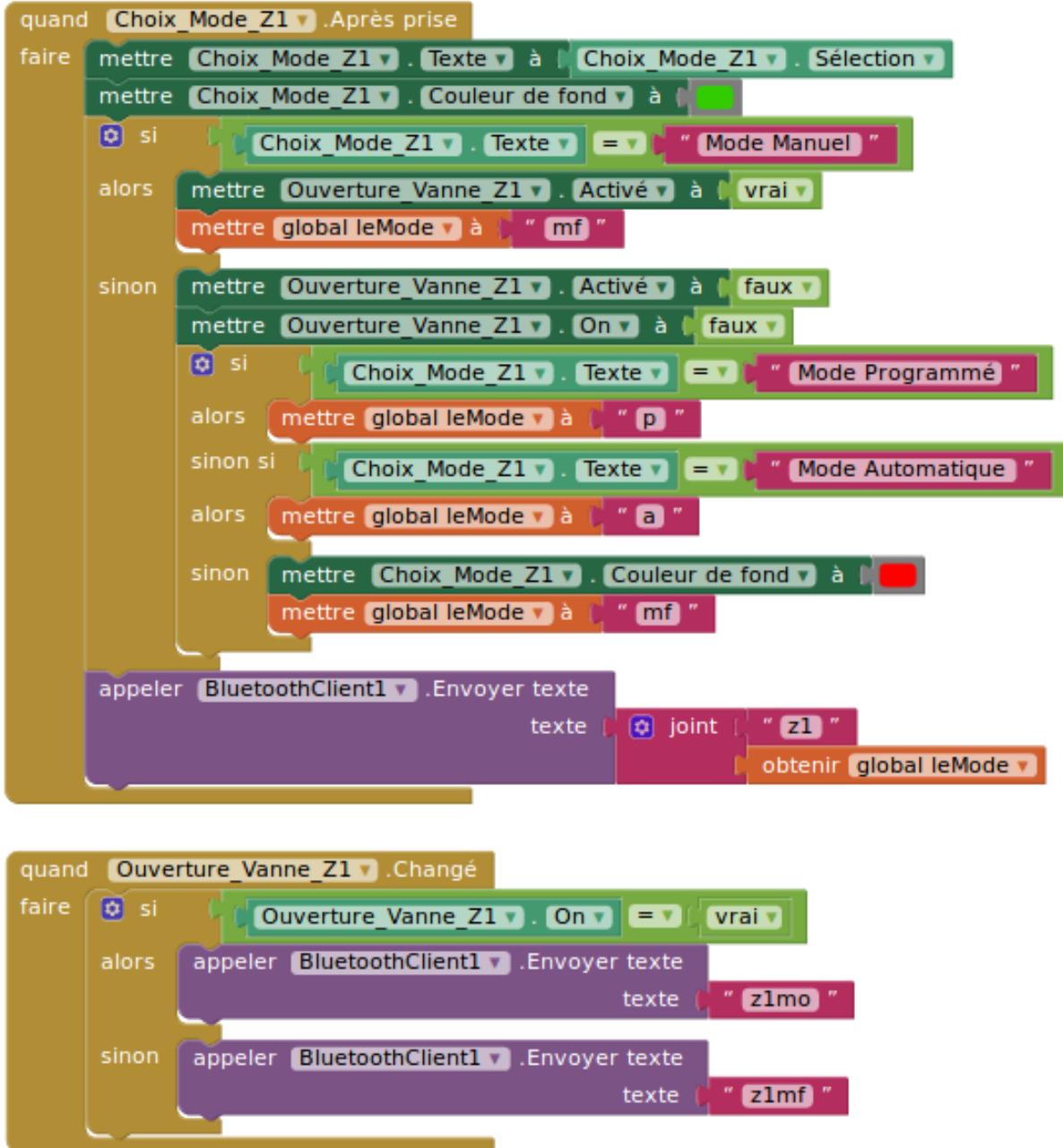


Lorsque nous cliquons sur le bouton connexion, tous les Bluetooth alentour sont répertoriés et affichés.

Puis lorsque l'on se connecte sur un Bluetooth, le bouton connexion se cache et le bouton déconnexion apparaît.

Ensuite, nous avons programmé les envois des chaînes de caractères lorsque l'on appuie sur un bouton pour changer de mode par exemple.

Dans un premier temps, nous avons travaillé sur une vanne seulement, puis nous avons dupliqué nos blocs de manière à gérer plusieurs vannes.



6. Conclusion

Pour le moment, toutes les classes présentées fonctionnent. En revanche, certaines ne sont pas complètes et il manque la classe débit d'eau. En effet, le mode programmé et le mode automatique doivent être finalisés, il faut donc obtenir l'humidité pour pouvoir gérer le mode automatique et aller chercher les seuils se situant sur la carte SD. Ce qui nous permettra de déclencher les vannes en fonction du taux d'humidité. Mais aussi pouvoir lire le fichier que se trouve sur une carte SD pour pouvoir obtenir les paramètres du mode programmé.

II/ Réalisation personnelle : étudiant n°2, AUVÉ Killian

1. Classe Capteur

1. Récupération des données météorologiques

Lors de l'exécution du programme, on va pouvoir faire appel aux 3 méthodes de récupération de données.

Pour pouvoir relever les données du capteur BME280, il faut inclure la bibliothèque associée :

```
#include <Adafruit_Sensor.h>
#include "Adafruit_BME280.h"
```

2. Constructeur de la classe

Ensuite pour pouvoir relever la température et l'hygrométrie, il faut instancier l'objet bme. On le fait dans le constructeur de la classe.

```
Capteur::Capteur() {
    bool status;
    status = bme.begin();
    if (!status) {
        Serial.println("Le capteur BME280 ne fonctionne pas" );
        while (1);
    }
}
```

Figure 1: Constructeur de la classe Capteur

Une fois l'objet instancié, nous n'avons plus qu'à faire appel à la méthode ReleverTemperature et ReleverHygrometrie.

3. ReleverTemperature

```
/**
 * @brief Capteur::ReleverTemperatureInterieur
 * @details Permet de relever la température du capteur bme280
 */
void Capteur::ReleverTemperature() {

    temperature = bme.readTemperature();

}
```

Figure 2: méthode ReleverTemperature()

Nous faisons appel à la méthode readTemperature() de la bibliothèque Adafruit sur l'objet bme pour obtenir la température. La donnée est ensuite mise dans la variable de la classe temperature.

4. ReleverHygrometrie

```
/**
 * @brief Capteur::ReleverHygrometrieInterieur
 * @details Permet de relever la hygrométrie du capteur bme280
 */
void Capteur::ReleverHygrometrie() {

    hygrometrie = bme.readHumidity();}

}
```

Figure 3: méthode ReleverHygrometrie()

Pour obtenir l'hygrométrie, nous faisons appel à la méthode readHumidity() sur l'objet bme pour obtenir l'hygrométrie. La donnée est ensuite mise dans la variable de la classe hygrometrie.

5. Inclusion de bibliothèque

Pour pouvoir relever la luminosité du capteur TSL2591, il faut inclure la bibliothèque:

```
#include "Adafruit_TSL2591.h"
```

6. ReleverLuminosite

Ensuite, nous faisons appel à la méthode ReleverLuminosite().

```
/**  
 * @brief Capteur::ReleverLuminositeInterieur  
 * @details Permet de relever la luminosité du capteur tsl2591  
 */  
void Capteur::ReleverLuminosite() {  
  
    luminosite = tsl.getLuminosity(TSL2591_VISIBLE);  
  
}
```

Figure 4: méthode ReleverLuminosite()

Nous n'avons pas besoin d'instancier l'objet tsl. Pour pouvoir obtenir la luminosité, nous faisons appel à la méthode getLuminosity.

Nous récupérons les données de l'étudiant n°1 liées au capteur de débit d'eau des vannes et d'humidité du sol afin de pouvoir aussi les envoyer au serveur.

2. Classe StockageDonnees

1. Envoi des données

Nous avons besoin d'inclure les bibliothèques suivantes pour pouvoir communiquer avec le serveur.

```
#include <WiFi.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <Ethernet.h>
```

Pour pouvoir envoyer des données, il faudra tout d'abord se connecter au réseau Wi-Fi à l'aide de la méthode **ConnexionWifi**

2. ConnexionWifi

```
/**
 * @brief StockageDonnees::ConnexionWifi
 * @details Permet de se connecter à un réseau wifi
 */
void StockageDonnees::ConnexionWifi() {
    while (!Serial); // wait for serial port to connect

    Serial.printf("\nConnexion à %s", loginWIFI);
    status = WiFi.begin(loginWIFI, mdpWIFI);
    Serial.println(status);
    delay(5000);
    if (status == WL_CONNECTED) {
        delay(500);
        Serial.printf("Pas de connexion au réseau %s. \n", loginWIFI);
        WiFi.begin(loginWIFI, mdpWIFI);
    } else {
        Serial.printf("Connecté au réseau %s. \n", loginWIFI);
    }
}
```

Figure 5: méthode ConnexionWifi()

On se connecte à un réseau Wi-Fi à l'aide de la ligne.

```
WiFi.begin(loginWIFI, mdpWIFI);
```

Les login et mot de passe Wi-Fi sont déjà définis dans la classe.

Ensuite, il faudra se connecter au serveur avec la méthode **ConnexionServeur()**

3. ConnexionServeur

```
/*
 * @brief StockageDonnees::ConnexionServeur
 * @details Permet de se connecter au serveur ou se trouve la base de données
 */
MySQL_Connection StockageDonnees::ConnexionServeur() {
    MySQL_Connection conn(&client);

    Serial.print("Connexion à la base de données... ");
    if (conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur)) {
        Serial.println("CONNEXION SERVEUR ETABLIE");
    } else {
        Serial.println("ECHEC CONNEXION SERVEUR");
    }
    return conn;
}
```

Figure 6: méthode ConnexionServeur()

La commande qui nous permettra de nous connecter est la suivante

```
conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur)
```

L'adresse IP, le login et le mot de passe du serveur sont définis dans la classe en attribut privé.

Cette méthode retournera l'objet conn qui nous permettra de faire des opérations sur la base de données.

Pour la suite du programme, nous avons besoin de récupérer les données des capteurs à l'aide des méthodes de la classe Capteur précédemment vue afin de les envoyer au serveur.

Ensuite une fois les données récupérées, nous avons besoin de les encapsuler dans une requête INSERT pour pouvoir les envoyer au serveur.

Pour encapsuler les données liées à la serre dans une requête INSERT, je vais utiliser la méthode **InsererDonneesMesuresInterieurSerre** qui va inclure les données dans une String et pour celle liée aux vannes, je vais utiliser la méthode InsererDonneesMesuresZone.

4. InsererDonneesMesuresInterieurSerre

Elles vont se charger de caster les variables (température, hygrométrie...) de type FLOAT en String et de les concaténer dans la requête SQL.

```
/**
 * @brief StockageDonnees::InsererDonneesMesuresInterieurSerre
 * @details Permet d'insérer les données météorologiques dans la requête sql
 */
void StockageDonnees::InsererDonneesMesuresInterieurSerre(float temperature, float hygrometrie, float luminosite){
    String chaineTemperature, chaineHygrometrie, chaineLuminosite;
    chaineTemperature = String(temperature);
    chaineHygrometrie = String(hygrometrie);
    chaineLuminosite = String(luminosite);

    now = rtc.now();
    String annee, mois, jour, heure, minute, seconde;

    annee = String(now.year());
    mois = String(now.month());
    jour = String(now.day());
    heure = String(now.hour());
    minute = String(now.minute());
    seconde = String(now.second());
    String date;
    date = annee + "-" + mois + "-" + jour + " " + heure + ":" + minute + ":" + seconde;

    requeteMesureInterieur = "INSERT INTO laSerre.MesuresSerreInterieur(temperatureInterieur,hygrometrieInterieur,
    luminositeInterieur,DateMesureInterieur,idSerre,idZone) VALUES (" + chaineTemperature + "," +
    chaineHygrometrie + "," + chaineLuminosite + "," + date + " , " + IDSERRE + ",1)";
    Serial.println(requeteMesureInterieur);
}
```

Figure 7: méthode InsererDonneesMesuresInterieurSerre()

Cette méthode permet de créer la requête d'insertion de donnée pour la table MesureSerreInterieur.

5. InsererDonneesMesuresZone

```
/**
 * @brief StockageDonnees::InsererDonneesMesuresZone
 * @param debitEau
 * @param humiditeSol
 * @details Permet d'insérer dans la requête SQL les données de débit d'eau, d'humidité
 */
void StockageDonnees::InsererDonneesMesuresZone(float debitEau, float humiditeSol) {

    String chainedebitEau, chainehumiditeSol;

    chainedebitEau = String(debitEau);

    chainehumiditeSol = String(humiditeSol);

    now = rtc.now();
    String annee, mois, jour, heure, minute, seconde;

    annee = String(now.year());
    mois = String(now.month());
    jour = String(now.day());
    heure = String(now.hour());
    minute = String(now.minute());
    seconde = String(now.second());
    String date;
    date = annee + "-" + mois + "-" + jour + " " + heure + ":" + minute + ":" + seconde;

    requeteMesureZone = "INSERT INTO laSerre.MesuresZone(debitEau,humiditeSol,dateMesure,idZone) VALUES
    (" + chainedebitEau + "," + chainehumiditeSol + "," + date + " , " + IDZONE + ")";
    Serial.println(requeteMesureZone);
}
```

Figure 8: méthode InsérerDonneesMesuresZone

Cette méthode permet de créer la requête d'insertion de donnée pour la table MesureZone.
Ensuite je vais envoyer les requêtes au serveur grâce à la méthode **EnvoyerDonneesBdd**.

6.EnvoyerDonneesBDD

```

void StockageDonnees::EnvoyerDonneesBdd(MySQL_Connection conn) {

    char INSERT_MESUREINTERIEUR[255];
    char INSERT_MESUREZONE[255];

    // creation du curseur mySQL
    cursor = new MySQL_Cursor(&conn);

    delay(1000);

    requeteMesureInterieur.toCharArray(INSERT_MESUREINTERIEUR, 255);
    requeteMesureZone.toCharArray(INSERT_MESUREZONE, 255);

    if (conn.connected()) {
        Serial.print("Connecté à la base de données");
        cursor->execute(INSERT_MESUREINTERIEUR);
        cursor->execute(INSERT_MESUREZONE);
    }
}

```

Figure 9: méthode EnvoyerDonneesBdd

Cette méthode permet d'exécuter la requête SQL précédemment créée, elle la convertit en tableau de caractère, car c'est le seul format que tolère l'envoi de requête.

On créer un curseur qui va nous permettre d'exécuter la requête SQL

Si la connexion est bien établie, on exécute le curseur avec la requête et elle est envoyée au serveur.

Toutes les opérations se répètent toutes les 15 minutes grâce au timer que l'on aura défini.

7. Création du Timer

```

//Création du timer fixé à 15 minutes
attente.Lancer(0, 15, 0);

```

Figure 10: Création du timer**8. Lancement du Timer**

```

    if (attente.Scruter())
    {

```

Figure 11: Lancement du timer

3. Stockage des données sur la carte SD

En cas de perte de connexion, les données récupérées seront stockées sur la carte SD. l'ESP32 communiquera avec la carte SD grâce au Shild SD.

L'ESP32 va vérifier que la connexion Wi-Fi et la connexion au Serveur sont toujours établies avec les méthodes suivantes.

1. VerifierConnexionWifi

```
/**
 * @brief StockageDonnees::VerifierConnexionWifi
 * @details Permet de se vérifier si l'ESP32 est toujours connecté au wifi
 */
void StockageDonnees::VerifierConnexionWifi() {
    // delay(500);
    if (status == WL_CONNECTED) {
        Serial.printf("Pas de connection au réseau %s. \n", loginWIFI);
        WiFi.begin(loginWIFI, mdpWIFI);
    }
}
```

Figure 12: méthode VérifierConnexionWifi

2. VerifierConnexionServeur

```
/**
 * @brief StockageDonnees::VerifierConnexionServeur
 * @param conn
 * @details Permet de se vérifier si l'ESP32 est toujours connecté au serveur
 */
void StockageDonnees::VerifierConnexionServeur(MySQL_Connection conn) {
    while (conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur) == false) {
        Serial.println("plus de connexion au serveur");
        conn.connect(serverAddr, 3306, loginUtilisateurServeur, mdpUtilisateurServeur);
        VerifierConnexionWifi();
    }
}
```

Figure 13: méthode VérifierConnexionServeur()

3. EnregistrerDonneesFichierLocal

Si la connexion est perdue, on enregistre les données sur la carte sd avec la méthode EnregistrerDonneesFichierLocal.

```
void StockageDonnees::EnregistrerDonneesFichierLocal(float temperature, float hygrometrie, float luminosite) {
    if (fichier == false) {
        OuvrirFichier();
    }

    String chaine, dates;

    dates = RecupereDate();
    chaine += temperature;
    chaine += ",";
    chaine += hygrometrie;
    chaine += ",";
    chaine += luminosite;
    chaine += ",";
    chaine += dates;

    //Serial.println(chaine);

    fichier.println(chaine);
    if (fermeture == true) {
        fichier.close();
    }
}
```

Figure 14: méthode EnregistrerDonneesFichierLocal

Cette méthode fait appel à la méthode OuvrirFichier() si le fichier n'est pas encore ouvert. Le fichier sera ouvert en écriture. Ensuite, on concatène toutes les données dans une chaîne de caractère et l'on enregistre les données dans le fichier texte sur la carte SD.

Une fois que la connexion sera rétablie, on fera appel à la méthode **LireDonneesFichierLocal()** afin de récupérer les données enregistrées sur la carte SD pour les envoyer au serveur.

4. Récupération du mode de programmation

Pour le bon fonctionnement de l'application Bluetooth, Théo Michaud a besoin de récupérer le mode de programmation de l'application. Pour cela je dois aller récupérer le mode programmation et aller l'enregistrer dans un fichier sur la carte SD.

4. Création de la base de données

Pour créer la base de données, nous avons fait un script SQL. Nous avons respecté le modèle de la base de données que l'on avait défini précédemment. Nous avons tout d'abord créé toutes les tables de la base de données avec l'opérateur CREATE TABLE.

Exemple de table créée en script SQL :

```
CREATE TABLE `Incident` (
    `idIncident` int(11) NOT NULL AUTO_INCREMENT,
    `alerteSMS` varchar(20) NOT NULL,
    `messageSMS` varchar(255) NOT NULL,
    `dateIncident` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `natureIncident` varchar(255) NOT NULL,
    `idZone` int(11) NOT NULL,
    `confirmationEnvoiMessage` varchar(20) NOT NULL,
    `confirmationLecture` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ensuite nous avons défini nos clés primaires de toutes les tables.

```
ALTER TABLE `Incident` ADD PRIMARY KEY (`idIncident`) ADD KEY `Incident_ibfk_1`(`idZone`);
```

Et pour finir, nous avons défini les clés étrangères de celle-ci.

```
ALTER TABLE `Incident` ADD CONSTRAINT `Incident_ibfk_1` FOREIGN KEY (`idZone`)
REFERENCES `Zone`(`idZone`);
```

Une fois le script créé, il a fallu l'importer dans la base de données.

Pour cela, nous avons tout d'abord créé une base de données.

Ensuite nous avons cliqué sur *Import*.

Nous avons cliqué sur *Parcourir*, et avons choisi le script SQL.

Importation dans la base de données «laSerre»

Fichier à importer :
Le fichier peut être comprimé (gzip, bzip2, zip) ou non.
Le nom du fichier comprimé doit se terminer par **[format].[compression]**. Exemple: **.sql.zip**
Parcourir : Aucun fichier sélectionné. (Taille maximum: 2 048Kio)
Vous pouvez également faire glisser et déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier :

Importation partielle :
 Permettre l'interruption de l'importation si la limite de temps configurée dans PHP est sur le point d'être atteinte. (Ceci pourrait aider à importer des fichiers volumineux, au détriment du respect des transactions.)
Ignorer ce nombre de requêtes (pour SQL), à partir du début :

Autres options :
 Activer la vérification des clés étrangères

Format :

Options spécifiques au format :
Mode de compatibilité SQL :
 Ne pas utiliser AUTO_INCREMENT pour la valeur zéro

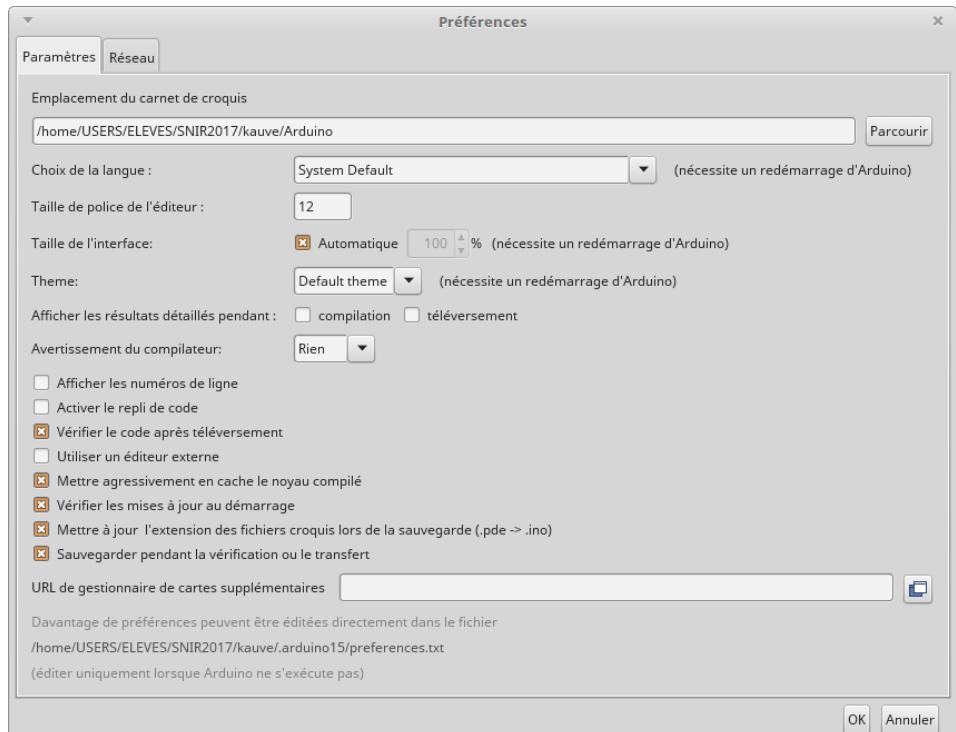
Et pour finir, nous avons appuyé sur *Exécuter*

5. Installation Arduino

Arduino va nous permettre de coder la carte ESP32, de définir son rôle et ses fonctions.

L'utilisation de l'ESP32 demande une installation particulière pour utiliser ses modules :

- Commencez par installer Arduino IDE
- Une fois Arduino lancé, allez dans **Fichier < Préférence**
- Puis relevez le répertoire et ouvrez un terminal dans celui-ci.



- Créer un répertoire « hardware » dans le dossier Arduino
- Puis déplacez-vous-y
- Créer un répertoire « expressif » dans le dossier hardware
- Puis déplacez-vous-y
- Télécharger le package ESP32 pour Arduino
- Déplacez-vous dans le dossier « esp32 »
- Mettez à jour les modules
- Déplacez-vous dans le dossier « tools »
- Lancer l'application des scripts en python
- Mettre « ESP32 Dev Module » dans le type de carte



6. Conclusion

Le système est bien capable d'effectuer les relevés de données météorologiques (température, hygrométrie, débit d'eau...) avec la date et l'heure, et de les envoyer au serveur afin de les enregistrer dans la base de données. Le système agit de manière automatique, c'est-à-dire qu'il est capable de continuer à fonctionner sans aide extérieur en cas d'incident (perte de connexion), il continue toujours à faire les relevés météorologiques et les enregistre dans un fichier local situé sur la carte SD afin de pouvoir les renvoyer quand la connexion sera revenue. Par contre, je n'ai pas eu le temps de réaliser la récupération du mode de programmation et des seuils dans la base de données nécessaire au bon fonctionnement de l'application.

III/ Réalisation personnelle : étudiant n°3, VILLERMIN Maxime

1. La page « Accueil »

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ma serre connectée</title>
        <link rel="shortcut icon" type="image/png" href="../img/carotte.png"/>
    </head>

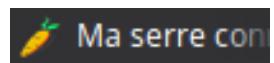
    <body>
        <?php require 'navBar.php'; ?>
        <div class="scroll-to-top d-lg-none position-fixed">
            <a class="js-scroll-trigger d-block text-center text-white rounded" href="#page-top">
                <i class="fa fa-chevron-up"></i>
            </a>
        </div>
        <?php require "piedPage.php"; ?>
    </body>
</html>
```

1. HTML

Pour expliquer simplement l'HTML, on écrit notre code en les balises <html> et </html>.

La partie <head> </head> accueillera :

- les scripts JavaScript.
- le titre du site que l'on peut voir dans l'onglet du navigateur web
- <link> : l'image du site



Le <body>, </body> est le corps du site, c'est ici qu'on y place nos éléments dans des <div>, </div>.

Le '<?PHP' sert à inclure des fichiers PHP dans notre page html.

Dans cet exemple on a inclus :

- 'navBar.php' qui est la barre de navigation du site
- 'piedPage.php' qui est le pied de page du site.

Copyright © Projet serre connectée SNIR 2019

Les classes contenues dans « class » sont disponibles sur le site ‘w3school’ et sur ce site, permettent de voir ce qu’elles produisent réellement. Ainsi j’ai pu placer mes éléments graphiques beaucoup plus rapidement et éviter de moi-même créer mes classes, choisir la disposition, etc.

2. SQL

Nous allons expliquer dans cette partie, l'utilisation du langage SQL avec quelques exemples de requêtes que nous avons utilisé et de comment cela fonctionne.

Dans toutes les fonctions que j'ai utilisées (dans le fichier 'fonctions.inc.php') et qui ont un rapport avec les requêtes SQL, j'ai utilisé :

```
$bdd = connexionBD();
```

On donne à la variable \$bdd le résultat de la fonction 'connexionBD() ;'. Mais d'où sort cette fonction ? On la retrouve dans le fichier 'connectionBD.php'.

```
//Variable indispensable à la connexion à la base de données.
define("SERVEURBD", "172.18.58.213");
define("LOGIN", "serre");
define("MOTDEPASSE", "Touchard72");
define("NOMDELABASE", "laSerre");

//fonction de connexion a la bdd
function connexionBD() {
    try {
        $bdd = new PDO('mysql:host=' . SERVEURBD . ';dbname=' . NOMDELABASE, LOGIN, MOTDEPASSE);
    } catch (Exception $ex) {
        die('<br/>Pb connexion serveur BD : ' . $ex->getMessage());
    }
    return $bdd;
}
```

On définit alors à la variable « SERVEURBD » l'adresse « 172.18.58.213 » là où se trouve ma base de données.

On défini à « LOGIN » « serre » qui est le login de connexion de ma base, « MOTDEPASSE » qui est le mot de passe et « NOMDELABASE » qui est « laSerre » c'est le nom de ma base de données.

Cette fonction remplit la variable '\$bdd' avec les valeurs définies, et crée un objet PDO qui représente une connexion à la base. En cas d'erreur, la fonction retourne 'Pb connexion serveur BD' avec la raison de l'erreur.

La fonction retourne donc '\$bdd'.

Ensuite, intéressons-nous au fichier ‘fonctions.inc.php’ qui contient les fonctions qui interagissent via des requêtes avec la base de données.

Ce fichier contient alors :

Nom de la fonction	Description
getNomSerre()	récupère le nom de la serre
getNomZone(\$idSerre)	récupère le nom des zones par rapport à l’idSerre renvoyé lors de la sélection du selec
recuperationDonneesTemperatureInterieur()	récupère les données de température intérieure
recuperationDonneesTemperatureExterieur()	récupère les données de température extérieure
recuperationDonneesLuminosite()	récupère les données de luminosité
recuperationDonneesHygrometrie()	récupère les données d’hygrométrie de l’air
getResponsable()	récupère la liste des responsables
getData(\$idUtilisateur)	récupère les données des responsables par rapport à l’id de l’utilisateur que renverra le <select> lorsqu’on choisi un utilisateur
ajouterUtilisateur(\$nom, \$prénom, \$numTel, \$mail)	ajoute un utilisateur à la base en prenant en paramètre le nom, le prénom, le numéro de téléphone et l’adresse mail
modifierUtilisateur(\$id, \$nom, \$prénom, \$email, \$numTel)	modifie un utilisateur dans la base en prenant en paramètre le nom, le prénom, le numéro de téléphone et l’adresse mail
getDataZone(\$idSerre)	récupère les données des zones par rapport à la serre sélectionnée
modifierValeurSerre(\$idSerre, \$seuilMinTempInt,\$seuilMaxTempInt, \$seuilMinLum,\$seuilMaxLum, \$seuilMinHygroAir,\$seuilMaxHygroAir)	modifie les valeurs des différents seuils de la serre

On va voir maintenant différentes syntaxes que j'ai utilisées :

J'ai utilisé le ‘**select**’ pour récupérer des données, ‘**insert into**’ pour insérer des données, ‘**update**’ pour mettre à jour des données, ‘**delete**’ pour supprimer des données.

Exemple pour `getNomSerre()` : ‘**select idSerre, nomSerre from Serre;**’

→ on sélectionne idSerre et nomSerre dans la table Serre.

Exemple pour `ajouterUtilisateur($nom, $prénom, $numTel, $mail)` : ‘**INSERT INTO Utilisateur(`nom`, `prénom`, `numTel`, `mail`) VALUES(:nom,:prenom,:numTel,:mail);**’

→ on insère dans la table Utilisateur (‘nom’...) qui prend comme valeur (:nom,...)

Exemple pour `modifierUtilisateur($id, $nom, $prénom, $e-mail, $numTel)` : ‘**UPDATE Utilisateur set `nom`=:nom , `prénom`=:prénom , `numTel`=:numTel , `mail`=:mail WHERE idUtilisateur=:id;**’

→ on met à jour la table Utilisateur avec ‘nom’ = ‘valeur qu’a le nouveau nom’...

Exemple pour `supprimerUtilisateur($id, $nom, $prénom, $e-mail, $numTel)` :

Nous n'avons pas encore codé cette requête, mais elle sera sous la forme : ‘**DELETE Utilisateur WHERE condition**’

→ on supprime de la table Utilisateur la condition

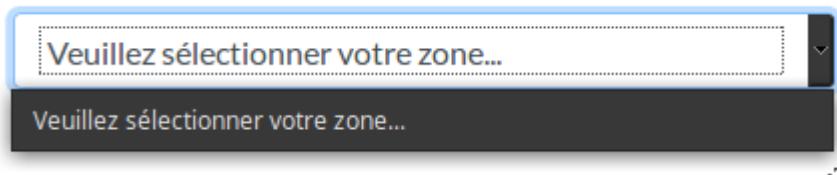
2 La page « Visualiser »

1.genereNomSerre() ;

Dans mon fichier ‘visualiser.php’, on a le code HTML, en voici une partie :

```
<!--Select pour selectionner la serre-->
<div class="col-lg-6">
    <select class="form-control" id="selectionSerre" name="selectionSerre">
        </select>
</div>
<!--Select pour selectionner la zone à visionner par rapport à la serre-->
<div class="col-lg-6">
    <select class="form-control" id="selectionZone" name="idZone">
        <option value="-1">Veuillez sélectionner votre zone...</option>
    </select>
</div>
<input type="hidden" name="typeDemande" id="typeDemande" value="r" />
```

Dans cette partie on a créé un `<select>` (liste déroulante), qui prend la classe ‘col-lg-6’. C’est une classe bootstrap qui met en forme mon select comme dans les IHMs vus précédemment. On a le premier select qui a pour id ‘selectionSerre’. Le second select, même classe, a pour id ‘selectionZone’. On lui a fait passer comme premier élément de la liste (valeur ‘-1’) une option qui affichera ‘Veuillez sélectionner votre zone...’



On a un `input` de type ‘hidden’ (caché) qui a pour id ‘typeDemande’ et comme valeur ‘r’.

Cette ligne envoie un typeDemande avec la valeur ‘r’. Dans le switch case situé dans mon ‘contrroleur.php’ on a :

```
$source = $_GET['typeDemande'];

switch ($source) {
    case 'r': getNomSerre();
    break;
    case 'd': getNomZone($_GET['selectionSerre']);
    break;
}
```

la case ‘r’ on fait appel à la fonction ‘getNomSerre() ; »

On voit que pour

```

function getNomSerre() {
    $bdd = connexionBD();
    $requete = $bdd->prepare("select idSerre, nomSerre from Serre");
    $requete->execute() or die(print_r($requete->errorInfo()));
    $tabnomSerre = array();

    while ($tab = $requete->fetch()) {

        // ajout d'une case dans le tableau
        array_push($tabnomSerre, array(
            'idSerre' => $tab['idSerre'],
            'nomSerre' => $tab['nomSerre']
        ))
    }
    $requete->closeCursor();
    //on previent qu'on repond en json
    header('Content-Type: application/json; charset=utf-8');
    // envoyer les données au format json
    echo json_encode($tabnomSerre);
}

```

Cette fonction récupère alors l'idSerre et le nomSerre dans la table Serre. Le range dans un tableau (array()). Pour chaque valeur récupérée, on les change dans le tableau. On ferme la requête. On change le content-type qui sert à préciser au client quel type de données il va devoir traiter dans la réponse du serveur (générée par PHP), ici du json et met au format json le contenu du tableau.

Les données sont alors sous cette forme (en json) :

```

▼ 0: {...}
  idZone: 2
  nomZone: Carotte
▼ 1: {...}
  idZone: 4
  nomZone: Poireaux
▼ 2: {...}
  idZone: 3
  nomZone: Potiron
▼ 3: {...}
  idZone: 1
  nomZone: Tomate

```

Voici la fonction ‘genereNomSerre()’ ; cette fonction permet de générer le nom des serres depuis la base de données.

```

2 function genereNomSerre()
3 {
4     $("#typeDemande").val('r');
5     //on vide la liste
6     $("#selectionSerre").empty();
7     $("#selectionSerre").append('<option value="-1">Veuillez sélectionner votre serre </option>');
8
9     $.ajax({
10         url: '../php/controleur.php',
11         data: $("#typeDemande").serialize(),
12         type: 'GET',
13         dataType: 'json',
14         success:
15             function (objetJson) {
16
17                 $.each(objetJson, function (index, ligne) {
18                     $("#selectionSerre").append('<option value="' + ligne.idSerre + '">' + ligne.nomSerre + '</option>');
19                 });
20             },
21         error:
22             function (xhr, status, error) {
23                 console.log("param : " + JSON.stringify(xhr));
24                 console.log("status : " + status);
25                 console.log("error : " + error);
26
27             }
28         });
29     });
30 }
31 
```

Elle est appelée dans la partie « Visualiser » du projet. Voici l’explication du fonctionnement prévu à cet égard :

Ligne 4 : on a du Jquery → \$().val() ; typeDemande à pour valeur ‘r’

Ligne 6 : on vide la liste selectionSerre

Ligne 7 : on ajoute en valeur ‘-1’ : la première de la liste, une option qui aura comme texte « Veuillez sélectionner votre serre »

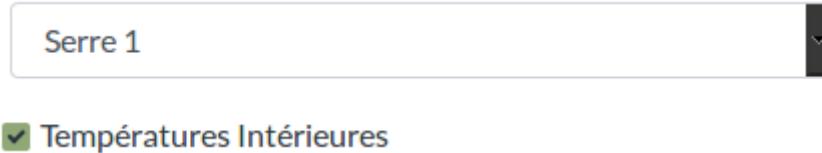
ensuite on a de l’Ajax. On donne l’adresse de mon contrôleur, on sérialise la valeur de type demande ('r') et on indique que le type de donnée est du 'json'. Si tout ce passe bien, pour chaque élément du tableau json, on remplit le nom de la serre dans la liste déroulante, on indique que chacune des valeurs (nom des serres) prendra comme id son id associé à la base de données. S'il y a une erreur, on aura des indications sur celle-ci.

Avec tout cela, on peut remplir correctement mes listes de serres. Pour mettre à jour la liste de mes zones par rapport à la serre sélectionnée, c'est le même principe. On envoie une valeur de typeDemande, le contrôleur l'interprète et fait appel à la fonction getNomZone, sauf qu'il prend en paramètres l'id de la serre sélectionnée pour ainsi sélectionné uniquement les zones de la serre correspondante.

2. La partie graphique

Nous nous trouvons sur la page de visualisation. Nous rappelons que sur cette page, l'utilisateur va pouvoir choisir les grandeurs qu'il souhaite visualiser sur un graphique. Il commence par choisir une serre parmi la liste.

Nous allons donc procéder avec le choix de la température intérieure, car l'utilisation et la création d'un graphique sont la même pour chacune des grandeurs.



J'ai donc créé un fichier dans le dossier 'PHP' qui s'appelle 'graphique.js'. C'est dans ce fichier que la création de mes graphiques se fera.

Il faut donc inclure mon fichier dans la page « visualiser.php » entre les balises <head></head>.

```
<script src="../js/graphique.js"></script>
```

Voici le code de la fonction afficherGraphTemperatureInterieur() ;

```

var seriesOptions = [],
    seriesCounter = 0,
    names = ['Température (en °C)', 'Hygrométrie (en %)', 'Luminosité (en Lux)'];

//Fonction graphique température
function afficherGraphTemperatureInterieur()
{
    $.getJSON('../php/contrôleur.php', 'commande=graphTemperatureInt', function (data, status) {
        // Create the chart
        console.log("ok");
        console.log("stat:" + status);

        seriesOptions[0] = {
            name: names [0],
            data: data,
            color: '#BB0000'
        };

        Highcharts.stockChart('graphTemperatureInt', {
            chart: {
                zoomType: 'xy'
            },
            title: {
                text: 'Courbe de températures intérieures en °C',
                style: {
                    color: 'black',
                    fontSize: '16px',
                    fontWeight: 'bold'
                }
            },
            yAxis: [
                {
                    labels: {
                        format: '{value} °C'
                    },
                    title: {
                        text: 'Températures'
                    }
                }],
            rangeSelector: {
                selected: 4
            },
            series: seriesOptions
        });
    });
}

```

Dans les 3 premières lignes, nous définissons un tableau serieOption qui contient 3 types de valeurs.

Pour la création de mon graphique, nous le mettons dans la fonction. Chaque grandeur à une fonction associée. On indique qu'on lui insère du JSON, nous lui donnons l'adresse du contrôleur pour envoyer ‘commande’, la commande qui sera envoyée.

La ligne Highcharts.stockChart(‘graphTemperatureInt’ → permet de nommer le graphique soit ‘graphTemperatureInt’. Le reste du code est des options qui définissent, le style, le titre, les informations affichées sur un point.

Dans le contrôleur :

Si \$commande = 'graphTemperatureInt' on fait appel à la fonction recuperationDonneesTemperatureInterieur() .

```
$commande = $_GET['commande'];

if ($commande == "graphTemperatureInt") {
    recuperationDonneesTemperatureInterieur();
}

if ($commande == "graphTemperatureExt") {
    recuperationDonneesTemperatureExterieur();
}

if ($commande == "graphLuminosite") {
    recuperationDonneesLuminosite();
}

if ($commande == "graphHygrometrie") {
    recuperationDonneesHygrometrie();
}
```

Dans le fichier 'fonctions.inc.php', on a la fonction qui récupère les données de la température intérieure via une requête SQL :

```
function recuperationDonneesTemperatureInterieur() {
// Si tout va bien, on peut continuer
    $bdd = connexionBD();
// On récupère tout le contenu de la table MesureSerreInterieur
    $reponse = $bdd->prepare("SELECT * FROM TemperatureHumidite ORDER BY DateMesureInterieur ");
    $reponse->execute() or die(print_r($reponse->errorInfo()));

    $nbLigne = $reponse->rowCount();

    if ($nbLigne == 0) {
        echo "Il n'y a pas de données correspondants.";
    }
    if ($nbLigne > 0) {
        // création du tableau
        $donneesJson = array();

        while ($tab = $reponse->fetch()) {
            $date = new DateTime($tab['DateMesureInterieur']);
// ajout d'une case dans le tableau
            array_push($donneesJson, array(
                // $tab['DateMesureInterieur'],
                $date->getTimestamp() * 1000,
                $tab['temperatureInterieur'],
                )
            );
        }

        // on prévient qu'on répond en json
        header('Content-Type: application/json; charset=utf-8');
        // envoyer les données au format json
        echo json_encode($donneesJson, JSON_NUMERIC_CHECK);
    }
    $reponse->closeCursor();
}
```

Même principe que pour la récupération du nom des serres, on récupère avec la requête et on convertit en JSON. Tout le principe de création de graphiques, du contrôleur et de la requête est le même pour chaque donnée que l'on veut récupérer et afficher (données de température, utilisateurs, seuils, etc.). La différence sera ce que l'on veut récupérer, et le nom que l'on donne aux variables du contrôleur.

Problème pour le graphique :

Le graphique Highcharts attendait des valeurs en JSON. Il attendait aussi la date au format 'Timestamp'. Qu'est-ce que le timestamp (Unix)? Il désigne le nombre de secondes écoulées depuis le 1^{er} janvier 1970 à minuit UTC précise.

Les avantages sont multiples :

- facilite la comparaison de date, puisque ça revient à faire une différence entre 2 nombres
- universels, puisque pas de notion de fuseaux horaires

quelques inconvénients :

- le nombre en lui-même n'est pas parlant pour un humain, obligé de passer par un convertisseur
- ne gère pas (nativement) avant 1970, donc pour gérer les dates de naissance par exemple, ce n'est pas l'idéal

J'ai passé plusieurs jours à me rendre compte de cela, car nous pensions qu'il attendait une date au format classique 'DATETIME'.

Pour résumer, le comparatif MySQL Date et Time Types :

- Un DATETIME occupe 8 octets alors qu'un TIMESTAMP n'en occupe que 4.
- L'intervalle des valeurs légales d'un TIMESTAMP est [1970; 2037] alors qu'un DATETIME peut être compris entre '1000-01-01 00:00:00' et '9999-12-31 23:59:59'.
- Seul le TIMESTAMP permet l'initialisation et la mise à jour automatique d'un champ à la date et heure courante.

Après les deux diffèrent assez à l'utilisation puisqu'on ne peut pas afficher directement un TIMESTAMP, il faut le convertir, on peut par contre considérer que le format d'affichage 'YYYY-MM-DD HH:MM:SS' du DATETIME est "lisible".

Dans la base de données, les dates sont au format DATETIME, sauf que le graphique les veut en TIMESTAMP. Avec la méthode 'getTimestamp', nous pouvons convertir la date au format TIMESTAMP. J'ai ajouté '*1000', car il manquait 3 unités à mes dates.

```
$date->getTimestamp() * 1000,
```

3 L'authentification

L'utilisateur devra se connecter lorsqu'il voudra accéder à la page « Programmer ». Un modal de connexion, comme vue précédemment, sera visible lors du clique sur cet onglet.

Nous avons donc choisi de gérer cette authentification avec un fichier « .htaccess ».

1. Configuration du fichier .htaccess :

```
AuthName "Veuillez vous identifier"  
AuthUserFile /etc/.htpasswd  
AuthType Basic  
require valid-user
```

La première ligne est le message affiché lors de la connexion de l'utilisateur.

La seconde indique le chemin du fichier .htpassword qui contiendra les mots de passe valide pour la connexion. La directive AuthType définit la méthode utilisée pour authentifier l'utilisateur. La méthode la plus courante est Basic. Il faut cependant garder à l'esprit que l'authentification Basic transmet le mot de passe depuis le client vers le serveur en clair. Cette méthode ne devra donc pas être utilisée pour la transmission de données hautement sensibles ce qui n'est pas notre cas.

Enfin, la directive Require implémente la partie autorisation du processus en définissant l'utilisateur autorisé à accéder à cette zone du serveur (utilisateur valide).

2. Configuration du fichier .htpassword :

Ce fichier sert à lister les différents utilisateurs valides, avec le login et le mot de passe. Ce fichier est placé dans le même dossier que le fichier '.htaccess'.

Voici un exemple :

```
|toto:$1$w6ZTi8pm$0Gg0/Vt0D6mkuUWBSJmSR.
```

La norme d'écriture pour ce fichier est login:motdepasse.

Le login est 'toto' et le mot de passe est 'toto'. On voit ici que le mot de passe est sous forme de caractères différents. Eh oui, le mot de passe est crypté. J'ai trouvé en ligne un générateur de .htpassword, qui offre une interface où on a plus cas mettre le login et le mot de passe et le site nous renvoie l'identifiant valide pour le format .htpassword avec le mot de passe crypter.

Par défaut les fichiers .htaccess ne sont pas activé avec apache. Pour les activer, et donc pouvoir les utiliser, il faut le faire par le terminal du serveur web:

Activer les fichiers .htaccess :

Editer le fichier : /etc/apache2/sites-enabled/000-default

sudo nano /etc/apache2/sites-enabled/000-default

Mettre ces lignes à la fin du fichier :

<Directory /var/www/le_rep_ou_vous_voulez_activer_le_.htaccess>

AllowOverride all

</Directory>

Enregistrez le fichier

4 Partie « Programmer »

Dans cette partie, on peut donc gérer les utilisateurs (ajout, modification, suppression), configurer les seuils d'alertes (hygrométrie, températures, etc.) et configurer l'arrosage (le mode, les dates de début, associé un responsable, etc.). La création du code et de mes fonctions reste la même pour chacune des tâches. C'est-à-dire que : on envoie au contrôleur une valeur, il vérifie la valeur, si elle existe il appelle la fonction associée à cette valeur. Cette fonction utilise les requêtes SQL pour aller chercher dans la base de données ce qu'on lui demande et les retourne en JSON. Puis avec l'un des fichiers JAVASCRIPT qui sont composés de fonction, on interprète les valeurs reçues pour les afficher sur le site web.

1.Gestion des boutons et des champs du site

Seuil de température intérieure

Mini : °C	<input type="button" value="^"/>	<input type="button" value="v"/>	Maxi : °C	<input type="button" value="^"/>	<input type="button" value="v"/>
-----------	----------------------------------	----------------------------------	-----------	----------------------------------	----------------------------------

Seuil de luminosité

Mini : Lux	<input type="button" value="^"/>	<input type="button" value="v"/>	Maxi : Lux	<input type="button" value="^"/>	<input type="button" value="v"/>
------------	----------------------------------	----------------------------------	------------	----------------------------------	----------------------------------

Seuil d'hygrométrie de l'air

Mini : %	<input type="button" value="^"/>	<input type="button" value="v"/>	Maxi : %	<input type="button" value="^"/>	<input type="button" value="v"/>
----------	----------------------------------	----------------------------------	----------	----------------------------------	----------------------------------

Pour la partie des configurations des seuils, j'ai choisi de mettre des boutons de type :

```
<input type="number" name="seuilMinLum" placeholder="Lux" >
```

Le <input> signifie que c'est une entrée de données dans le système informatique, c'est-à-dire ce que l'utilisateur veut donner à la machine. Chacun de ses boutons contient un nom et type 'number' qui correspond à des nombres. Le 'placeholder' correspond à ce qui est affiché en fond dans le champ (sert à indiquer le type de valeurs attendu : pour luminosité des lux, pour l'hygrométrie des %...)

```
<input type="number" name="seuilMinHygroAir" placeholder "%" min="0" max="100" >
```

Pour les seuils d'humidité on a une valeur min et max. J'ai choisi ses valeurs, car l'hygrométrie et un pourcentage, donc compris entre 0 et 100. L'utilisateur ne pourra donc pas saisir une valeur supérieure à 100 et inférieure à 0. Ces paramètres sont valables pour les 4 zones de la serre pour l'hygrométrie du sol et pour l'hygrométrie de l'air de la serre.

Pour la partie configuration de l'arrosage, on a :

Zone 1

Mode de programmation :

Date de l'arrosage

Début : Heure Minutes
 Périodicité : Durée :

Zone 2

Mode de programmation :

Date de l'arrosage

Début : Heures Minutes
 Périodicité : Durée :

Zone 3

Mode de programmation :

Date de l'arrosage

Début : Heures Minutes
 Périodicité : Durée :

Zone 4

Mode de programmation :

Date de l'arrosage

Début : Heures Minutes
 Périodicité : Durée :

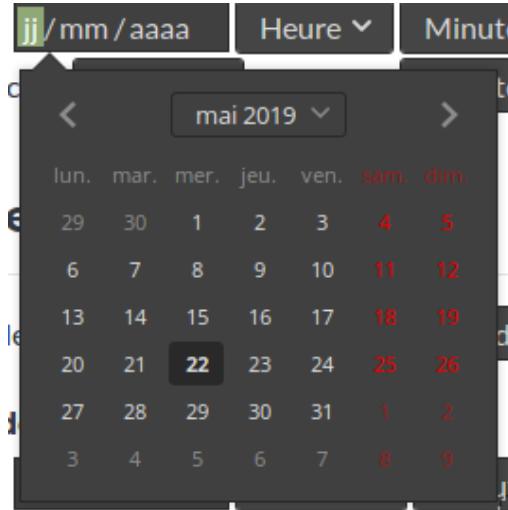
Pour le mode de programmation, on a un type <select> (liste déroulante) avec comme option le choix des modes:

```
<select name="modeZ4" id="modeZ4">
    <option value="-1">Choisissez le mode ...</option>
    <option value="00">Mode automatique</option>
    <option value="01">Mode manuel</option>
    <option value="02">Mode programmé</option>
</select>
```

Le date début est sous forme

```
<input type="date" name="dateDebutZone3" id="dateDebutZone3">
```

C'est un input de type 'date'. Ce type date permet à l'utilisateur de saisir l'année, le mois, le jour de début d'arrosage. Par défaut, la date sélectionnée est celle du jour.

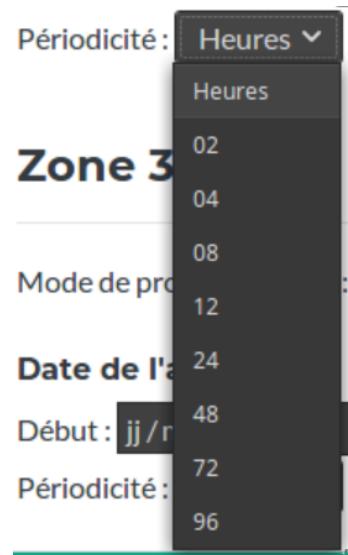


L'heure et la minute de début sont aussi des <select> avec comme valeur :

- Pour les heures : de 0 à 23 h
- Pour les minutes : de 0 à 55 avec un pas de 5

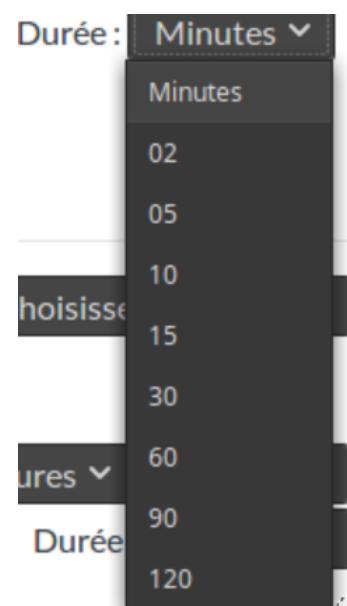
La périodicité, aussi un select. La périodicité peut se traduire par : « On débute l'arrosage tout les... »

Nous avons choisi une périodicité de :



Le temps d'arrosage (durée), aussi un select. La durée peut se traduire par : « On arrose pendant... minutes»

Nous avons choisi une durée de :



5. Conclusion

Pour conclure, le projet m'apporte de nouvelles connaissances sur de nouvelles technologies pour le développement web comme Boostrap et me permet de renforcer et mettre en pratique les différentes connaissances acquises pendant ma période de formation comme l'HTML ou l'Ajax pour n'en citer que deux. Le projet me permet de voir aussi les différentes difficultés auquel on peut faire face lors de la participation à un projet d'équipe de cette envergure.

Ayant rencontré plusieurs difficultés sur la récupération de certaines données (pour les graphiques par exemple), ou pour les mises à jour de mes listes en Ajax, j'ai été retardé sur l'avancement du projet.

Il reste donc encore beaucoup de travail à faire pour aboutir à la demande de ce projet.

Lors de la rédaction de ce dossier, quelques parties du projet n'ont pas encore été traitées, notamment :

- Dans la partie visualiser : Associer les données des zones par rapport à la serre sélectionnée, pour pouvoir gérer l'affichage des graphiques sachant que l'utilisateur peut afficher certaines données globales pour la serre et d'autres données uniquement pour les zones associées à cette serre
- Dans la partie configuration d'un utilisateur : Coder la partie « modifier » et « supprimer » ainsi qu'ajouter un « <select> » pour associer un responsable à une serre lors de la création d'un utilisateur
- Dans la partie météo : préremplir tous les champs lorsqu'une serre est sélectionnée, car il y a quelques problèmes et lorsque l'on clique sur valider toute les valeurs contenues dans le champ doivent être enregistrer dans la base de données
- Pour la partie arrosage même chose : préremplir les champs et enregistrer les valeurs remplies dans les champs
- Enfin si le temps nous le permet, un système d'alerte sur le site web lorsqu'un seuil est dépassé avec la confirmation de lecture pour éviter que le message soit renvoyé toutes les 15 minutes.

IV/ Réalisation personnelle : étudiant n°4, ROCHER Marvyn

1. Réalisation du programme permettant d'envoyer des SMS

Une fois que l'utilitaire Gammu a été installé, nous pouvons dès lors nous atteler à l'élaboration du programme principal concernant l'étudiant n°4.

Ce programme est uniquement codé en script Bash notamment pour des raisons pratiques.

Pour faire le programme, plusieurs points ont été retranscrits :

- Pouvoir se connecter à la base de données depuis le programme principal
- Pouvoir lire les données présentes dans la base de données
- Pouvoir lire les seuils qui sont présents dans la base de données
- Pouvoir comparer les valeurs présentes dans la base de données et les seuils associés
- Pouvoir gérer l'envoi de SMS de manière automatique lorsqu'une donnée n'est pas comprise dans les seuils préalablement définis.

Ainsi grâce à tous ces points, le code se compose de plusieurs parties qui correspondent aux points retranscrits au préalable.

- La connexion à la base de données
- La lecture des données
- L'obtention des seuils
- Les tests de comparaisons et l'envoi de SMS

1.1 Connexion à la base de données

Afin de pouvoir nous connecter à la base de données, nous avons opté pour la solution qui est d'inscrire directement les paramètres dans des variables afin de ne pas avoir à retaper lors du code les informations de la base de données, mais juste le nom des variables affectées : l'adresse IP, le nom de la base de données, les identifiants de connexion.

```
hostname="172.18.58.213"
dataBaseName="LaSerre"
userNome="serreUtilisateur"
password="Touchard72"
```

Affectation des paramètres dans des variables

1.2 Lecture des données

Afin de pouvoir lire les données, il nous a fallu savoir quelles sont les données à recueillir :

- L'hygrométrie (*intérieure*) exprimée en %
- La luminosité (*intérieure*) exprimée en lux
- La température (*intérieure*) exprimée en °C
- Le débit d'eau exprimé en L/min
- L'humidité exprimée en %

Comme nous venons interroger la base de données configurée (MariaDB), c'est par le biais de requêtes du langage SQL qu'une interaction est possible entre le serveur et le programme.

Le problème se posant étant le suivant : comme toutes les mesures sont actualisées toutes les 15 minutes il faut pouvoir avoir accès et pouvoir lire la dernière valeur enregistrée sur la base de données, cela se traduit très bien dans le langage SQL.

Ainsi voici une requête type que j'ai écrite afin de pouvoir lire la dernière donnée de l'hygrométrie intérieure :

```
SELECT hygrometrieInterieur  
FROM laSerre.TemperatureHumidite  
ORDER BY DateMesureInterieur  
DESC LIMIT 1 ;
```

Si nous voulons commenter cette requête, nous pouvons voir que nous sélectionnons le champ « hygrometrieInterieur » dans la table « TemperatureHumidite » qui est présente dans la base Serre

On choisit de ranger les données du champ HygrometrieInterieur par date la plus récente, mais nous classons ces valeurs par ordre de date la plus récente afin d'obtenir la dernière valeur comme souhaité lors du cahier des charges.

1.3 Obtention des seuils

Après avoir recueilli les données, il nous a fallu savoir quels sont les seuils qui sont nécessaires au bon fonctionnement du programme, quels sont les seuils à recueillir.

- Seuil minimum et maximum d'hygrométrie
- Seuil minimum et maximum de température
- Seuil minimum et maximum de luminosité
- Seuil maximum de débit d'eau
- Seuil minimum et maximum d'humidité

Afin de vouloir obtenir les seuils, nous avons opté pour la même manière que les relevés, par des requêtes SQL.

1.4 Concaténation des chaînes

Dans un souci d'optimisation de code, il nous a paru logique de, pour ce programme, concaténer des chaînes de caractères afin d'avoir un minimum de variables.

Cette concaténation de chaînes de caractères en Bash fut une nouveauté à part entière pour notre part.

Cette concaténation de chaînes nous permet de condenser justement deux requêtes SQL en une seule.

L'exemple que nous avons choisi de développer est celui de l'hygrométrie.

En effet, l'hygrométrie possède deux seuils : un minimum et un maximum. Pour éviter de créer deux variables, nous avons choisi de créer une seule variable qui se nomme hygrométrie ; dans laquelle nous cherchons le minimum et la maximum.

Cela donne :

```
SELECT minHygrometrie, maxHygrometrie  
FROM laSerre.Serre  
ORDER BY idSerre  
ASC LIMIT 1 ;
```

Ainsi si l'on exécute cette requête, nous obtenons deux valeurs, le but de la concaténation de chaînes est de pouvoir les séparer et prendre uniquement le minimum quand il en est nécessaire et le maximum également.

Afin de se faire, la concaténation de chaînes est simple il suffit de connaître :

- La longueur de la chaîne
- Son point de départ

Avec ces deux informations, nous sommes en mesure d'effectuer la concaténation.

Toujours pour l'exemple de l'hygrométrie, voici ce que donne la concaténation :

```
"${hygrometrie:0:2}"
```

Comme dit précédemment, nous avons besoin de la longueur de la chaîne; or le seuil minimum de l'hygrométrie étant de 70, la chaîne de caractères possède donc 2 caractères.

Comme nous voulons le minimum, et que dans la requête SQL, nous avions demandé le minimum en premier, la position de départ est de 0.

Ainsi si nous voulions traduire cette concaténation cela donnerait :

- Partir de 0
- Aller jusqu'au deuxième caractère

- S'arrêter

Nous récupérons donc bien la valeur qui est le minimum de l'hygrométrie.

Pour le maximum cela fonctionne de manière similaire, hormis qu'au lieu de commencer à 0 nous commençons à 3 en effet, car dans la requête SQL, lorsque le résultat nous est donné entre le minimum et la maximum il y a un espace. Or cet espace compte bien comme étant un caractère.

Pour le maximum cela donne :

```
"${hygrometrie:3:2}"
```

Ensuite lorsque nous voulons comparer le débit d'eau, il a fallu penser que chaque zone a son propre débit d'eau, comme une serre est composée de 4 zones. Nous avons opté pour faire 4 évaluations de tests pour savoir si le débit de chaque zone était supérieur au maximum qui a été fixé.

Ainsi pour les diverses zones la concaténation se présente comme ce qui s'en suit :

- Pour la zone 1 : le point de départ est de 0 et la longueur est de 2
- Pour la zone 2 : le point de départ est de 3 et la longueur est de 2
- Pour la zone 3 : le point de départ est de 6 et la longueur est de 2
- Pour la zone 4 : le point de départ est de 9 et la longueur est de 2

Enfin pour finir sur la concaténation des chaînes, nous avons eu du mal à pouvoir comparer les températures, car déjà elles sont exprimées en tant que réel. Mais également, nous n'avions pas pensé au premier abord que les températures pouvaient avoir des longueurs différentes.

En effet si la température est normalement prévue comme étant un réel de 4 caractères (10.0 et plus), nous n'avions pas pensé au cas où elle peut être inférieure, par exemple 5.1, dans ce quel cas la chaîne de caractères est égale à 3. Lorsque nous faisions nos tests, nous n'avions pas remarqué de prime abord cette erreur qui empêchait donc de pouvoir comparer, car le Bash Linux ne compare que des entiers, des réels de même longueur.

Pour résoudre cette erreur, nous avons donc réagi de la manière suivante :

Au lieu de faire une requête SQL simple en ne demandant que la valeur brute ; telle que :

```
SELECT temperatureInterieur
```

```
FROM laSerre.MesuresSerreInterieur  
ORDER BY DateMesureInterieur  
DESC LIMIT 1 ;
```

Nous allons demander via la requête SQL de compléter la valeur souhaitée avec des « 0 » afin d'obtenir une chaîne de caractères de la longueur souhaitée, ainsi pour la température nous avons opté pour que toutes les chaînes de caractères aient la même longueur de 4.

Ceci se traduit comme ceci en langage SQL :

```
SELECT LPAD(temperatureInterieur, 4, '0')  
FROM laSerre.MesuresSerreInterieur  
ORDER BY DateMesureInterieur  
DESC LIMIT 1 ;
```

Nous avons opté pour cette même solution pour la luminosité, en effet son minimum étant de 10 et son maximum étant de 100 000 lux. Nous ne pouvions comparer des valeurs de longueur de chaînes différentes.

Pour la luminosité, nous avons opté pour une longueur de chaîne de 6 correspondants à la longueur du seuil maximal.

La requête SQL permettant cela est sensiblement la même, nous avons juste modifié la longueur de la chaîne.

```
SELECT LPAD(luminositeInterieur, 6, '0')  
FROM laSerre.MesuresSerreInterieur  
ORDER BY DateMesureInterieur  
DESC LIMIT 1 ;
```

1.5 Tests de comparaisons

1.5.1 Comparaison d'entiers

Afin de pouvoir prendre en charge la demande qui est de recevoir des alertes sous forme de SMS en cas de dépassement de seuil, il a fallu comparer les diverses valeurs des capteurs qui sont présentes dans la base de données dans les tables associées, et les seuils qui ont été fixés sur le site Internet dédié à la gestion de la serre, ces seuils se retrouvent également au sein de la base de données dans les tables affectées à cet effet.

Afin de rendre compte d'un test de comparaison, nous pouvons voir ci-dessous un exemple de test de comparaison codé en script Bash Linux

```
if [ "$hygrometrieInterieur" -ge "${hygrometrie:3:2}" ] && [ "$confirmationLecture" == "1" ]
then
    echo "hygro int au dessus du max"
    gammu sendsms TEXT $numTel -text "Alerte! L'hygrométrie intérieure ($hygrometrieInterieur%) est au dessus du seuil maximum (${hygrometrie:3:4}%) qui a
été défini dans la zone $idZoneMesuresInterieur de la serre $idSerreMesuresInterieur"
    mysql -h "$hostname" -u "$userName" -D"$dataBaseName" -p"$password" -N -B --execute="INSERT INTO laSerre.Incident(alerteSMS,messageSMS,natureIncident,
idZone,confirmationEnvoiMessage) VALUES('true','L hygrométrie intérieure ($hygrometrieInterieur%) est au dessus du seuil maximum (${hygrometrie:3:4}%) dans la
zone $idZoneMesuresInterieur de la Serre $idSerreMesuresInterieur','hygro int au dessus du max','1','envoyé');"
fi
```

Ce test de comparaison est celui qui permet de comparer l'hygrométrie intérieure au seuil maximum de l'hygrométrie fixée. Nous y ajoutons une condition supplémentaire, que la confirmation de lecture du dernier message soit égale à 1, afin de ne pas envoyer des messages toutes les 15 minutes si l'incident n'est pas pris en compte.

Une fois la confirmation de lecture étant égale à 1, nous pouvons effectuer ce test de comparaison. Si ce test est vrai, c'est-à-dire que l'hygrométrie intérieure est supérieure au seuil maximum, alors un SMS est envoyé via l'utilitaire Gammu installé.

En même temps que le SMS s'envoie vers le numéro du responsable à prévenir, en ce qui concerne cet envoi la ligne de commande sera détaillée plus bas dans ce dossier. Nous ajoutons une ligne dans la table Incident répertoriant l'ensemble des incidents qui se produisent au sein de la serre, ainsi dès qu'un incident se produit, une alerte est émise afin de prévenir le responsable, mais également les incidents sont répertoriés dans la base de données afin de pouvoir les prendre en compte et assurer une meilleure gestion de la serre.

1.5.2 Comparaison de réels

En dépit de la comparaison d'entiers tels que :

- L'hygrométrie
- La luminosité
- Le débit d'eau

Il nous reste à comparer la température, or celle-ci est exprimée en tant que réel, donc un nombre à virgule, cette comparaison est quelque peu différente des autres, en effet elle n'est pas possible de base avec le script Bash Linux.

L'étudiant n°4 a donc dû installer un utilitaire qui est Basic Calculator en ligne de commande depuis la Raspberry.

Afin de pouvoir l'installer, la ligne de commande permettant d'installer Basic Calculator est la suivante :

```
sudo apt-get install bc
```

Une fois que Basic Calculator fut installé, la syntaxe afin de comparer deux réels est différente de celle qui permet de comparer deux entiers.

```
if [ 1 -eq "${echo "${temperatureInterieur} < ${temperature:0:4}" | bc)" ] && [ "$confirmationLecture" == "1" ]
```

Nous pouvons également remarquer une différence entre la comparaison de réels et la comparaison d'entiers : l'opérateur mathématique.

En effet lorsque nous travaillons sur des réels nous utilisons les opérateurs dits classiques tels que :

- « > » : strictement supérieur
- « < » : strictement inférieur

Tandis que dans le cas d'une comparaison d'entiers, ce ne sont plus les opérateurs classiques.

- « -ge » : correspond à strictement supérieur
- « -lt » : correspond à strictement inférieur

1.6 Ajout de l'incident dans la base de données

Afin de rendre état des incidents qui se produisent dans la serre, nous avons opté pour créer une table nommée Incident qui répertorie l'ensemble des incidents qui se sont produits.

Chaque incident est caractérisé par :

- Un ID propre, chaque incident à son ID unique
- Une alerte SMS qui est un booléen qui correspond au choix du responsable, si la valeur du champ alerte SMS est true alors il a choisi une alerte par SMS
- Le message SMS qui est une copie du message envoyé au responsable
- La date de l'incident avec l'heure à laquelle l'incident s'est produit (avec une plage de plus ou moins 15 minutes, car nous venons interroger la base de données toutes les 15 minutes)
- La nature de l'incident qui est un résumé de l'incident
- L'ID de la Zone si l'alerte concerne l'humidité ou le débit d'eau
- La confirmation de l'envoi du message : si le message est envoyé, nous notons que la confirmation est égale à truc
- La confirmation de lecture du message, pour cette dernière il faut absolument passer par le site Internet qui aura la page alerte dédiée afin de pouvoir marquer les messages comme lus.

Ainsi chaque fois qu'un incident se produit, une alerte est envoyée au responsable, mais elle est aussi stockée dans la base de données pour éventuellement améliorer la surveillance de la Serre.

Ce stockage d'incident est alors réalisé par des requêtes SQL, telles que celle-ci :

```
minimum (${hygrometrie:0:2}) qui a été défini dans la Serre numéro $idSerre
mysql -h "$hostname" -u "$userName" -D"$dataBaseName" -p"$password" -N -B --execute="INSERT INTO laSerre.Incident(aler
teSMS,messageSMS,natureIncident,idZone,confirmationEnvoiMessage) VALUES('true','L hygrométrie intérieure ($hygrometrieInterieu
r%) est en dessous du seuil minimum ${hygrometrie:0:2}%) dans la Serre numéro $idSerre','hygro int en deça du min','1','envoy
é');"
fi
```

1.7 Envoi de SMS

Une fois toutes les informations prêtes, comparées, nous avons besoin d'envoyer un SMS afin de prévenir le responsable si un incident est en train de se passer.

Pour ce faire, nous avons stocké le numéro de téléphone dans une variable nommée **\$numTel** afin de pouvoir l'utiliser et ainsi ne pas taper le numéro de téléphone en tant que chaîne de caractères sur l'ensemble des commandes liées à Gammu.

```
numTel=`mysql -h "$hostname" -u "$userName" -D"$dataBaseName" -p"$password" -N -B --execute="connect laSerre; SELECT numTel FROM laSerre.Utilisateur order by idUtilisateur asc limit 1;"`
```

Une fois le numéro de téléphone récupéré et stocké dans la variable associée qui est **\$numTel** nous l'avons utilisé dans le script en ligne de commande permettant l'envoi d'un SMS.

```
gammu sendsms TEXT $numTel -text "Alerte! L'hygrométrie intérieure ($hygrometrieInterieur%) est en dessous du seuil minimum (${hygrometrie:0:2}%) qui a été défini dans la zone $idZoneMesuresInterieur de la Serre $idSerre MesuresInterieur"
```

Voici la requête afin de pouvoir prévenir le responsable via SMS, nous voyons donc que le SMS se décompose en plusieurs parties :

- La valeur de la grandeur physique mesurée
- Le seuil (minimum/maximum) associé
- L'ID de la Zone
- L'ID de la Serre

1.8 Test de connexion à la base de données

Au début du programme ./scriptEnvoiSMS.sh nous avons déclaré l'ensemble de nos variables nécessaires au bon fonctionnement du programme. Mais nous avons choisi de tester la bonne connexion à la base de données, si tel est le cas alors le programme s'effectue et se lance ; or si ce n'est pas le cas, le programme est quitté de manière directe.

Une sorte de gage de sécurité afin de s'assurer que la connexion est bien assurée.

Ainsi le test de connexion se présente de la manière suivante :

```
if [ "$hostname" = "172.18.58.213" ] && [ "$dataBaseName" = "laSerre" ] && [ "$userName" = "serreUtilisateur" ] && [ "$password" = "Touchard72" ]
then
    echo "Accès à la base de données OK"
else
    echo "Veuillez vérifier vos identifiants de connexion (adresseIP, nom de la base, nom d'utilisateur, mot de passe). La connexion a échoué"
```

Ainsi nous pouvons voir que si l'adresse IP, le nom de la base, le nom d'utilisateur, le mot de passe réuni ne sont pas exactement les mêmes que ceux qui sont précisés, le programme ne fonctionnera pas et se quittera de manière automatique. De la même manière, ceci est un incident et donc si une perte de connexion se produit, elle sera répertoriée dans la base de données.



1.9 Échec de la connexion à la base de données

Toujours concernant les numéros de téléphone et l'envoi de SMS, nous n'avons toujours pas évoqué le fait qu'il faille prévenir le responsable en cas d'échec de connexion à la base de données.

Ainsi pour se faire, nous avons créé une seconde variable liée au numéro de téléphone, une sorte de numéro de téléphone d'urgence. La particularité dans cette variable c'est qu'effectivement si la connexion échoue nous ne pouvons pas récupérer les informations de la base de données. Nous avons donc décidé de passer en paramètres, les paramètres directs de la base de données afin de récupérer le numéro de téléphone du responsable.

2. Complément d'alerte : l'alerte mail

En effet, le responsable de la serre choisit s'il décide de vouloir être alerté par SMS ou bien par mail.

Il a donc fallu prendre en compte cette demande, ce qui en soit, relève du même niveau que le SMS, car c'est par un utilitaire installé que nous allons envoyer un mail.

Une fois PostFix installé, configuré, il nous a fallu ajouter cette option au niveau du code. Nous l'avons incorporé dans le programme déjà existant afin de ne pas créer de doublons.

PostFix se manipule via ligne de commande, nous avons donc décidé de tester l'envoi de mail en cas de dépassement de seuils si tel est le cas.

Afin de pouvoir envoyer un mail, il nous a fallu stocker l'adresse mail du responsable présente dans la base de données, qu'il aura remplie directement depuis le site Internet dédié. Le numéro de téléphone que nous récupérons étant celui du premier responsable dans la liste.

```
mail=mysql -h "$hostname" -u "$userName" -D "$dataBaseName" -p "$password" -N -B --execute="connect laSerre; SELECT mail FROM laSerre.Utilisateur order by idUtilisateur asc limit 1;"`
```

Une fois que nous avons récupéré le mail, nous l'avons affecté dans une variable **\$mail**, afin de ne pas avoir à retaper le mail du responsable dans chaque commande. Mais également afin de prévoir s'il y a un changement de responsable, que ce changement soit pris en compte.

Ainsi dès qu'un incident se produit, en fonction du choix du responsable, une alerte est émise.

Afin de pouvoir envoyer un mail nous avons utilisé une commande que nous avons mise dans le test de comparaison, si celui est vrai alors un mail est envoyé.

La syntaxe afin de pouvoir envoyer un mail en cas d'alerte d'hygrométrie supérieure au seuil maximum se présente comme ce qui s'en suit :

Echo «L'hygrométrie est au-dessus du seuil autorisé ! Voici la valeur relevée
\$hygrometrieInterieur tandis que le seuil maximum est de

\${hygrometrie:3:2}» | mail -s « Hygrométrie intérieure au-dessus du seuil fixé » \$mail -a
« From : Raspberry PI »

3.CRONTAB

Une fois que l'ensemble des utilitaires fonctionne, une fonctionnalité qui fut attendue également fut celle de pouvoir brancher le Raspberry PI, lors de ce lancement le programme principal se lance de manière totalement autonome.

Pour se faire, nous avons décidé d'utiliser crontab qui est un utilitaire permettant de planifier des tâches sous Linux ainsi que sur Raspbian.

Il a fallu l'installer, en ligne de commande, et ensuite pouvoir éditer les actions préalablement définies.

Pour accéder aux actions de crontab :

```
crontab -e
```

Une fois arrivée dans l'éditeur des actions, il nous a juste fallu rajouter une ligne précisant que nous voulions que le programme ./scriptEnvoiSMS.sh se lance dès le démarrage du Raspberry PI. Mais également, nous devions indiquer quelques précisions, car crontab étant un planificateur de tâches :

- La fréquence de temps : toutes les x minutes, x heures
- Quel programme lancer

```
# m h dom mon dow   command
*/15 * * * * ./scriptEnvoiSMS.sh
```

Ainsi nous pouvons voir que nous avons choisi de lancer le programme dès le démarrage du Raspberry, mais également le lancer toutes les 15 minutes afin de garder un certain contrôle de la gestion des alertes, mais également de la base de données.

4. Conclusion

Pour ma part, en guise de conclusion je voudrais conclure sur divers points tels que :

Le projet m'a permis avant tout de travailler en équipe sur une longue période et cela m'a permis de voir comment s'organiser, comment travailler à plusieurs, j'en tire de très bonnes conclusions.

Concernant le programme, celui-ci marche sans difficulté, sans aucune intervention extérieure dès que les paramètres de la base de données (adresse IP, nom de la base, login/mot de passe) ont été définis.

Ce projet a pu me permettre de m'intéresser fortement au script Bash Linux, lequel je n'avais que peu de connaissances. Ce projet m'a également permis de revoir et de voir l'utilisation du langage SQL dans un projet.

Au moment de la rédaction de ce dossier, les points qui seraient à même d'être améliorés seraient peut-être les lignes de codes qui sont un peu répétitives.

F – Planning prévisionnel

1. Revue 1

Nom	Durée	Début	Fin		14 janv. 19	21 janv. 19	28 janv. 19
					S D L M M V	S D L M M V	S D L M M V
Début du projet	0 jours	15/01/19 08:00	15/01/19 08:00				
Etude du cahier des charges	5 jours	15/01/19 08:00	21/01/19 17:00				
Rédaction des parties communes	2 jours	22/01/19 08:00	23/01/19 17:00				
Rédaction des parties individuelles	4 jours	24/01/19 08:00	29/01/19 17:00				
Mise en commun du dossier	1 jour	30/01/19 08:00	30/01/19 17:00				
Réalisation diaporama revue n°1	4 jours	24/01/19 08:00	29/01/19 17:00				
Revue n°1	1 jour	31/01/19 08:00	31/01/19 17:00				

2. Étudiant n°1

Choix du capteur d'humidité de sol	1 jour	04/02/19 08:00	04/02/19 17:00
Documentation contrôleur de vannes	2 jours	05/02/19 08:00	06/02/19 17:00
Test du capteur d'humidité de sol	1 jour	07/02/19 08:00	07/02/19 17:00
Test de connexion sur le contrôleur de vannes	3 jours	08/02/19 08:00	26/02/19 17:00
Réalisation dossier revue n°2	3 jours	27/02/19 08:00	01/03/19 17:00
Réalisation diaporama revue n°2	1 jour	04/03/19 08:00	04/03/19 17:00
Revue n°2	1 jour	05/03/19 08:00	05/03/19 17:00
Contrôler les vannes (ouverture/fermeture)	5 jours	06/03/19 08:00	12/03/19 17:00
Test de mesure du débit d'eau	5 jours	13/03/19 08:00	19/03/19 17:00
Gérer l'arrosage localement	5 jours	20/03/19 08:00	26/03/19 17:00
Réalisation dossier revue n°3	6 jours	27/03/19 08:00	03/04/19 17:00
Réalisation diaporama revue n°3	1 jour	20/03/19 08:00	20/03/19 17:00
Revue n°3	1 jour	04/04/19 08:00	04/04/19 17:00
Création d'un mode manuel	5 jours	05/04/19 08:00	26/04/19 17:00
Préparation Dossier Final	9 jours	13/05/19 08:00	23/05/19 17:00
Création d'un mode programmé	5 jours	24/05/19 08:00	30/05/19 17:00
Création d'un mode automatique	5 jours	31/05/19 08:00	06/06/19 17:00
Création de la fonction MAJ des cycles d'arrosage	5 jours	29/04/19 08:00	03/05/19 17:00
MAJ des seuils	5 jours	06/05/19 08:00	10/05/19 17:00
Réalisation du guide-utilisateur	10 jours	13/05/19 08:00	24/05/19 17:00
Préparation revue finale	1 jour	27/05/19 08:00	27/05/19 17:00
Revue finale	1 jour	28/05/19 08:00	28/05/19 17:00

3. Étudiant n°2

Choix des capteurs	1 jour	04/02/19 08:00	04/02/19 17:00
Programmation des relevés des données météorologiques	3 jours	05/02/19 08:00	07/02/19 17:00
Test	1 jour	08/02/19 08:00	08/02/19 17:00
Intégration dans le système	2 jours	25/02/19 08:00	26/02/19 17:00
Réalisation dossier revue n°2	4 jours	27/02/19 08:00	04/03/19 17:00
Réalisation diaporama revue n°2	1 jour	05/03/19 08:00	05/03/19 17:00
Revue n°2	1 jour	06/03/19 08:00	06/03/19 17:00
Création de la BDD	6 jours	07/03/19 08:00	14/03/19 17:00
Implémentation de la BDD	3 jours	15/03/19 08:00	19/03/19 17:00
Installation du serveur	5 jours	20/03/19 08:00	26/03/19 17:00
Réalisation dossier revue n°3	6 jours	28/03/19 08:00	04/04/19 17:00
Réalisation diaporama revue n°3	1 jour	05/04/19 08:00	05/04/19 17:00
Revue n°3	1 jour	23/04/19 08:00	23/04/19 17:00
Enregistrer localement les données météorologiques	6 jours	24/04/19 08:00	01/05/19 17:00
Mémoriser incident/perte de connexion	6 jours	02/05/19 08:00	09/05/19 17:00
Envoyer les données de la serre	5 jours	10/05/19 08:00	16/05/19 17:00
MAJ BDD	5 jours	17/05/19 08:00	23/05/19 17:00
Réalisation du dossier d'analyse	6 jours	24/05/19 08:00	31/05/19 17:00
Préparation de la revue finale	1 jour	03/06/19 08:00	03/06/19 17:00
Revue finale	1 jour	12/06/19 07:00	12/06/19 17:00

4. Étudiant n°3

Conception détaillée	73 jours	01/02/19 08:00	12/06/19 17:00
Installation serveur web raspberry	3 jours	04/02/19 08:00	06/02/19 17:00
Configuration serveur web	5 jours	07/02/19 08:00	27/02/19 17:00
Configuration serveur apache	1 jour	28/02/19 08:00	28/02/19 17:00
Réalisation dossier revue n°2	4 jours	27/02/19 08:00	04/03/19 17:00
Réalisation diaporama revue n°2	1 jour	05/03/19 08:00	05/03/19 17:00
Revue n°2	1 jour	06/03/19 08:00	06/03/19 17:00
Création du site web	10 jours	07/03/19 08:00	20/03/19 17:00
Réalisation dossier revue n°3	6 jours	21/03/19 08:00	28/03/19 17:00
Réalisation diaporama revue n°3	2 jours	29/03/19 08:00	01/04/19 17:00
Revue n°3	1 jour	02/04/19 08:00	02/04/19 17:00
Création page web d'authentification	10 jours	03/04/19 08:00	01/05/19 17:00
Récupération des données de la BDD	3 jours	02/05/19 08:00	06/05/19 17:00
Visualisation des données sur le site	10 jours	07/05/19 08:00	20/05/19 17:00
Programmation des paramètres d'une serre	5 jours	21/05/19 08:00	27/05/19 17:00
Test d'intégration	2 jours	28/05/19 08:00	29/05/19 17:00
Réalisation du dossier de conception	8 jours	30/05/19 08:00	10/06/19 17:00
Préparation revue finale	1 jour	11/06/19 08:00	11/06/19 17:00
Revue finale	1 jour	12/06/19 08:00	12/06/19 17:00

5. Étudiant n°4

Réalisation du diagramme de cas d'utilisation "	1 jour	01/02/19 08:00	01/02/19 17:00
Réalisation du diagramme de séquence "Contrôler l'état d'une serre"	1 jour	04/02/19 08:00	04/02/19 17:00
Réalisation du diagramme de classe "Contrôler l'état d'une serre"	1 jour	05/02/19 08:00	05/02/19 17:00
Réalisation du diagramme de cas d'utilisation "Envoyer une alerte"	1 jour	06/02/19 08:00	06/02/19 17:00
Réalisation du diagramme séquence "Envoyer une alerte"	1 jour	07/02/19 08:00	07/02/19 17:00
Réalisation du diagramme de classe "Envoyer une alerte"	1 jour	08/02/19 08:00	08/02/19 17:00
Documentation sur les outils	2 jours	25/02/19 08:00	26/02/19 17:00
Élaboration dossier revue n°2	3 jours	27/02/19 08:00	01/03/19 17:00
Élaboration diaporama revue n°2	1 jour	04/03/19 08:00	04/03/19 17:00
Revue 2	1 jour	05/03/19 08:00	05/03/19 17:00
☒ Contrôler l'état d'une serre	9 jours	06/03/19 08:00	18/03/19 17:00
Installation d'un serveur de BDD	1 jour	06/03/19 08:00	06/03/19 17:00
Documentation script Batch Linux	3 jours	07/03/19 08:00	11/03/19 17:00
Installation de Raspbian	2 jours	12/03/19 08:00	13/03/19 17:00
Configuration de la BDD mariaDB	3 jours	14/03/19 08:00	18/03/19 17:00
☒ Configuration des seuils	50,25 jo...	19/03/19 08:00	12/06/19 10:00
Programmation des intervalles de temps	8 jours	19/03/19 08:00	28/03/19 17:00
Réalisation dossier revue n°3	6 jours	21/03/19 08:00	28/03/19 17:00
Réalisation diaporama revue n°3	1 jour	29/03/19 08:00	29/03/19 17:00
Revue n°3	1 jour	02/04/19 07:00	02/04/19 17:00
Programmation de l'alerte SMS	6 jours	29/03/19 08:00	05/04/19 17:00
Programmation de l'alerte courriel	6 jours	23/04/19 08:00	30/04/19 17:00
Réalisation fiche de test "Contrôler l'état d'une serre"	2 jours	01/05/19 08:00	02/05/19 17:00
Réalisation fiche de test "Envoyer une alerte"	2 jours	03/05/19 08:00	06/05/19 17:00

G– Les tests unitaires

I/ Test unitaire : étudiant n°1, MICHAUD Théo

Fiche de test					
Nature :	Fonctionnel	Référence :	F1.1		
Module :	Classe Zone				
Objectif :	Gérer l'ouverture et la fermeture des vannes				
Condition du test					
État initial du module		Environnement du test			
Ordinateur	Sous Linux	Ordinateur avec Arduino L298N présent Antelco ezyvalve 4 ESP32 Smartphone avec terminal Bluetooth			
Programme	/GestionSerre				
Conditions initiales					
L'ESP32 est branché Le système des vannes est bien câblé Le programme est téléchargé sur la carte Lancer le terminal Arduino Le terminal Bluetooth est installé Smartphone connecté en Bluetooth sur l'ESP32					
Procédure de test					
<i>Il est possible d'ouvrir ou de fermer, en mode manuel, une vanne pour permettre le passage ou non de l'eau dans le système.</i>					
Repère	Opérations	Résultats attendus			
1	Lancer le terminal Bluetooth et se connecter sur l'ESP32	Le terminal Bluetooth est bien connecté à l'ESP32 Un message affiche «Vous êtes connectés à la Serre»			

2	Envoyer 'z1mo' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z1mo'</p> <p>Le terminal Arduino nous indique que l'on passe en mode manuel pour la zone 1 et que la vanne 1 s'ouvre</p> <p>Une impulsion de 250ms est envoyée sur la vanne 1 , Un clac nous confirme qu'elle est ouverte, Sur l'oscilloscope, on voit notre impulsion de 250ms</p>
3	Envoyer 'z1mf' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z1mf'</p> <p>Le terminal Arduino nous indique que l'on passe en mode manuel pour la zone 1 et que la vanne 1 se ferme</p> <p>Une impulsion de 250ms est envoyée sur la vanne 1 , Un clac nous confirme qu'elle est fermée, Sur l'oscilloscope, on voit notre impulsion de 250ms</p>
4	Envoyer 'z1p' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z1p'</p> <p>Le terminal Arduino nous indique que l'on passe en mode programmé pour la zone 1</p>
5	Envoyer 'z1a' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z1a'</p> <p>Le terminal Arduino nous indique que l'on passe en mode automatique pour la zone 1</p>
6	Envoyer 'z2mo' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z2mo'</p> <p>Le terminal Arduino nous indique que l'on passe en mode manuel pour la zone 2</p> <p>La vanne 2 s'ouvre</p> <p>Une impulsion de 250ms est envoyée sur la vanne 2 , Un clac nous confirme qu'elle est ouverte, Sur l'oscilloscope, on voit notre impulsion de 250ms</p>
7	Envoyer 'z2mf' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z2mf'</p> <p>Le terminal Arduino nous indique que l'on passe en mode manuel pour la zone 2</p> <p>La vanne 2 se ferme</p> <p>Une impulsion de 250ms est envoyée sur la vanne 2 , Un clac nous confirme qu'elle est fermée, Sur l'oscillo, on voit notre impulsion de 250ms</p>
8	Envoyer 'z3xr' dans le terminal Bluetooth	<p>l'ESP32 reçoit la chaîne 'z3xr'</p> <p>Le terminal Arduino nous indique que l'on agit sur la zone 3, mais rien ne se passe, car 'xr' n'est pas connu.</p>

II/ Test unitaire : étudiant n°2, AUVÉ Killian

Fiche de tests					
Nature :	Fonctionnel	Référence :	F2.1		
Module :	Classe Capteurs, StockageDonnees, ControleurDeSerre				
Objectif :	Mesure de la température, humidité de l'air, luminosité et envoi des données sur le serveur				
Condition du test					
État initial du module		Environnement du test			
Ordinateur Programme	Linux ESP32.ino	Arduino IDE BME280 TSL2591, ESP32 Module RTC Serveur Raspberry Pi PhpMyAdmin			
Conditions initiales					
L'objet connecté ESP32 est sous tension Les capteurs sont branchés, le module RTC est branché Lancement du programme sur l'ESP32, le serveur est lancé (@IP:172.18.58.213)					
Repère	Opérations	Résultats attendus			
1	Se connecter au serveur à l'adresse suivante 172.18.58.213	La page de connexion PhpMyAdmin s'affiche			
2	Rentrer : <ul style="list-style-type: none">Identifiant : serreUtilisateurMot de passe : Touchard72 Appuyer sur le bouton exécuter	La page d'accueil PhpMyAdmin s'affiche			
3	Sélectionner la base de données laSerre	La liste des tables s'affiche			
4	Cliquer sur la table MesuresSerreInterieur	Les données de température, d'hygrométrie et de luminosité s'affichent			
5	Au bout de 30 secondes (pour le test) Rafraîchir la page	Les données ont bien été insérées			
6	Poser les doigts sur les capteurs se température et de luminosité	Les données insérées varient bien			

III/ Test unitaire : étudiant n°3, VILLERMIN Maxime

<i>Fiche de tests</i>			
Nature :	Fonctionnel	Référence : F3.1	
Module :	Visualisation des données de la serre		
Objectif :	Visualiser des données dans un graphique		
<i>Condition du test</i>			
État initial du module		Environnement du test	
Ordinateur Programme	Linux Supervision de serre	Ordinateur avec connexion internet et un moteur de recherche Raspberry Pi 3 (serveur web) - PhpMyAdmin : 172.18.58.213/phpmyadmin Dans la BDD : laSerre Dans la table : MesureSerreInterieur Login : serre Mdp : Touchard72	
<i>Conditions initiales</i>			
<ol style="list-style-type: none"> 1. Le serveur web sur raspberry est branché 2. Aller sur: http://172.18.58.213/phpmyadmin et vérifier que des données de mesure de la serre sont enregistrés dans la table Serre 3. Aller sur la page: http://172.18.58.213/ProjetSerre/php/visualiser.php 			
<i>Procédure de test</i>			
<i>Lancement de l'application en mode exécution normale</i>			
Repère	Opérations	Résultats attendus	Résultats obtenus
1	Cliquer sur la 1ère liste déroulante de choix d'une serre	La liste des serres s'affiche	Résultat attendu
2	Sélectionner depuis la 1ère liste déroulante « Veuillez sélectionner votre serre »	Dans la liste déroulante des choix de zone aucune zone n'est disponible	Résultat attendu
3	Sélectionner depuis la 1ère liste déroulante « Veuillez sélectionner votre serre »	Si vous cliquez sur un checkbox de choix de grandeur, aucun graphique n'est disponible car pas de serre sélectionnée	L'association des grandeurs par rapport à l'Id serre n'est pas encore codée
4	Sélectionner une serre	La seconde liste déroulante est mise à jour en fonction de la	Résultat attendu

		serre sélectionnée	
5	Sélectionner depuis la 2ème liste déroulante une zone	Une zone est sélectionnée et selon la zone sélectionnée les graphiques seront mis à jour	Comme pour la liste déroulante de la serre, les grandeurs ne sont pas encore associés à la zone de serre
6	Cliquer sur le checkbox « Luminosité »	Un graphique contenant les données de luminosité s'affichera	Résultat attendu
7	Cliquer sur le checkbox « Luminosité »	Le graphique contenant les données de luminosité disparaîtra	Résultat attendu
8	Cliquer sur le checkbox « Température »	Un graphique contenant les données de température s'affichera	Résultat attendu
9	Cliquer sur le checkbox « Hygrométrie »	Un graphique contenant les données d'hygrométrie s'affichera	Résultat attendu
10	Cliquer sur le checkbox « Hygrométrie »	Le graphique contenant les données d'hygrométrie disparaîtra	Résultat attendu
11	Cliquer sur plusieurs checkbox	Les graphiques apparaissent les uns en dessous des autres	Résultat attendu
11	Actualiser la page	Les checkbox sont désactivés et les graphiques ont disparu	Résultat attendu
12	Sur un des graphiques, insérer une échelle de date	Le graphique se mettra à jour par rapport à l'échelle souhaitée si des données sont disponibles à cette date	Résultat attendu
13	Sur un des graphiques, déplacer le curseur en bas de celui-ci pour changer l'échelle de date	La courbe interagie avec le curseur	Résultat attendu
14	Déplacer le curseur de la souris sur la courbe d'un des graphiques	Vous visualisez la donnée correspondante au point du curseur, ainsi que sa date et sa grandeur	Résultat attendu

IV/ Test unitaire : étudiant n°4, ROCHER Marvyn

<i>Fiche de tests</i>					
Nature :	Fonctionnel	Référence :	F4.1		
Module :	<i>Module d'envoi de SMS</i>				
Objectif :	Envoyer une alerte SMS en cas de dépassement de seuil				
<i>Condition du test</i>					
État initial du module		Environnement du test			
Raspberry	Sous Raspbian	PhpMyAdmin 172.18.58.213/phpmyadmin Id: serreUtilisateur Mot de passe: Touchard72 Base: laSerre			
Programme	scriptEnvoiSMS.sh				
<i>Conditions initiales</i>					
<ul style="list-style-type: none"> - le Raspberry est branché et connecté au réseau - Les seuils ont été définis dans la table : laSerre. Serre - Quelques relevés des capteurs (hygrométrie, température) sont présents dans les tables : laSerre.MesuresSerreExterieur et laSerre.MesuresSerreInterieur -Les coordonnées (numéro de téléphone) du responsable à prévenir sont présentes dans la table : laSerre.Utilisateur 					
<i>Procédure de test</i>					
<i>Lancement de l'application en mode exécution normale</i>					
Repère	Opérations	Résultats attendus			
0	Insérer un numéro de téléphone valide dans le champ « numTel » de la table Utilisateur	Le numéro de téléphone du responsable a été défini			
1	Insérer 20 dans le champ « minHygrometrie » de la table Serre	Le seuil minimum de l'hygrométrie a été fixé			

2	Insérer 80 dans le champ « minHygrometrie » de la table Serre	Le seuil maximum de l'hygrométrie a été fixé
3	Insérer 5 dans le champ « minTemperature » de la table Serre	Le seuil minimum de la température a été fixé
4	Insérer 35 dans le champ « minTemperature » de la table Serre	Le seuil maximum de la température a été fixé
5	Insérer 2 dans le champ «temperatureInterieur » de la table MesuresSerreInterieur	La température intérieure a été fixée à 2 Un SMS est reçu pour signaler que la température est en dessous du minimum
6	Insérer 80 dans le champ «hygrometrieInterieur» de la table MesuresSerreInterieur	L'hygrométrie intérieure a été fixée à 80
7	Insérer 20 dans le champ «temperatureInterieur » de la table MesuresSerreInterieur	La température intérieure a été fixée à 20
8	Insérer 85 dans le champ «hygrometrieInterieur» de la table MesuresSerreInterieur	L'hygrométrie intérieure a été fixée à 85 Un SMS est reçu pour signaler que l'hygrométrie intérieure est au-dessus du maximum