Université du Québec en Outaouais

DÉPARTEMENT
D'INFORMATIQUE ET
D'INGÉNIERIE DE L'UQO

GEN 1273 - Modélisation et simulation

Automne 2016

Modélisation/simulation et réalisation d'une serre complètement automatisée pour une horticulture optimale.

Travail réalisé par :

Arielle Fanbeu Sipeyou

Cheikh Charles Diaw

Francis Normand

Julien Combattelli

Kadiatou Diallo

Zineb El Moutia

Présenté à

M. Kamel Adi

Vendredi 6 janvier 2017

Table de matières

Introduction

1. But

Le but de ce document est de spécifier les exigences d'un système de serre complètement automatisé pour une horticulture optimale.
Des exigences jusqu'à la réalisation tout en passant par la modélisation, ce document présente tous les éléments importants à la compréhension de la solution livrée.

2. Portée du système

La serre automatisée sera utilisée par Monsieur Sylvain Delagrange, professeur au département de sciences naturelles à l'UQO, dans le cadre de ses recherches en plantations à croissance rapide et en régénération forestière.

Le système de serre automatisée devra offrir un cadre de mesure et de contrôle des paramètres principaux dans une serre. Ces paramètres sont les principaux acteurs du processus bioénergétique de la photosynthèse.

3. Définitions

1) Contrainte : Sauf mention contraire, représente dans ce document les contraintes fournies par le client.

2) Photosynthèse[1] : processus de synthèse de la matière organique par les plantes, algues et certaines bactéries en utilisant l'énergie lumineuse.

3) Paramètres principaux : ces paramètres ont été choisis par le client comme étant les acteurs principaux pour l'automatisation de la serre.

   - Dioxyde de Carbone : agit sur le développement de la plante en tant que composant essentiel de la photosynthèse.
   - Eau : est un oxydant de la réaction chimique lors de la photosynthèse.

[1] http://www.futura-sciences.com/planete/definitions/botanique-photosynthese-227/

- Horticulture : art de cultiver les plantes.
- Humidité relative : facteur dépendant de la température qui influence la croissance de la plante.
- Lumière : provenant du soleil ou d'une source artificielle, est la source d'énergie pour le processus de la photosynthèse.
- Nutriments : les principaux étant l'azote (N), le potassium(K), le calcium(Ca) et le phosphore(P).
- Oxygène : est un des composants produits lors de la photosynthèse.
- Serre[2] : structure qui peut être parfaitement close destinée en général à la production agricole.
- Température : comme facteur d'influence de la consommation d'eau par la plante.

Description générale

1. Fonctionnalités

La serre automatisée devra offrir un certain nombre de fonctionnalités. Elle devra entre autres mesurer et contrôler les paramètres principaux.

Des mesures sont effectuées sur les paramètres suivants :

1. La température ambiante et l'humidité relative dans la serre.
2. Le niveau d'eau dans chacun des pots introduits dans la serre.
3. Le taux de CO_2 dans l'air.
4. Le taux de dioxygène (O_2) dans l'air qui peut être polluant pour les plantes.

Les paramètres suivants seront activement contrôlés :

1. La lumière interne de la serre (Couleur et intensité de la lumière).
2. La température ambiante et l'humidité relative de l'air.
3. Le niveau d'eau dans chacun des pots introduits dans la serre.

Une fonctionnalité souhaitée, mais dernière en priorité est l'ajout d'un module de prise

[2] https://fr.wikipedia.org/wiki/Serre

de photos des plantes de manière planifiée. Ceci dans le but d'aider le professeur Sylvain à développer des applications en phénologie[3].

2. Contraintes générales

- Durant le développement du projet, la serre doit être placée à l'abri des intempéries pour protéger les systèmes électroniques. Cependant, elle devra rester facilement accessible pour les étudiants amenés à travailler sur la serre.
- Toute l'installation électrique haute tension (110 volts et plus) sera faite par un électricien. Pour des raisons de sécurité, l'étudiant ne devra pas intervenir sur l'installation électrique haute tension.
- Le système doit permettre un accès facile et sécuritaire à l'ensemble des plantes : on parle principalement ici de la disposition des différents composants et de l'isolation électrique.
- Le système doit être réalisé avec les composants imposés.

3. Caractéristiques utilisateur

L'utilisateur doit avoir des connaissances en sciences naturelles afin comprendre paramètres mesurés et contrôlés par le système.

Description détaillée

Contraintes techniques

Le choix de plusieurs composants et dispositifs a été fait par le professeur. Ils seront par conséquent considérés comme des contraintes techniques du projet. Ce sont ces composants qui serviront à implémenter la solution.

Capteurs et autres dispositifs

[3] Le professeur Sylvain cherche à observer la croissance des plantes au moyen des photos et ainsi calculer des taux de croissance.

a. Acquisition de la concentration de CO_2 dans la serre

Le capteur de CO_2 SEN0159 a été choisi afin de mesurer la concentration de CO_2 dans la serre, considéré un composant essentiel de la photosynthèse. Le professeur Sylvain souhaite prendre connaissance de ce paramètre en tout moment, car il agit directement sur le développement de la plante.

Le principe de fonctionnement de ce capteur est simple ; l'augmentation de la concentration de CO_2 est signalée par une diminution de la tension de sortie du capteur. Il incorpore un potentiomètre conçu pour régler la tension de seuil.

Puissance d'entrée	3.3-6 v (5V de préférence)
Voltage d'opération	0-5V
Interface	Analogique
Sortie digitale unique4	5V
Taille	32x42mm

Tableau 1. **Spécifications techniques du capteur de CO_2**

b. Acquisition de la concentration de O_2 dans la serre

Le capteur seedstudio 160328001 servira à tester la concentration de dioxygène dans l'air. Il est important de mesurer ce paramètre, car il constitue la sortie souhaitée du processus de photosynthèse. C'est la sortie mesurable et dépendent de tous les paramètres contrôlables.

Le principe de fonctionnement réside dans la proportionnalité existante entre la tension de sortie et la concentration de O_2. En tout moment, cette concentration est connue en se référant à un graphe caractéristique qui exprime cette relation de proportionnalité.

c. Contrôle du niveau de l'eau dans les pots en culture

[4] Une fois la concentration de CO_2 dépasse la valeur de seuil, la sortie s'active.

L'une des fonctionnalités de la serre automatisée pour une horticulture optimale est de pouvoir contrôler le niveau d'eau dans les pots en culture dans la serre. Ce contrôle se fait par ajout d'une certaine quantité d'eau selon le niveau d'eau (qui dépendra de ce que le capteur censé recueillir le niveau d'eau va afficher). Pour assurer cette fonctionnalité, nous utilisons une électrovanne qui est une vanne commandée de façon électrique. Le modèle d'électrovanne imposé pour le projet est **l'ORBIT 57253.**

Ce produit se présente sous la forme de 3 soupapes (vannes) préassemblées qui peuvent facilement être installées avec un système de canalisation d'eau. Il permet de contrôler le débit d'eau qui s'écoule à travers les 3 vannes selon un choix mesuré (qui peut être à partir d'un programmateur). Les vannes sont donc contrôlées par un courant électrique à travers un électro-aimant.

Figure 1. **Orbit 57253**

d. Activation d'une partie du circuit grâce au Relais FOTEK SSR-40 DA (Solide state module) :

Qu'il s'agisse de contrôler l'allumage de la plaque chauffante, de l'humidificateur ou encore du débit de l'eau à travers les électrovannes, nous allons utiliser un relais électronique et le modèle imposé est le FOTEK SSR-40 DA.

Ce composant est un interrupteur qui se commande avec une tension continue de faible puissance (comme la tension que peut livrer une carte d'un microcontrôleur, en l'occurrence ici il s'agit de la carte Freescale FRDM-KL26). La partie interrupteur sert à piloter des charges provenant du secteur jusqu'à 40 A).

Figure 2. **Relai SSR 40 DA**

e. Acquisition du niveau d'eau dans chacun des pots

Les besoins en eau d'une plante varient en fonction de son genre, de son espèce, de sa taille, de son environnement et de la saison en cours. Il s'avère donc d'importer et de mesurer le niveau d'eau dans chaque pot afin de s'assurer que l'humidité de la terre reste convenable pour chacune des plantes. Ceci permettra de savoir s'il doit y avoir arrosage ou pas.

Ainsi, la mesure de ce paramètre sera faite grâce au capteur Soil Moisture Sensor SEN0114 compatible avec Arduino. Il est doté de senseurs à humidité qui permettent la lecture précise de l'humidité du sol. Aussi, il est couvert d'une membrane appelée immersion gold pour permettre une fiabilité de longue durée.

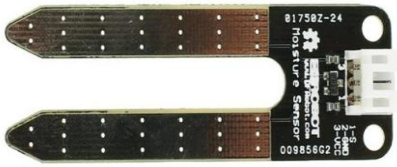

Figure 3. **Soil Moisture Sensor SEN0114**

f. Acquisition de la température de l'air

Le Capteur de Température Waterproof DS18B20 avec puce est adapté pour la mesure précise de la température. Ce Capteur avec câble étanche de 1 mètre doit être soudé et est adapté pour une utilisation en extérieur ou sous l'eau. Il servira à mesure la température ambiante de la serre qui est un paramètre non négligeable qu'il est important de connaitre pour mieux le contrôler, car, des températures élevées ou trop

basses ont des effets négatifs sur la croissance et le développement des plantes. Le DS18B20 est composé d'un convertisseur analogique - numérique, d'une zone mémoire de 8 octets et d'une EEPROM de 3 octets.

Ces zones de mémoire servent à communiquer avec le DS18B20 afin de récupérer les températures converties, configurer le convertisseur et configurer les valeurs de températures min et max pour la fonction "thermostat".

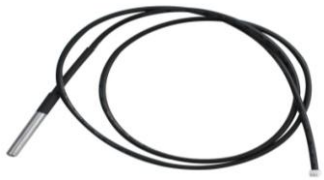

Figure 4. Température Waterproof DS18B20

g. Acquisition de l'humidité relative de l'air

Humidité relative de l'air est la quantité d'eau présente dans l'air par rapport à la quantité maximale d'eau que l'air peut contenir. Ce paramètre environnemental a des effets importants sur les maladies des plantes. Pendant le jour, il est souhaitable d'avoir une humidité relative se situant entre 60-80 %. La nuit, l'humidité relative dans la serre peut dépasser 95 % en raison de la respiration des plantes et des températures de l'air plus basses. Il s'avère donc important de bien contrôler ce paramètre pour la bonne croissance des plantes.

Ainsi, le capteur d'humidité et de température SEN0137 de DFROBOT permettra de mesurer, à la fois, la température et l'humidité sur un seul et même port, ce qui permet de doubler la capacité de mesure de chaque port d'un boîtier SENSOR. Le capteur SEN0137 2 en 1 permet une mesure de température comprise entre -40°C et +75°C et de 0 à 100% d'humidité relative. Chacune des 2 fonctions, température et humidité, possède son propre identifiant permettant ainsi de collecter, via le réseau, les données de l'une et/ou l'autre des mesures.

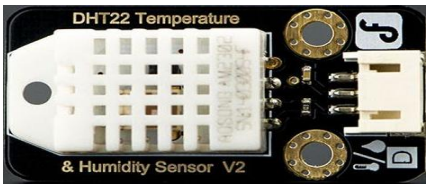

Figure 5. **Capteur de température et d'humidité SEN0137**

h. Microcontrôleurs utilisés

Freescale FRDM KL26Z :

Le microcontrôleur qui sera utilisé dans le cadre de ce projet est le **FRDM KL26Z** conçu par la compagnie **NXP semiconductors**. Il est fait avec le processeur ARM Cortex-M0+ qui est une architecture RISC 32 bits. Par ailleurs, il intègre une interface de débogage et de contrôle de l'exécution pour la programmation. Aussi l'environnement de développement Kinetis **Design Studio** va nous permettre d'éditer, de compiler et de déboguer nos différentes conceptions. En outre cet IDE contient le logiciel **Processor expert** permettant d'ajouter, de configurer et d'optimiser des composants ainsi que leurs codes sources. Processor Expert nous offre donc une abstraction permettant de créer des applications puissantes avec moins de code.

Figure 6. **Freescale FRDM KL26Z**

Arduino Mega 2560:

Cette carte sera utilisée pour contrôler certains de nos capteurs, dont le panneau de lumière. C'est une carte libre sur laquelle se trouve un microcontrôleur avec une architecture Atmel AVR. L'avantage de l'utilisation de celle-ci est que nous la maitrisons déjà et il y'a beaucoup de ressources en ligne qui seront utiles dans notre

implémentation.

Figure 7 **Arduino Mega**

Architecture globale de la solution

L'architecture proposée délimite la solution en trois grands modules.

Capteurs

Ce module sera chargé de faire l'étude et les expérimentations des capteurs.
Notamment les connexions avec la carte Freescale FRDM-KL26 ainsi que les tests de fonctionnement avant et après déploiement dans la serre.

Communication

Le module de communication est chargé de faire la liaison entre tous les composants de la serre. Il contrôle la manière dont l'information est diffusée à travers la serre, et s'assure que les données provenant des capteurs parviennent à l'interface utilisateur, et vice versa. Ce module est donc essentiel au fonctionnement général de la serre dans la mesure où sa défaillance bloque toute autre activité.
Les contraintes importantes dans la réalisation de ce module sont :

- Les différents modules devront communiquer sans fil.
- L'ajout de nouveaux composants doit être facile.

Interface Utilisateur

Ce module sera conçu afin d'accompagner le projet d'une interface utilisateur de mesure et contrôle des paramètres principaux à l'intérieur de la serre. Cet environnement virtuel devra faire abstraction des complexités techniques reliées à la mesure et contrôle de ces paramètres.

La seule contrainte spécifique à l'interface est l'utilisation d'openHAB pour son implémentation. Ce logiciel open source basé sur java offre un haut degré d'abstraction pour la conception d'applications domotiques. Il a été conçu pour fonctionner sur une variété de systèmes d'exploitations, ce qui explique sa flexibilité.

À l'aide d'openHAB nous serons en mesure d'offrir une interface utilisateur accessible à partir d'un navigateur standard à partir de laquelle l'utilisateur final sera en mesure de gérer tous les aspects de la serre automatique jugés pertinents.

Modélisation

Placement des capteurs

Pour des questions de sécurité et de praticité, nous allons utiliser une carte Freescale par capteur. Ainsi, le placement de chaque capteur dépend de son rayon d'émission et de la dimension de la serre.

La **plaque chauffante,** le **ventilateur et** l'**humidificateur** seront placés au fond de la serre sur la prise comportant un relai de 5v. Nous choisissons d'isoler ces équipements tout au fond de la serre question de faciliter l'accès et pour plus de sécurité.

Capteur de température de l'air : vu le placement de la plaque chauffante et de la disposition de la serre, nous jugeons plus réaliste de placer un capteur dans l'angle droit de la porte de la serre. Deux autres capteurs seront placés de part et d'autre sur les longueurs de la serre. La disposition de ces trois capteurs va nous permettre d'avoir la température moyenne dans la serre.

Le **capteur de température et d'humidité relative de l'air** sera placé près des capteurs de température afin de mesurer l'humidité relative moyenne dans la serre. Ainsi, nous aurons trois capteurs de température et d'humidité relative dans la serre.

Le nombre de c**apteurs de niveau d'eau** dépendra du nombre de pots introduits dans la serre. La position de ces capteurs est très importante, car ils doivent être placés à l'endroit où la plante tire la plupart de son eau, c'est-à-dire au niveau où la racine est la plus active.

Nous utiliserons deux **capteurs d'O_2** et deux **capteurs CO_2** qui seront placés de part et d'autre sur les longueurs de la serre. Chacun des couples de capteurs fournira une moyenne sur le taux d'O2 et de CO2 dans la serre.

Le **panneau de lumière** sera placé aux dessus des plantes à hauteur raisonnable de telle sorte que la lumière puisse être répartie de manière uniforme dans la serre.

Vue du dessus

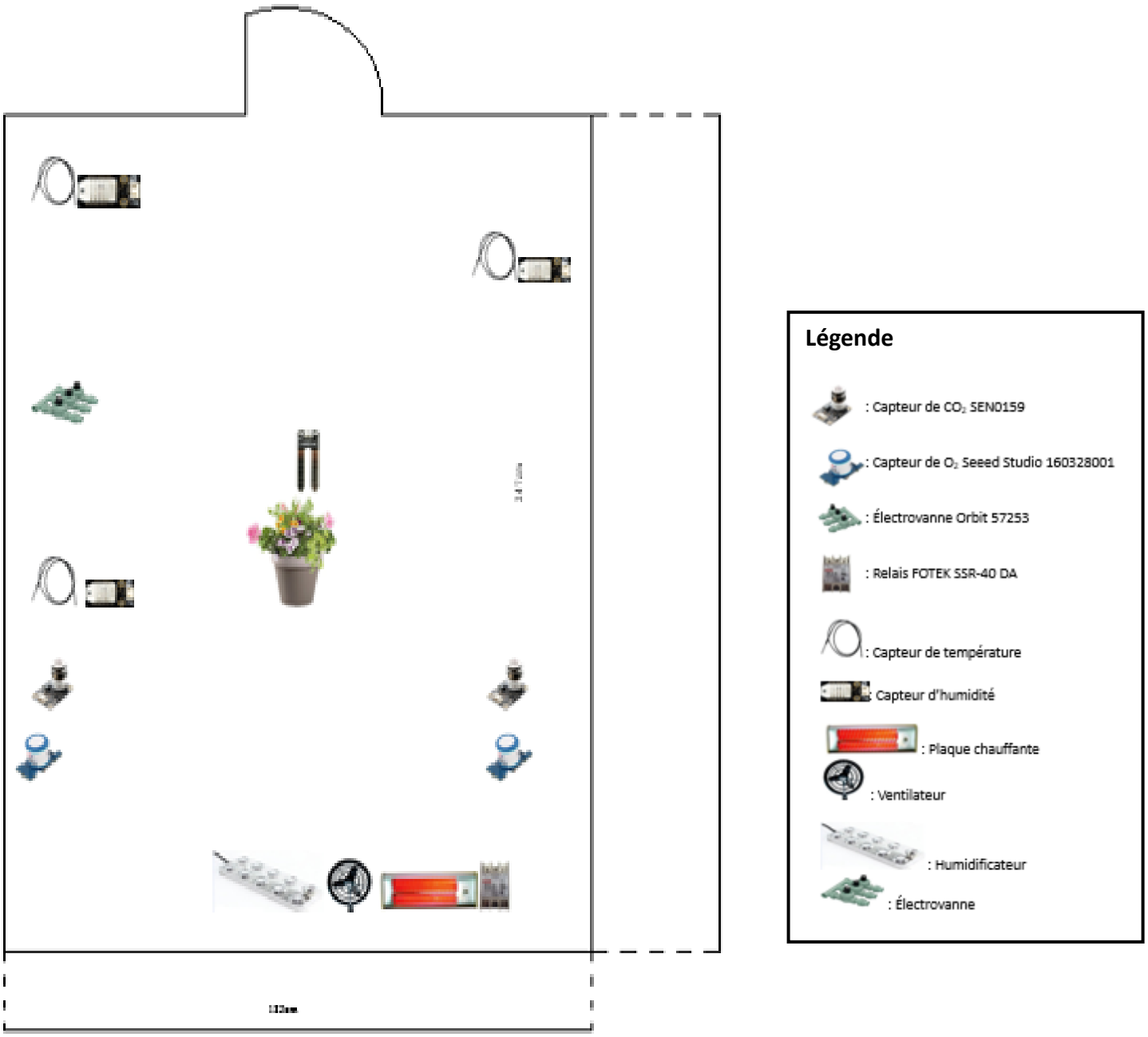

Figure 8. Modélisation placement capteurs

Architecture de communication

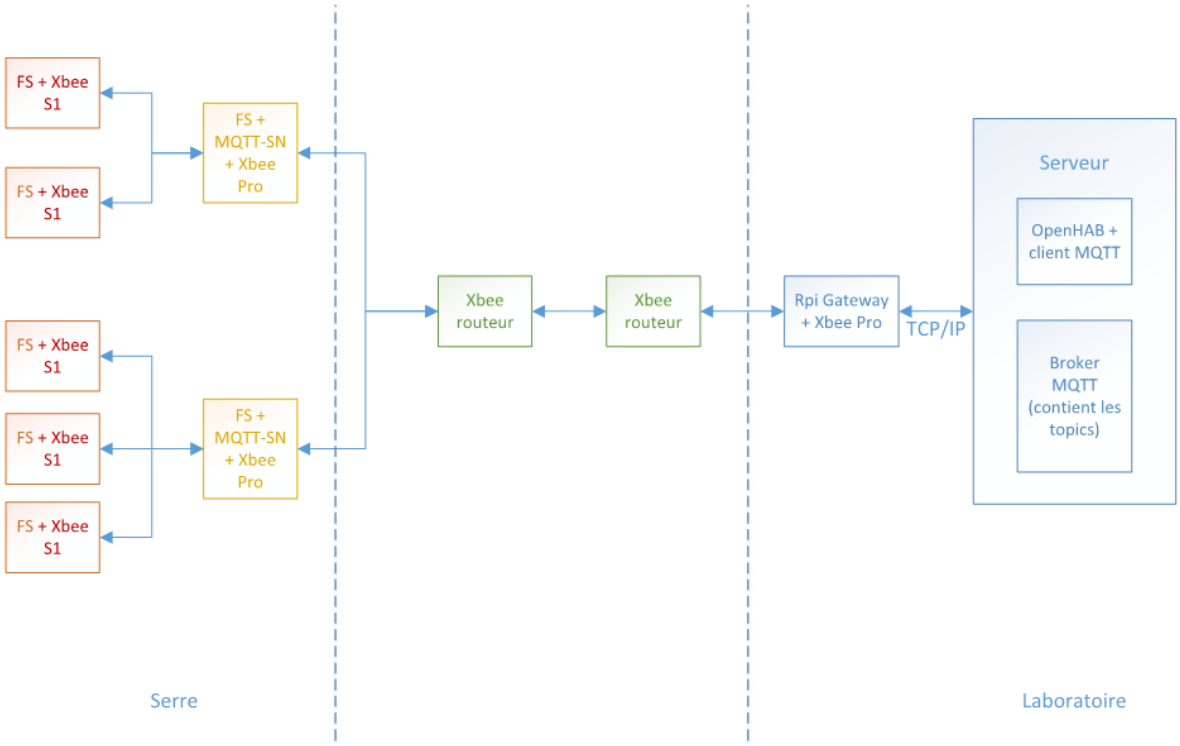

Figure 9: Architecture du réseau de l'iSerre

Le réseau de capteurs est défini tel que dans le diagramme général précédent. En observant ce diagramme de gauche à droite nous distinguons plusieurs niveaux (les protocoles sont indiqués selon le format haut niveau/bas niveau) :

- **1er niveau** : Les capteurs et actionneurs individuels (capteurs de température, d'humidité, relais plaque chauffante, relais humidificateur, contrôleur lumière, etc.) sont chacun implantés sur des cartes KL26Z de Freescale (FS). Chaque capteur/actionneur se comporte différemment dépendamment de son type mais de façon générale les nœuds à ce niveau interagissent directement avec la serre et transmettent l'information au prochain niveau. Les protocoles de communication utilisés sont iSN/DigiMesh.
- **2e niveau (Sinks)** : Pour chaque type de capteurs (température, humidité, actionneurs, etc.), il y a un nœud de synchronisation dédié, abrévié 'sink'. Le sink est implanté dans une carte FS et s'occupe du traitement intermédiaire de l'information

reçue par les capteurs individuels et de l'acheminement de cette information au niveau suivant (Routeurs -> Gateway). Par exemple dans le cas de la température, le sink de température peut faire la moyenne des différentes valeurs prises par chaque capteur individuel et envoyer le résultat vers le serveur. Les sinks peuvent également envoyer aux capteurs des paramètres de configuration (exemple : fréquence d'échantillonnage de la température). Les protocoles utilisés entre ce niveau et le prochain sont MQTT-SN/DigiMesh.

- **3e niveau (routeurs) :** les routeurs Xbee ne servent qu'à acheminer les trames de la serre vers le serveur, et ne font aucun traitement d'information. Le protocole interne de base est utilisé pour le routage (DigiMesh).
- **4e niveau (Gateway) :** La passerelle implémentée sur une Raspberry Pi (RPI) fait la conversion entre MQTT-SN et MQTT. Cette passerelle reçoit des trames MQTT-SN à travers le réseau Xbee et les transforme en requêtes MQTT vers le serveur (broker) à travers le réseau TCP/IP.
- **5e niveau (Serveur) :** Le serveur est une machine exécutant le système d'exploitation Linux dans laquelle se trouvent le broker MQTT ainsi que le serveur openHAB. Le broker MQTT contient et met à jour tous les topics ainsi que la base de données. Le serveur openHAB est un client MQTT qui ne communique qu'avec le broker à travers le réseau TCP/IP.

Interface utilisateur

Une aperçue de l'interface utilisateur montrée dans la figure suivante ainsi qu'une démonstration de l'exécution des cas d'utilisation soulevés ont été présentés au professeur Delagrange afin de valider les fonctionnalités principales qui doivent être présentes dans l'interface de contrôle de la serre. Suite aux discussions avec notre client, nous avons repéré les fonctionnalités suivantes :

Principales

Affichage :

1. Température ambiante à l'intérieur de la serre.
2. Taux d'humidité ambiante.
3. Humidité du sol.
4. Taux de CO_2
5. Taux de O_2
6. Historique de données. Cette fonctionnalité mérite d'une attention spéciale, car elle interagît avec toutes les précédentes. Il s'agit de la vue des informations existantes dans le modèle[5] de l'application.

Contrôle :

1. Température ambiante à l'intérieur de la serre par le contrôle d'une plaque chauffante.
2. Taux d'humidité ambiante par le contrôle d'un humidificateur.
3. Humidité du sol par l'irrigation à l'aide d'une électrovanne.

Supplémentaires

Affichage :

1. Prise de vue synchronisée de l'inventaire de plantes à l'intérieur de la serre.

Contrôle :

1. Couleur et intensité lumineuse du panneau de lumière à l'intérieur de la serre.
2. Caméras à l'intérieure de la serre pour la prise de photo en temps réel.

[5] Le modèle englobe l'information concernant les éléments de l'application selon le patron de conception MVC.

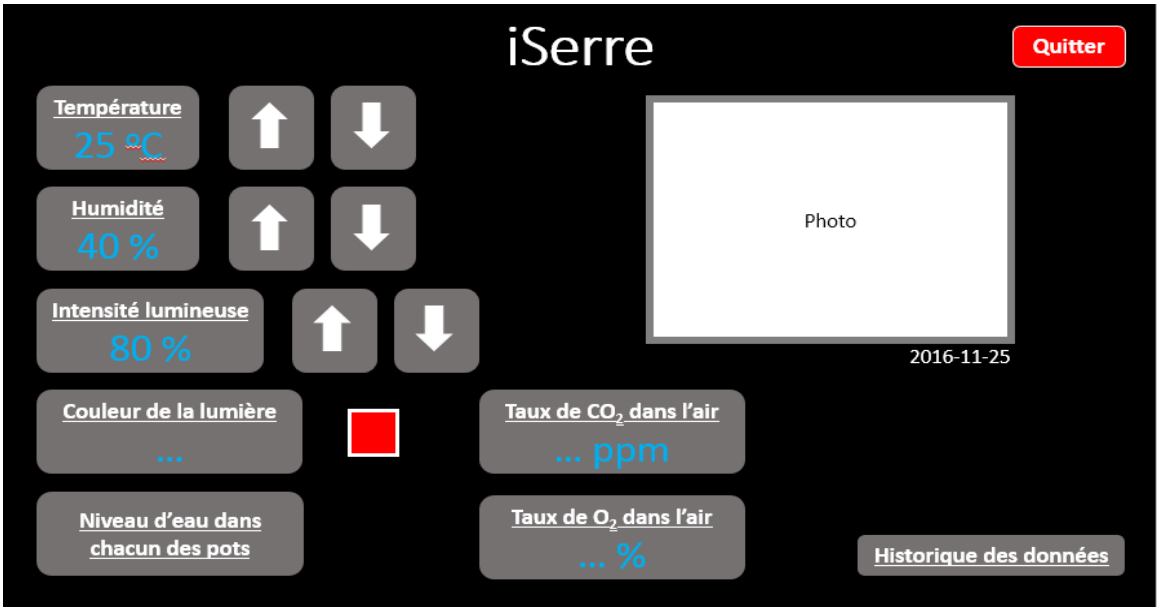

Figure 10. Ébauche de l'interface utilisateur.

La manière dont certaines fonctionnalités ont étés pensées dans l'ébauche de l'interface ne tiennent pas compte de certains aspects pratiques. Comme exemple, nous retrouvons l'affichage de la température en temps réel à l'intérieur de la serre confondue avec la température de consigne. Dans la pratique, n'importe quelle routine de régulation, aussi précise et rapide soit-elle, passe par un écart entre la température ambiante et la température de consigne.

Au niveau de la réalisation, ces fonctionnalités seront implémentées autrement afin de garder la cohésion entre tous les cas d'utilisation de l'interface.

Réalisation

Module des capteurs

Capteur de température (Waterproof DS18B20)

Principe de fonctionnement :

Le capteur de température DS18B20 communique sur un bus 1-Wire avec le microcontrôleur. C'est un bus qui nécessite qu'une seule ligne de donnée et il est basé sur une architecture maitre-esclave. Dans notre cas, le maitre sera la carte FRDM- KL26Z et l'esclave est le capteur de température.

En plus, le capteur contient un « scratchpad » mémoire tampon sécurisé en lecture et/ou en écriture. Cette mémoire devra contenir la valeur de la température dans les deux premiers octets. La valeur de la température mesurée étant une virgule flottante, celle-ci est sous forme d'un flottant avec un exposant, une mantisse et un signe.

La séquence de transaction pour accéder au capteur et ainsi à la valeur de température qui est enregistré dans sa mémoire est :

- L'initialisation : impulsion de réinitialisation transmise par le maitre (carte FRDM KL26Z) suivi par une impulsion de présence de l'esclave (capteur DS18B20)
- Les Commandes ROM du maitre: après que le maitre ait détecté la présence d'un ou plusieurs esclaves sur le bus, il envoie une commande ROM pour choisir l'esclave spécifique avec lequel communiquer (dans le cas où il y'en a plusieurs). Dans notre cas, nous aurons un capteur par carte, donc la commande **Skip ROM** permettant au maitre de communiquer avec tous les esclaves sera utilisée.
- Les « fonctions commandes » du capteur de température DS18B20 : après que le maitre (carte FRDM KL26Z) ait utilisé la commande ROM appropriée pour communiquer avec le capteur, il utilise d'autres fonctions pour lire et écrire sur celui-ci. Nous utiliserons donc les commandes **Convert T** pour

convertir la température et la stocké dans la mémoire du capteur et **Read Scratchpad** pour lire le contenu de la mémoire du capteur.

Implémentation de l'application d'acquisition de la valeur du capteur :

Pour implémenter l'application de l'acquisition de la température provenant du capteur de température DS18B20, nous avons dû implémenter un bus 1-Wire avec un UART. Il existe déjà une librairie générique pour implémenter ce bus, nous l'avons donc réutilisé avec le composant **fsl_uart** de Processor Expert qui va nous permettre d'établir la communication entre la carte et le capteur. Nous avons donc implémenté les fonctions d'initialisation de l'Uart, de réglage du baud rate, de lecture et d'écriture sur le composant connecté à l'Uart en question. Dans la librairie qui nous ait fournit, ces fonctions sont utilisées pour l'implémentation de celle qui établissent la communication 1-wire directement. Il s'agit de la lecture d'un octet du capteur par le maitre (**OneWire_Read),** de l'écriture d'un octet par le maitre sur le capteur (**OneWire_Write)**, et de la réinitialisation de la communication (**OneWire_Reset**).

Une fois la communication 1-wire implémenté, nous avons créé une application de lecture de la valeur du capteur qui utilise la librairie générique 1-wire que nous avons complété. Nous avons donc implémenté la séquence de transaction d'accès au capteur. Ensuite la valeur des 2 octets de la mémoire sont convertis en valeur décimale afin d'avoir la valeur de la température exacte.

Capteur d'humidité relative de l'air (DHT22)

Principe de fonctionnement :

Le capteur d'humidité relative de l'air DHT22 utilise un protocole de communication propriétaire avec un seul fil de donnée et nécessitant un timing précis. C'est ainsi que les signaux envoyés par la carte FRDM KL26Z se distingue de par leur durée, de même les réponses du capteur seront interprétées en fonction de la durée du signal de réception.

Ainsi la séquence de communication s'effectue comme suit :

- La FRDM-KL26Z maintient à l'état bas le bus de données pendant 800 µs
- Le DHT22 répond au maître, en maintenant le bus de données à l'état bas pendant 80 µs, puis à l'état haut pendant 80 µs en guise d'accusé de réception.
- Le DHT22 transmet une série de 40 bits (5 octets). Lorsque le signal de réception dure 30 us, alors le bit reçu est interprété comme 1 sinon le bit reçu est 0.

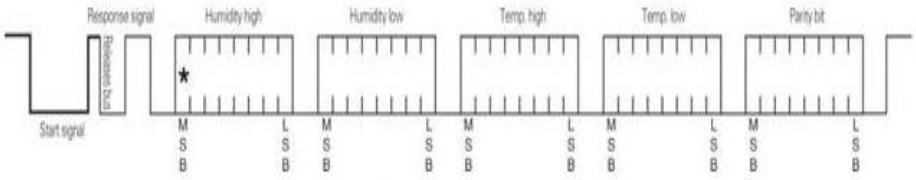

Figure 10 : **Timing du protocole de communication DHT22**

Implémentation de l'application d'acquisition de la valeur du capteur :

L'application s'est faite principalement avec les composants **BitIo** et **WAIT** de Processor Expert. BitIo permet de configurer le capteur en entrée ou en sortie et de transmettre la valeur du bit qui a été interprété (dépendamment du timing). Nous avons donc implémenté une fonction d'établit le protocole de communication avec les timings, ensuite nous avons interprété tous les bits du capteur afin de les stocker dans un tableau. En fin nous avons converti les valeurs binaires du tableau en décimale pour avoir l'humidité relative de l'air.

Capteur de CO2 (MG811-SEN0159)

Principe de fonctionnement :

Ce capteur possède une interface analogique et en fonction de la concentration du CO2 que détecte le capteur une sortie digitale est enregistré et est transmise. La variation de la concentration du CO2 influe sur la sortie en tension du capteur et la courbe suivante montre cette variation.

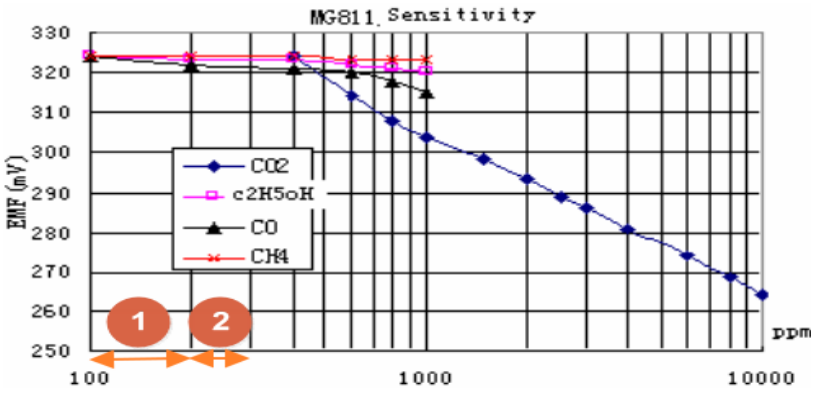

Figure 11 : **Variation de la tension de sortie du capteur en fonction de la concentration du CO2**

Implémentation de l'application d'acquisition de la valeur du capteur :

Pour acquérir la valeur du capteur en utilisant la carte FRDM KL26Z nous avons les composants processor expert suivant :

- ADC: un convertisseur analogique numérique qui vient avec des fonctions de conversion et de mesure.
- Wait : pour prendre plusieurs échantillons de la valeur du CO2
- RingBufferUInt8 et Serial_LDD : pour l'affichage sur une console (TeraTerm dans notre cas).

Ensuite l'application d'acquisition implémente une fonction de lecture de la tension aux bornes du capteur (Celle-ci utilise les fonctions de l'ADC). Il y'a aussi une fonction de calcul de la concentration du CO2 en fonction du voltage qui a été lue.

Capteur d'humidité du sol (SEN0114) :

Principe de fonctionnement :

Le capteur d'humidité du sol possède une sortie analogique avec des plages de valeurs en fonction de l'humidité du sol :

- 0 ~300 : pour un sol sec
- 300~700 : pour un sol humide
- 700~950 : pour un sol mouillé

Implémentation de l'application d'acquisition de la valeur du capteur :

L'acquisition de la valeur analogique de sortie du capteur a été faite en utilisant le composant fsl_adc16 qui est un convertisseur analogique à numérique dont les fonctions de calibration, de mesure et de conversion ont été utilisé afin de lire la valeur de sortie du capteur et de l'afficher sur une console.

Actionneur : Relais

Principe de fonctionnement :

Pour contrôler les dispositifs de la plaque chauffante, de l'humidificateur, de l'électrovanne et du ventilateur avec la carte FRDM KL26Z nous avons implémenté un relai pour ouvrir et fermer les différents circuits. Pour cela une tension de 0V est émise pour désactiver le relai et une autre de 5v pour activer le relai. La désactivation et l'activation du relai agit directement sur le dispositif qui y est connecté.

Le circuit du relai ci-dessus a été conçu pour être connecté à la carte FRDM KL26Z.

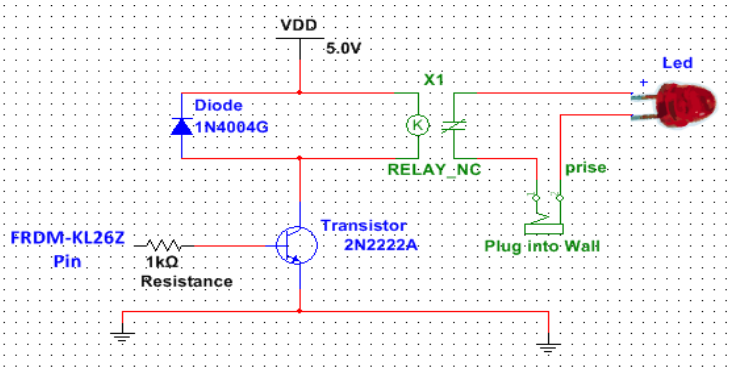

Figure 12 : **Montage du relai.**

Implémentation:

Pour créer le programme qui va implémenter ce relai, nous avons utilisé un **Fsl_gpio** qui est un composant avec des fonctionnalités du port entrée-sortie sur lequel nous pouvons écrire des valeurs de tensions (0 ou 5v).

Actionneur : Panneau de LED :

Principe de fonctionnement :

Le panneau de lumière a été conçu avec plusieurs rubans de LED RGB. Les LEDS RGB fonctionnent avec une télécommande (contrôleur RGB), elles seront alimentées par le secteur et seront contrôlés par une carte Arduino. Le panneau de lumière peut être contrôlé suivant la couleur et l'intensité.

Nous avons donc réalisé un programme qui affiche sur une console les codes de la télécommande pour chaque couleur ou combinaison de couleur (RGB), ensuite ces différents codes ont été utilisés pour faire varier les couleurs mais aussi l'intensité de la lumière du panneau de LED à travers un programme.

Le panneau de LED est contrôlé à distance via un nœud réseau actionneur. Un module Xbee connecté sur la carte Arduino Mega 2560 sur laquelle est connectée la diode émettrice infrarouge, va se charger de recevoir les commandes envoyées depuis l'interface utilisateur. La diode émettrice va se chargée de transmettre les codes de couleur reçus aux récepteurs infrarouges du panneau de LED.

Implémentation:

L'implémentation du programme d'envoie des codes de la télécommande utilise la librairie IRremote qui contient les classes suivantes :

- « IRrecv » : qui contient les fonctions pour la réception infrarouge.
- « IRsend » : qui contient les fonctions pour l'émission de codes infrarouges.

La fonction sendNEC(unsigned long data, int nb) de la classe « IRsend » a été utilisée pour l'émission du code « data » en « nb » bits.

Module de Communication

En considérant l'architecture de communication établie, les protocoles suivants ont dû être implémentés afin de véhiculer efficacement l'information dans la serre :

- MQTT
- MQTT-SN
- Xbee/DigiMesh
- iSN (iSerre Sensor Network)

MQTT

MQTT (MQ Telemetry Transport) est un protocole léger de haut-niveau implanté par-dessus le protocole TCP/IP. Il est basé sur le principe de la publication-souscription. On distingue dans ce protocole des clients et le broker/serveur. Le broker contient et met à jour des topics auxquels les clients peuvent s'abonner. Les clients peuvent ainsi s'abonner ou publier des messages sur un topic. Tout client abonné à un topic reçoit tous les messages que d'autres clients publient sur ce topic.

Le broker utilisé dans le cadre de ce projet est le broker open-source appelé « mosquitto » [8]. Il est installé dans la machine serveur au laboratoire et implémente MQTT. Le serveur openHAB est également installé dans cette machine, et agit comme client MQTT.

MQTT-SN

MQTT-SN est une version plus légère de MQTT, développée pour une utilisation du protocole dans les réseaux de capteurs, qui sont typiquement très limités au niveau de leurs ressources. MQTT est en effet implanté par-dessus le TCP/IP, qui est assez lourd pour être implanté de tels réseaux.

Trois types de composants sont définis selon la spécification de MQTT-SN : Le client, le *forwarder* (routeurs), et le *Gateway* (passerelle). Les clients sont typiquement des capteurs (les sinks dans notre cas) et le Gateway fait la traduction MQTT-SN vers MQTT. Le schéma suivant est tiré de la documentation de MQTT-SN [9].

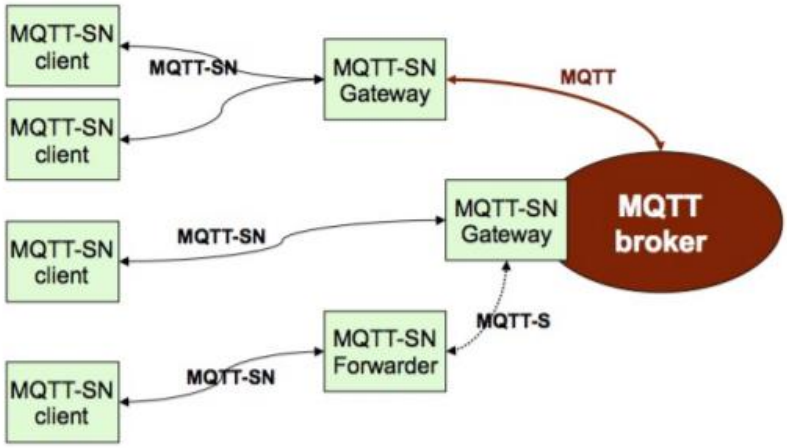

Figure 13: **Architecture MQTT-SN**

L'implémentation de la passerelle sur la RPI est faite en compilant et en exécutant le projet 'MQTT-SN' open-source disponible sur GitHub [10].

L'implémentation du client MQTT-SN sur les FS faisant office de sink s'est également faite à partir du même code, auquel des modifications ont dû être apportées pour adapter le projet au matériel de la carte KL26Z.

Une fois le broker installé sur la machine, il suffit d'exécuter la passerelle sur la RPI en lui passant en paramètre l'adresse IP du broker. Le client MQTT-SN de son côté est tout simplement exécuté, le protocole est défini de la manière suivante :

- Le client lance un message *searchGateway* en diffusion à travers le réseau Xbee (prolongé par les routeurs).
- Une fois que la passerelle reçoit ce message, il renvoie ses informations de connexion (message *gwInfo*) au client.
- Le client peut ensuite souscrire ou publier sur des topics, en ne communiquant qu'avec la passerelle, qui elle transmettra les messages au broker.

Xbee / DigiMesh

Les différentes cartes (Freescale, Raspberry Pi) communiquent à l'aide de modules radio

Xbee qui permettent d'établir une communication sans fil. On distingue principalement deux types de modules Xbee : ceux de 1 mW d'assez courte portée qui sont utilités pour faire communiquer les composants à l'intérieur de la serre et ceux de 100 mW de plus longue portée qui font office de routeur.

Ces modules ont différents modes de fonctionnement mais par souci de simplicité nous allons utiliser le protocole de haut-niveau DigiMesh ayant les caractéristiques suivantes :

Il n'y a pas de coordinateur, tous les nœuds sont des nœuds DigiMesh, et peuvent être des routeurs ou des nœuds terminaux.

Nous avons implémenté pour les capteurs/actionneurs un module Xbee en C capable d'effectuer les opérations principales (envoyer une trame, recevoir une trame) en utilisant les fonctionnalités des UART présentes sur la carte KL26Z.

Le code MQTT-SN repris contient par ailleurs un module Xbee similaire, qui a dû être adapté à la carte KL26Z.

iSN

Le protocole suivant est défini pour la communication entre les sinks et les capteurs (*les codes hexadécimaux indiqués sont sujet à changement dans l'implémentation finale*) :

Trames (frame_type)	Description
Search_sink **(0x01)**	Envoyé par le nœud (capteur ou actionneur) pour signaler sa présence au sink de son type. Le contenu est le type du capteur (température, actionneur température, humidité, ...).
Search_sink_ack (0x05)	Envoyé par le sink en réponse à un search_sink dans le but d'indiquer au capteur l'adresse du sink.
Config (0x02)	*Trame réservée aux capteurs.* Envoyé du sink vers le capteur et contenant une donnée de configuration. Le contenu dans l'ordre est : nombre de paramètres de configuration, code du paramètre 1, valeur du paramètre 1, …, code du paramètre n, valeur du paramètre n. Cette trame sert également d'acquittement : elle est envoyée en réponse au search_sink pour fournir une config. initiale au capteur.
Commande (0x03)	*Trame réservée aux actionneurs (en réception)*. Contient la commande et envoyée par le sink des actionneurs vers le nœud actionneur.

Mesure (0x04)	Trame de données envoyée au sink contenant la valeur mesurée par le capteur.

Spécification des trames

- **Search_sink :**

Frame_type	Device_type

Frame_type (1 octet) : 0x01 pour les trames Search_sink.

Device_type (1 octet) : Type du nœud.

Les nœuds de type capteur auront des codes compris entre 0x01 et 0x7F.

Ceux de type actionneur auront des codes compris entre 0x81 et 0xFF.

Type	**Code**
Capteur température	0x01
Capteur humidité	0x02
Actionneur température (plaque chauffante)	0x81
Actionneur humidité (humidificateur)	0x82

- **Search_sink_ack :**

Frame_type

Frame_type (1 octet) : 0x05 pour les trames Search_sink_ack.

- **Config :**

Frame_type	Count	Code_1	Value_1	…	Code_n	Value_n

Frame_type (1 octet) : 0x02 pour les trames Config.

Count (1 octet) : Nombre de paramètres de configuration (couples code/value) envoyés.

Code (1 octet) : Code d'un paramètre de configuration. Pour chaque type de capteur il y'a plusieurs paramètres de configuration et chaque paramètre a un code assigné. (Ex pour le capteur de température le paramètre 'fréquence d'échantillonnage' a pour code 0x01.)

Value (2 octets) : Valeur du paramètre de configuration pour le code précédent.

- **Commande :**

Frame_type	Commande

Note : L'adresse ou le type d'actionneur n'est pas spécifiée car il est supposé que le sink contrôle le destinataire (l'adresse est donc déjà dans la trame Xbee).

Frame_type (1 octet) : 0x03 pour les trames Commande.

Commande (1 octet) : Commande de l'actionneur.

- **Mesure :**

Frame_type	Data

Frame_type (1 octet) : 0x04 pour les trames de mesures (Data).

*Data (4 octets) : Valeur mesurée par le capteur (toutes les données envoyées par les capteurs seront du type **float**).*

Fonctionnalités implémentées

- Module Xbee pour les capteurs/actionneurs : permet à partir de la carte KL26Z de générer et d'envoyer des trames (soit en diffusion soit en spécifiant l'adresse) et de décoder des trames reçues.
  Note : Ce module a été réalisé en utilisant le kit de développement Kinetis SDK (v1.3).
- MQTT-SN : l'implémentation de client MQTT-SN sur une carte KL26Z a été réalisée et implantée sur les sink. Les sinks de température, d'humidité et d'actionneurs ont ainsi été réalisés (utilisant les protocoles iSN et MQTT-SN par-dessus le DigiMesh).
- Gateway MQTT-SN -> MQTT : la passerelle a été compilée et exécutée sur une Raspberry Pi 2.

- MQTT : Installation du broker, définition des différents topics et spécification du format des messages dans chaque topic (voir documentation technique).
- iSN : le protocole a été partiellement implémenté afin de faire fonctionner toute la serre à temps pour une démonstration. Seules les trames de commande et de mesure sont implémentées et le fonctionnement actuel est le suivant :
    - Toutes les adresses des modules Xbee sont hardcodées (sinks, capteurs, actionneurs).
    - Les capteurs envoient en permanence (trame *iSN_Mesure*) la valeur mesurée au sink correspondant).
    - Le sink fait la moyenne de toutes les mesures qu'il a reçues et publie sur le topic correspondant.
    - Le sink des actionneurs est abonné aux différents topics 'actionneur'. Lorsqu'un message est publié sur un des topics, il envoie une trame iSN_Command à l'actionneur correspondant et celui-ci s'active ou se désactive.

Ainsi, le seul point à compléter pour la communication serait de terminer l'implémentation du protocole iSN, afin d'introduire une routine de découverte des différents composants et d'éliminer la contrainte de devoir coder en dur les adresses des modules Xbee.

Interface utilisateur

Ce module constitue le dernier maillon de la chaîne. L'interface est chargée d'interpréter et présenter de manière significative pour l'utilisateur toute l'information générée par le réseau de capteurs et acheminée par le module de communication. Elle est également responsable de livrer les informations nécessaires au module de communication pour le contrôle des dispositifs à l'intérieur de la serre suite à la commande de l'utilisateur.

Le « *Front End* » de la solution a été implémentée à l'aide du logiciel Openhab. Il s'agit d'un logiciel permettant l'intégration de différents systèmes et technologies d'automatisation résidentielle en une seule solution, permettant ainsi de configurer et de gérer les règles d'automatisation. De plus, ce logiciel offre des interfaces utilisateur uniformes sans toutefois se préoccuper des divers protocoles et spécifications des systèmes intervenant dans la solution. Nous le définissons comme un système de systèmes. Même s'il s'agit d'un logiciel d'automatisation résidentielle, nous avons été en mesure de l'utiliser efficacement dans le cadre de notre projet. Ce dernier nous a permis d'offrir une interface utilisateur professionnelle et complète.

OpenHAB supporte un langage spécifique dont la morphologie est celle d'un langage de configuration. L'implémentation des différentes fonctionnalités passe par la compréhension et la maîtrise de la syntaxe de ce langage, inconnu auparavant.

Plus concrètement, openHAB intègre différentes technologies à l'aide des « *addons* ». Ces fichiers, implémentés en Java, permettent à openHAB de supporter un service spécifique à une technologie ou un dispositif particulier.

Au niveau de l'implémentation de certaines fonctionnalités, il est nécessaire d'accéder au code source Java des « *addons* » afin de comprendre la manière dont un service a été implémenté. Cette action est nécessaire aussi pour la compréhension de certains messages d'erreur au niveau du serveur.

La maîtrise du langage de configuration d'openHAB et son interprétation nous ont permis d'implémenter l'interface ainsi que les actions respectives de manière à offrir des résultats professionnels.

Bien que l'interface utilisateur implémentée est basée sur une solution HTML/JS, elle simule une application IPhone optimisée pour un fonctionnement tactile avec une aperçue sobre.

Fonctionnalités implémentées

Nous avons été en mesure d'implémenter toutes les fonctionnalités d'affichage et de contrôle définies comme principales.

Dans la vue principale de notre interface, nous retrouvons la section réseau de capteurs, qui correspond à la lecture des grandeurs prélevées par de tous les capteurs qui seront déployés à l'intérieur de la serre (température, humidité de l'air, humidité du sol, gaz carbonique, méthane et oxygène),

La section de contrôle prévoit le contrôle de la température, de l'humidité de l'air, de l'humidité du sol ainsi que le contrôle de la lumière.

Nous retrouvons également la section de l'historique de données comme prévu.

Nous avons également implémenté la vue de fonctionnalité de prise de photos à l'intérieur d'une section dédiée à la phénologie. Le contrôleur de cette fonctionnalité n'a pas pu être implémenté.

1. **Réseau de capteurs**

Au niveau de la section du réseau de capteurs, toutes les mesures prélevées par chacun des capteurs sont visibles en temps réel à partir de la page titre. Toutes les unités de mesure sont également présentées à la droite des données de chacun des capteurs, permettant ainsi une précision optimale. De plus, il est possible de sélectionner un capteur, ce qui nous apporte dans une nouvelle fenêtre montrant un graphique contenant toutes les données recueillies dans la dernière heure. Ces graphiques permettent une analyse à court terme des résultats recueillis dans la serre. Tous les capteurs offrent un graphique montrant ainsi les résultats recueillis depuis la dernière heure passée. Dans le cadre de notre projet, seul le capteur de température a été déployé pour le livrable final. Par contre, cela ne nous a certainement pas empêchés

d'implémenter la vue et les contrôleurs de tous les autres capteurs. Si le déploiement des autres capteurs est fait, il suffit de publier les mesures prélavées dans les « *topics* » indiqués ou changer leurs noms dans le fichier « *projet.items* ». Tout le reste du code est déjà implémenté.

2. Contrôle de dispositifs

Au niveau de la section de contrôle, la consigne de la température, de l'humidité de l'air et de l'humidité du sol sont modifiables. Il est possible d'augmenter ou de diminuer ces valeurs. Ainsi, le résultat de la consigne respective sera affiché en temps réel tout en indiquant l'unité de mesure associée.

De plus, des règles de régulation tout ou rien ont été implémentés afin de garantir le contrôle par l'utilisateur des paramètres mentionnés.

Dans ces règles simples, nous faisons une comparaison entre la valeur de la consigne et la valeur immédiate prélevée par le capteur pour ainsi activer les dispositifs pertinents à l'aide d'un relais.

Dans le cas du contrôle de température, l'augmentation de cette grandeur passe par l'activation d'une plaque chauffante. Dans le cas du contrôle de l'humidité de l'air, le taux d'humidité peut être augmenté à l'aide d'un humidificateur. Enfin, dans le cas du contrôle de l'humidité du sol, l'irrigation est prise en charge par une électrovanne, permettant ainsi le contrôle du niveau de l'eau dans les pots présents dans la serre.

Quant aux stratégies de régulation, dans une situation où la valeur de la consigne serait à 26 ^{0}C alors que la température immédiate dans la serre serait de 19 ^{0}C, la règle faite en sorte d'envoyer la valeur « 1 » au relais voulant ainsi dire d'allumer la plaque chauffante pour augmenter la température. Dans le cas contraire, si la valeur de la consigne était inférieure à la température immédiate dans la serre, la règle ferait en sorte d'envoyer la valeur « 0 » voulant ainsi dire d'éteindre la plaque chauffante pour diminuer ou stabiliser la température dans la serre. Cette logique est également implémentée pour le contrôle de l'humidité de l'air ainsi que pour le contrôle de l'humidité du sol dans le fichier « *projet.rules* » de OpenHAB. Dans notre cas, étant donné que seul le capteur de

température était fonctionnel dans le livrable final, seule la consigne de température était prise en compte. Par contre, lors du déploiement de nouveaux capteurs, seuls les noms des « *topics* » seront à changer dans le fichier « *projet.items* » et le fichier « *projet.rules* » afin de publier les valeurs de contrôle dans le topic spécifique à un dispositif particulier, car la logique est déjà implémentée.

En tout moment la stratégie de régulation tout ou rien ne peut être aisément remplacé par un contrôleur PID pour une régulation plus précise.

Bref, les trois premiers contrôles n'offrent que la possibilité d'ajuster une consigne à l'aide d'un bouton augmenter et d'un bouton diminuer. Par contre, le contrôle de lumière offre bien plus de possibilités, d'où l'état de la lumière (ON ou OFF), l'intensité de la lumière (10% à 100%) ainsi que la couleur de la lumière (Rouge, Vert, Bleue ou Blanc). Lors de la sélection du contrôle de lumière, une nouvelle fenêtre apparaît offrant ainsi la possibilité d'allumer ou éteindre la lumière, d'augmenter ou de diminuer l'intensité lumineuse ou alors changer la couleur. Si l'option de changer la couleur est sélectionnée, une nouvelle fenêtre apparaît montrant ainsi le choix de quatre couleurs. La couleur ayant comme indication « *UP* » est celle présentement sélectionnée, alors que les trois autres couleurs auront comme indication « *DOWN* ». Heureusement, le contrôle de la lumière a été totalement implémenté et il est fonctionnel dans le système livré.

3. Historique

Au niveau de la section de l'historique de données, la sélection de cette dernière nous apporte à une nouvelle fenêtre. Cette nouvelle fenêtre nous permet de sélectionner le type de capteur dont nous voulons observer les données ainsi que de sélectionner l'échelle du temps (1 heure, 4 heures, 8 heures, 12 heures, 1 journée, 3 journées, 1 semaine, 2 semaines et 1 mois). Suite à la sélection de ces deux paramètres, la sélection de « Afficher Graphique » permet l'affichage du graphique montrant ainsi un graphique avec les données du capteur sélectionné sur la période de temps définie. Cette section est sans doute une partie très importante de l'interface utilisateur dans le

cas où l'utilisateur voudrait faire une analyse de toutes les grandeurs prélevées depuis un certain temps. Cette section permet de faire une étude complète sur tous les paramètres de la serre.

À des fins de description visuelle, voici un aperçu de la page titre de notre interface utilisateur :

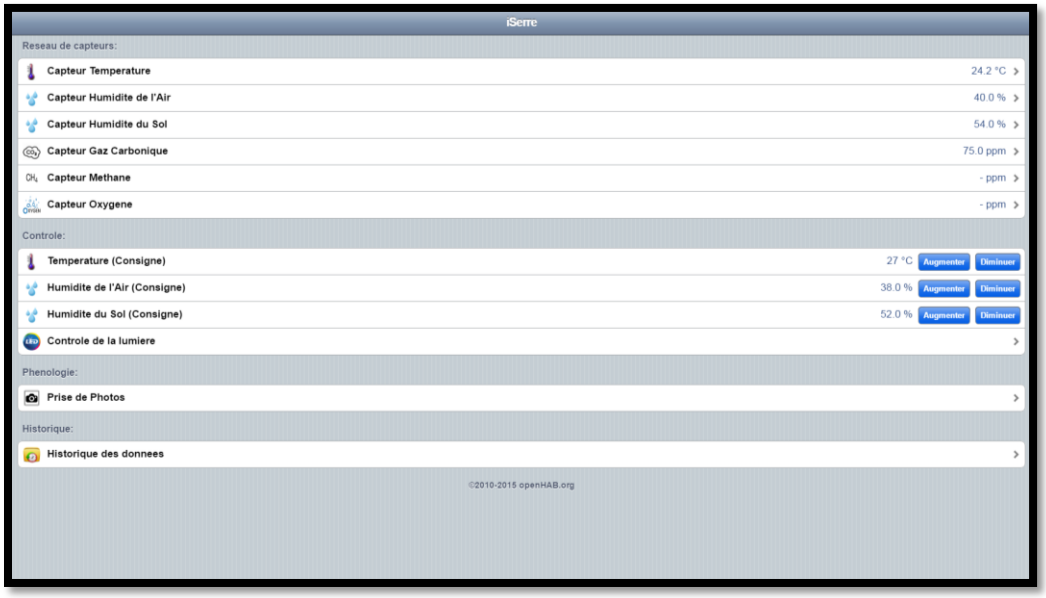

Figure 15: Interface Utilisateur

Évolutivité de l'interface

Bien évidemment, nous considérons que l'interface implémentée est sensible à plusieurs modifications avant de constituer un livrable final. Certains aspects sont manquants ou incomplets. La section « *Réseau de capteurs* » est sans doute complète et fonctionnelle. Par contre, les sections « *Contrôle* », « *Phénologie* » et « *Historique* » peuvent être optimisées.

Au niveau de la section de contrôle, seule une optimisation des règles est nécessaire, car le fonctionnement général est totalement complet et fonctionnel. Par exemple, dans le cas du contrôle de la température, notre logique fait en sorte que lorsque la valeur de la consigne est supérieure à la température immédiate, la valeur « 1 » est envoyée voulant ainsi dire d'allumer la plaque chauffante. Par contre, la plaque

chauffante ne s'éteindra que lorsque la température immédiate de la serre sera égale à la valeur de la consigne. Il ne fait certainement pas oublier le fait qu'une plaque chauffante continue à chauffer pendant plusieurs minutes même après son arrêt, car la plaque reste chaude. Il reste donc à faire une série de tests permettant ainsi de déterminer le temps nécessaire à la plaque chauffante à être totalement froide, c'est-à-dire le temps nécessaire, suite à l'arrêt de cette dernière, afin que la serre ne soit plus chauffée. Cela nous permettrait donc de faire une règle du genre : Si la valeur de la température immédiate est égale à la valeur de la consigne moins 3 degrés, envoyer un « 0 » voulant ainsi dire d'éteindre la plaque chauffante. Malheureusement, cette série de tests n'a pas été effectuée due au déploiement tardif. Par contre, les règles du contrôle de l'humidité de l'air et de l'humidité du sol sont complètes, mais non fonctionnelles due à l'absence du capteur d'humidité de l'air et du capteur d'humidité du sol à l'intérieur de la serre.

Éventuellement, il serait convenable de remplacer le contrôle tout ou rien par un contrôleur PID pour une régulation plus précise.

Au niveau de la section de la phénologie, l'implémentation est quasi-absente. Dû au manque de temps et à l'absence d'une caméra à l'intérieur de la serre, nous n'avons pas été en mesure d'implémenter convenablement cette section. Nous avons qu'implémenté une petite partie démontrant ainsi notre idée générale. Seuls trois paramètres peuvent être sélectionnés dans cette section, d'où l'année, le mois et le jour. Notre idée était assez simple, c'est-à-dire de faire en sorte que l'utilisateur puisse sélectionner l'année, le mois et le jour pour ainsi faire afficher la photo prise à cette date. Il reste donc à implémenter les règles permettant d'interpréter la date sélectionnée par l'utilisateur ainsi qu'à compléter l'implémentation au niveau de l'affichage de la photo à partir d'un dépôt qui viendrait recueillir les images prélevées de façon synchronisée par une ou plusieurs caméras présentes dans la serre.

Au niveau de la section de l'historique de données, seule une optimisation serait nécessaire. Dans notre cas, nous pouvons faire afficher un graphique d'un certain capteur selon une échelle de temps définie. Par contre, nous aurions aimé être en

mesure d'afficher également le tableau des données associées aux graphiques et permettre sa sauvegarde par l'utilisateur dans le format approprié. Bien que cette section soit totalement fonctionnelle, une simple optimisation reste à être faite.

Enfin, certains ajouts et optimisations peuvent sans doute être faits afin d'offrir une meilleure expérience utilisateur et converger vers l'interface d'un livrable finale de la serre. Nous avons implémenté une interface très simple permettant ainsi d'afficher des données prélevées par le réseau de capteurs, de contrôler certains paramètres et d'observer les données recueillies.

Dans la section interface utilisateur de la documentation technique, il se trouve toute information nécessaire pour faciliter l'évolutivité de l'interface.

Conclusion

Cette expérience nous a permis de nous familiariser avec le cadre d'un projet professionnel en ingénierie. Loin de nous limiter à la solution livrée, nous devons tenir compte des compétences techniques et professionnelles acquises.

La solution livrée est le résultat de plusieurs technologies, protocoles et même langages de programmation intégrés.

Nous avons été introduits à la programmation embarquée en langage C d'un microcontrôleur très répandu en industrie.

Nous avons été portés à la compréhension des aspects de l'intégration de plusieurs modules afin d'offrir une solution finale unique.

Ce projet, d'une envergure sans égale à d'autres projets réalisés dans notre cheminement, demandait directement et indirectement plusieurs des connaissances acquises. Le travail d'équipe a également été nécessaire dans le but d'offrir une solution fonctionnelle et optimale. Sans organisation, ce projet ne se serait sans doute pas déroulé ainsi.

Bibliographie

[1] Capteur de CO_2

https://www.dfrobot.com/wiki/index.php/CO2_Sensor_SKU:SEN0159

[2] Capteur de O_2

http://wiki.seeedstudio.com/wiki/Grove_-_Gas_Sensor(O%E2%82%82)

[3] Capteur de d'humidité du sol

https://www.dfrobot.com/wiki/index.php?title=Moisture_Sensor_(SKU:SEN0114)

[4] Capteur de température

https://www.dfrobot.com/index.php?route=product/product&product_id=689

[5] Capteur d'humidité et de température

http://www.robotshop.com/eu/fr/capteur-temperature-humidite-dht22.html

[6] Relais électronique

http://www.fotek.com.hk/solid/SSR-1.htm

[7] Electrovanne

http://www.ebay.ca/itm/Orbit-57253-3-Valve-Heavy-Duty-Preassembled-Manifold-New-/371216701062

[8] Mosquitto, "An Open Source MQTT v3.1/v3.1.1 Broker."

https://mosquitto.org/

[9] "MQTT For Sensor Networks (MQTT-SN) Protocol Specification."

http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf.

[10] T. Yamaguchi, "MQTT-SN".

https://github.com/ty4tw/MQTT-SN.

Annexe : Code

L'ensemble du code développé est disponible à l'adresse suivante :

https://github.com/Julien-Combattelli/iSerre