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.

Documentation technique

Travail réalisé par :

Arielle Fanbeu Sipeyou

Cheikh Charles Diaw

Francis Normand

Julien Combattelli

Kadiatou Diallo

Zineb El Moutia

TABLE DES MATIÈRES

Le but de ce document est d'assurer l'évolutivité du projet iSerre en fournissant tous les détails techniques nécessaires pour l'ajout, modification ou suppression de fonctionnalités.

CAPTEURS

Il est question ici de présenter comment effectuer la connexion ainsi que les tests des différents capteurs et actionneurs utilisés pour la mesure et le contrôle des paramètres dans la serre.

IMPORTER LES PROJETS SUR KINETIS DESIGN STUDIO

Les projets relatifs à chacun des capteurs sont disponibles sur le lien Github fourni. Afin d'exécuter un des projets relatifs à un capteur spécifique ou à un actionneur donné, il va falloir télécharger le projet en format zip puis, le dé-zipper et l'ajouter dans le l'espace de travail via lequel *Kinetis Design Studio* est démarré.

Ensuite, il faut démarrer l'outil de développement *Kinetis Design Studio* et sélectionner l'espace de travail dans lequel le projet a été ajouté. Une fois que Kinetis Design Studio est ouvert, sélectionnez **File – Import – Existing Projects into Workspace** puis cliquez sur **Next.** Dans la boite de dialogue Import Projects, sélectionnez le répertoire racine du projet que vous souhaitez importer, cliquez sur OK puis sur Finish pour terminer le processus.

Le projet importé est alors ajouté et peut maintenant être visualisé dans l'explorateur de projet **Project Explorer**.

CONNECTER LES COMPOSANTS SUR LA FREESCALE

Une fois que les projets ont été importés sur l'espace de travail de Kinetis Design Studio, il est primordial de connecter les capteurs ou les actionneurs associés à la carte Freescale FRDM-KL26Z pour une éventuelle simulation et exécution de chacun des projets. Sont donc présentées ci-dessous les méthodes de connexions des capteurs et actionneurs utilisés.

CAPTEUR DE TEMPÉRATURE DS18B20

Le capteur de température DS18B20 doit être connecté sur la carte FRDM-KL26Z avec un module radio XBEE qui va se charger d'envoyer la valeur de la température acquise. Afin d'éviter toute forme de conflits, le module XBEE doit être connecté sur UART1 tandis que le capteur est connecté sur UART2. Ainsi, les deux pins de signal transmetteur et émetteur du capteur DS18B20 doivent être connectées respectivement sur la pin PTD3 et PTD4 de la carte Freescale. Les pins transmetteur et émetteur du module XBEE doivent être connectées respectivement sur la pin PTE0 et PTE1.

Il ne faut surtout pas oublier d'alimenter le module Xbee et le capteur DS18B20 sur du 3.3 V et de les brancher à la masse. Une fois les branchements effectués, le programme pourra être compilé puis télécharger sur la carte Freescale.

CAPTEUR D'HUMIDITÉ DU SOL

Le capteur d'humidité doit être connecté sur la carte FRDM-KL26Z avec un module radio XBEE qui va se charger d'envoyer la valeur d'humidité acquise. Afin d'éviter toute forme de conflits, le module XBEE doit être connecté sur UART1 tandis que le capteur est connecté sur l'une des broches analogiques ADC0 disponible sur la carte.

Ainsi, les pins transmetteur et émetteur du module XBEE doivent être connectées respectivement sur la pin PTE0 et PTE1. Le capteur d'humidité du sol doit être connecté à la pin PTC0 de la carte Freescale. Il ne faut surtout pas oublier d'alimenter le module Xbee et le capteur sur du 3.3 V et de les brancher à la masse. Dès lors que les branchements sont effectués, le programme pourra être compilé puis télécharger sur la carte Freescale.

CAPTEUR D'HUMIDITÉ RELATIVE DE L'AIR DTH22

Le capteur d'humidité relative de l'air DTH22 possède une broche de signal. Celle-ci est connectée sur UART1 dans la pin PTE0 de la carte Freescale. Afin de visualiser les différentes valeurs d'humidité acquises par le capteur, vous devez ouvrir une console comme Tera Term ou Putty. Il ne faut surtout pas oublier d'alimenter le capteur sur du 3.3 V et de le brancher à la masse. Dès lors que les branchements sont effectués, le programme pourra être compilé puis télécharger sur la carte Freescale.

Le programme de ce capteur utilise des composants de Processor Expert. Afin de transférer les valeurs acquises par le capteur DTH22 via un module XBEE, il va falloir utiliser des composants KSDK étant donné que le protocole XBEE disponible a été programmé avec des composants KSDK. Le challenge ici est de trouver des composants KSDK assez similaires aux composants Processor Expert utilisés.

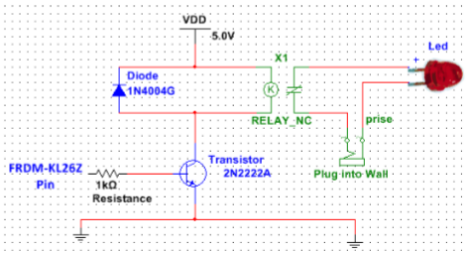
CAPTEUR DE CO2

Le capteur de CO2 tout comme le capteur d'humidité du sol est un composant analogique et doit de ce fait être connecté à un convertisseur analogique ADC0 de la carte Freescale. Ainsi, le capteur de CO2 doit être connecté sur la pin PTB2 de la carte. Il ne faut surtout pas oublier d'alimenter le capteur sur du 3.3 V et de le brancher à la masse. Dès lors que les branchements sont effectués, le programme pourra être compilé puis télécharger sur la carte Freescale.

Le programme de ce capteur utilise des composants de Processor Expert. Afin de transférer les valeurs acquises par le capteur via un module XBEE, il va falloir utiliser des composants KSDK étant donné que le protocole XBEE disponible a été programmé avec des composants KSDK. Le composant analogique KSDK à utiliser est le même que celui utilisé pour le capteur d'humidité du sol soit le ***fsl-adc-16***.

PLAQUE CHAUFFANTE

La plaque chauffante est commandée via un relais. Elle doit être connectée à la carte comme présentée dans le schéma ci-dessous.



La LED représentée dans le schéma est une illustration de la plaque chauffante. Le montage ci-dessous est connecté sur la pin PTE29 de la carte FRDM-KL26Z sur laquelle est connectée le montage. Le transmetteur et le récepteur du module XBEE chargé de recevoir la commande d'allumer ou d'éteindre la plaque chauffante sont respectivement connectés aux pins PTE0 et PTE1 de la carte.

Il ne faut surtout pas oublier d'alimenter la plaque chauffante et de le brancher à la masse. Dès lors que les branchements sont effectués, le programme pourra être compilé puis télécharger sur la carte Freescale. Le principe de câblage utilisé pour la plaque chauffante reste le même que pour le ventilateur, le l'humidificateur et l'électrovanne, car ils sont commandés par des relais.

PANNEAU DE LED

Le panneau de LED doit être branché à une alimentation. Un montage composé d'une diode infrarouge et d'un module XBEE doit également être mis en place afin de contrôler le panneau de lumière via son récepteur infrarouge.

Pour ce qui est du montage Diode infrarouge et XBEE, vous devez utiliser une carte Arduino Mega 2560. La borne positive de la Diode émettrice doit être connectée à une résistante de 330 Ohm puis, vers la pin 9 de la carte Arduino. Veillez à connecter la borne négative à la terre. Les pins de données réceptrices et transmetteurs du module XBEE sont connectées respectivement aux pins 10 et 11 de la carte Arduino Mega 2560.

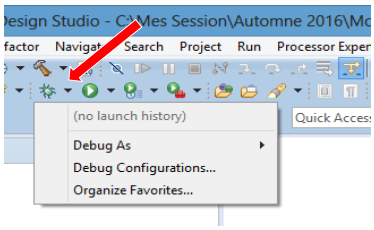
Dès lors que le montage est réalisé, veillez à le placer près du panneau de lumière de telle sorte que la diode émettrice soit face aux diodes réceptrices du panneau. Le programme Arduino du panneau de lumière peut alors être compilé et envoyé sur la carte Arduino.

EXÉCUTER LES DIFFÉRENTS PROJETS

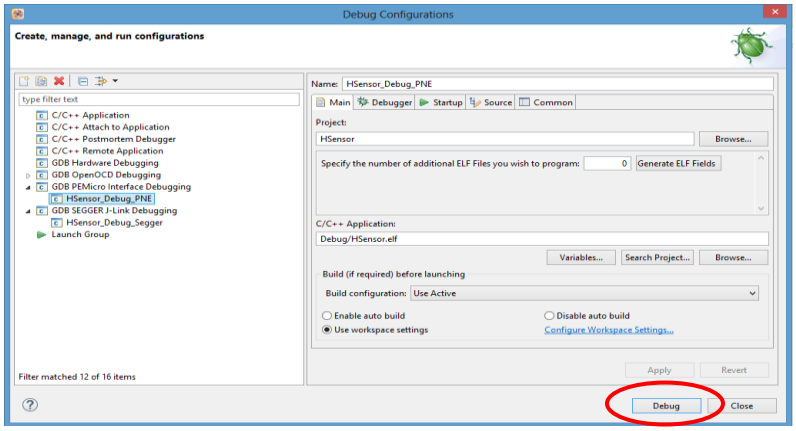
Avant de lancer la simulation du projet, il est primordial de le compiler et le déboguer afin de vérifier qu'il est sans erreur. Pour ce faire, il faut sélectionner le projet à simuler et cliquer sur le petit marteau suivant  dans la barre d'outils.

Ce marteau va compiler le projet et s'il est correct ou s'il y a un problème, vous pouvez visualiser les résultats de cette opération dans la fenêtre de la console.

Une fois la compilation terminée, il faut créer une configuration de débogage. Afin d'avoir la plupart des domaines de la configuration de débogage prépeuplée, sélectionnez le projet dans la vue Explorateur de projets. Puis cliquer sur le menu déroulant de l'icône "Debug" comme présenté ci-dessous.

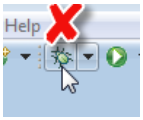


Ensuite, sélectionner **Debug Configuration** et la boite de dialogue ci-dessous apparait.



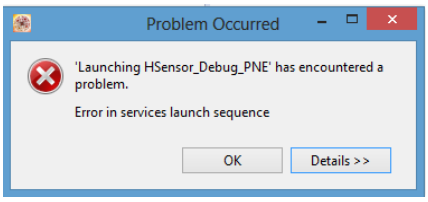
Puis, sélectionner le fichier à déboguer et utiliser le bouton "Debug" de la boîte de dialogue Configuration de débogage afin de lancer le débogage du projet.

Une simple pression sur l'icône 'Debug' n'est en général pas recommandée si elle ne montre pas dans l'infobulle une configuration de lancement:



Lorsque le débogage a fonctionné sans erreur, il faut cliquer sur l'icône *Resume* ou appuyez sur la touche « F8 » pour l'exécution du projet.

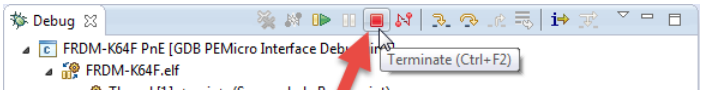
Si le débogage ne fonctionne pas comme le montre l'image ci-dessous, voici quelques conseils de dépannage:



Réglages : vérifiez que tous les paramètres de configuration sont OK et que vous déboguez sur une carte Freescale correcte avec une connexion correcte (PnE, OpenOCD, Segger).

Puissance : Vérifiez que la carte Freescale est sous tension et connecté au port USB approprié sur la carte.

Non Débogage deux fois : Assurez-vous que vous qu'une application ne soit pas déjà en cours d'exécution sur la carte Freescale et qu'une session de débogage ne soit pas en cours. Alors, il faut toujours mettre fin à une session de débogage avant de commencer une nouvelle



COMMUNICATION

TOPICS DU BROKER

Topic	Description
iserre/temperature/capteur	Valeurs de température mesurées par les capteurs.
iserre/temperature/actionneur	Commande de la plaque chauffante.
iserre/temperature/config	Paramètres de configuration (température).
iserre/humidite/capteur	Valeurs mesurées pour l'humidité de l'air
iserre/humidite/config	Paramètres de configuration (humidité air).
iserre/humidite/actionneur	Commande de l'humidificateur
iserre/led/etat	
iserre/led/intensite	
iserre/led/couleur	

SPÉCIFICATIONS DÉTAILLÉES (PAR TOPIC)

Les codes hexadécimaux utilisés dans les tableaux suivants sont sujets à changement, et ce document doit être mis à jour une fois l'implémentation terminée et chaque fois que ces valeurs seront modifié dans le code.

ISERRE/X/CAPTEUR

Les clients (sink) publient en clair la valeur mesurée par le capteur (température en degrés Celsius, humidité relative de l'air en pourcentage, etc.)

ISERRE/X/CONFIG

Le client (openHAB) publie la configuration désirée pour un type de capteur. La configuration est définie comme un ensemble de couples code/valeur où chaque paramètre de configuration a un code assigné.

Le format du message MQTT est : « *count code/valeur* ».

Le nombre de couples code/valeur doit correspondre à la valeur de count (chiffre indiquant le nombre de paramètres de configuration).

Ce principe est appliqué pour tous les topics de configuration de capteur.

Les paramètres actuellement définis pour les différents types de capteur sont :

Code (Température)	**Description**
0x01	Fréquence d'échantillonnage de la température.

Code (Humidité)	**Description**
0x01	Fréquence d'échantillonnage de l'humidité.

ISERRE/X/ACTIONNEUR

Le client (openHAB) publie 1 ou 0 pour activer ou désactiver l'actionneur.

ISERRE/LED/X

Le topic du panneau de lumière est un cas particulier : l'implémentation de ce panneau a été faite en utilisant une carte Arduino faisant office de télécommande infrarouge; le panneau est par conséquent contrôlé en envoyant des codes hexadécimaux, chacun correspondant à une commande particulière (couleur, contrôle d'intensité, etc.). Par conséquent, pour cet actionneur seulement, le contrôle se fait en publiant (à partir du client openHAB) directement les codes de commande.

CORRESPONDANCE CODES HEX – ACTION CORRESPONDANTE

Utiliser la télécommande et une carte Arduino avec un récepteur infrarouge pour retrouver tous les codes.

Action	**Code HEX**
Augmenter intensité lumière	FF906F
Diminuer intensité lumière	FFB847
Éteindre panneau (OFF)	FFF807
Allumer panneau (ON)	FFB04F
Couleur : Rouge	FF9867
Couleur : Vert	FFD827

Couleur : Bleu	FF8877
Couleur : Blanc	FFA857

MODULE XBEE – KINETIS

Le module implémentant la génération et le décodage de trames Xbee se trouve dans les fichiers Xbee_Driver.h et Xbee_Driver.c

Ces modules ont été implémentés pour fonctionner en mode API 2.

Les fonctionnalités suivantes sont implémentées :

ENVOI DE TRAME VERS UN AUTRE MODULE

La fonction *sendFrame*() prend en paramètre l'adresse de destination (uint64_t) , les données à envoyer (tableau d'octets), et le nombre d'octets du message (uint8_t).

DÉCODAGE DE TRAME REÇUE

Un buffer FIFO avec des fonctions de lecture et d'écriture a été implémenté (fichiers *fifo_buffer.c* et *fifo_buffer.h*) et associé à l'UART sur lequel le module Xbee est branché. La fonction callback de l'UART (se trouvant dans *Events.c*) qui est appelée chaque fois qu'un octet est reçu ne fait que remplir au fur et à mesure le buffer. Deux fonctions sont ensuite importantes pour décoder le message :

- **`lire_contenu_fifo`** (dans *`app.c`*) **:** prend en paramètre une structure fifo et un tableau d'octets, et sert à transférer le contenu du buffer dans le tableau passé en paramètre.
- **`createRxFrame`** (dans *`Xbee_Driver.c`*) **:** crée une structure xbee_rx_frame qui représente une trame de réception à partir d'un tableau brut d'octets. Les différents champs de cette structure sont ensuite facilement accessibles (adresse émettrice, checksum, payload, etc.)

COMPATIBLITÉ KINETIS SDK

Ces modules ont également été implémentés pour ne fonctionner qu'avec le Kinetis SDK v1.3. Cette dépendance n'est importante que pour l'utilisation du composant UART défini dans Processor Expert. Afin de rendre ce module compatible avec des composants autres que ceux du KSDK 1.3 il faut adapter :

- L'écriture vers l'UART dans la fonction *sendFrame* (*Xbee_Driver.c)* : lignes 258-260.
- L'implémentation du callback de l'UART pour la réception.

ARCHITECTURE DU PROGRAMME SINK

L'ensemble du programme exécuté par les différents sinks se trouve dans le fichier source « kinetisClientSample.cpp ».

Il comporte plusieurs sections contenant les différentes parties du programme que l'utilisateur peut personnaliser. Ce fichier utilise la compilation conditionnelle faite grâce aux directives du préprocesseur (directive #ifdef par exemple). Cela permet de personnaliser le comportement du programme sans avoir à réécrire de code. Il suffit juste de définir quelques « mots-clés ».

Ce fichier utilise la pile mqttsn, l'uart 1 de la carte freescale (Uart_Com1), la librairie du protocole iSerre-SN dont l'implémentation n'est pas tout à fait achevée, ainsi qu'une classe de gestion des mesures appelée SInkMeasureManager, qui s'occupe de faire la moyenne des mesures reçues. Ce gestionnaire de mesure peut également être personnalisé en cas de besoin (si un traitement autre qu'une moyenne est nécessaire par exemple).

Certaines des sections décrites ci-dessous seront amenées à être modifiées. Il est donc important de tenir cette partie à jour.

PARAMÈTRES GÉNÉRAUX DE L'APPLICATION 1 (LIGNE 19 À 24)

- Permets de choisir le type de sink à implémenter en décommentant le #define correspondant
- Permets d'ajuster le nombre de mesures nécessaires pour faire la moyenne, ainsi que la période à laquelle ces mesures sont faites

PARAMÈTRES GÉNÉRAUX DE L'APPLICATION 2 (LIGNE 34 À 54)

- Permets de configurer le baudrate de l'uart utilisé pour le module Xbee
- Permets de définir le nom du sink (sous forme de chaine de caractères)

ADRESSE DES MODULES EN LIAISON AVEC LE SINK (LIGNE 61 À 62)

- Liste toutes les adresses des capteurs et actionneurs avec lesquels le sink doit communiquer.
- Cette section devrait disparaitre quand la découverte du réseau sera implémentée (procédure SearchSink)

LISTE DES TOPICS (LIGNE 68 À 78)

- Liste tous les topics utilisés par les sinks
- Cette section pourrait également profiter de la compilation conditionnelle afin d'optimiser le code produit

TÂCHES ET CALLBACKS (LIGNE 90 À 246)

- Définis l'ensemble des tâches à exécuter périodiquement ainsi que l'ensemble des callbacks appelés lors de la réception d'un message
- Enregistre ces fonctions auprès de l'application MQTT-SN

INTERFACE UTILISATEUR

DÉMARRAGE DU SERVEUR OPENHAB

Le serveur openHAB se retrouve installé sur une machine Xubuntu dans le laboratoire de recherche en sécurité informatique.

User	Password
iSerre	gen12731

Démarrage du serveur :

sudo /etc/openhab/start.sh

Pour vérifier la connexion au serveur, nous pouvons accéder à l'interface dans un navigateur à l'adresse suivante:

La lecture des messages disponibles dans la ligne de commande est recommandée.

http://132.213.8.X[1]:8080/openhab.app?sitemap=projet#_Home

DÉMARRAGE DE MYSQL

Afin de pouvoir assurer le service de persistance sur openHAB, le serveur mySQL doit être en marche.

Démarrage du serveur mySQL :

sudo ./mysql start

Nous nous intéressons à la base de données **Openhab**. Afin de pouvoir explorer les données :

sudo mysql –u root -p

User	Password
openhab	iserre

Une invite à commandes SQL sera ouverte :

USE openhab,

SHOW TABLES,

SELECT * FROM NOM_TABLE,

[1] Vérifier toujours l'adresse IP de la machine Xubuntu.

BROKER MYMOSQUITTO

Pour pouvoir tester l'envoi et réception de message par le protocole de MQTT.

Démarrer le service :

sudo service mosquittto start

Vérifier le service :

sudo service mosquitto status

Envoie de messages :

mosquitto_pub -h 132.213.8.2 –d –t topic –m 'message'

Réception de messages :

mosquitto_sub -h 132.213.8.2 –t topic –d

AJOUTER DES FONCTIONNALITÉS

Pour ajouter des fonctionnalités, il faut d'abord créer un nouvel item dans le fichier « *projet.items* ». Plusieurs types d'item[2]s peuvent être utilisés. Il faut donc choisir le bon type d'item en fonction de la fonctionnalité dont nous voulons ajouter. Pour créer un item, il suffit de suivre la nomenclature ci-dessous :

Type_item "Nom_Option [type_unité]" <icone> (groupe) {topic (s'il y a lieu)}

Plusieurs exemples sont présents dans le fichier « *projet.items* ». Il est donc possible de se baser sur un item existant afin d'en créer un nouveau. Toutes les icônes utilisées sont dans le répertoire « */…/etc/openhabiSerre/webapps/images/* ». Il est également possible de télécharger des images sur Internet (format PNG) pour ensuite les insérer dans ce répertoire pour ainsi être en mesure d'utiliser des icônes personnalisées. Suite à la création d'un nouvel item, il faut l'ajouter dans le fichier « *projet.sitemap* ». Il y a une multitude de variations quant à l'ajout d'un item dans ce fichier, c'est-à-dire

[2] La liste des types d'items pouvant être utilisés est décrite dans ce lien : https://github.com/openhab/openhab/wiki/Explanation-of-items

que plusieurs types d'éléments[3] peuvent être utilisés. Il ne suffit donc juste que de se baser sur les éléments déjà insérés afin d'en ajouter de nouveaux. L'architecture du fichier « *projet.sitemap* » est assez simple à comprendre. Suite à la modification de ces deux fichiers, il faut implémenter des règles sur le nouvel item créé (si nécessaire). Pour ce faire, il ne suffit que de suivre l'architecture suivante :

```
Var type_var_var = X        /*créer une variable si nécessaire*/
Rule "Nom_Item"

        when
                Item Nom_Item <Action>
        then
                …
end
```

Suite à la création des règles, si la nouvelle fonctionnalité en utilise, l'ajout de cette nouvelle fonctionnalité est terminé. Pour résumer, il ne suffit que de modifier convenablement les fichiers « *projet.items* », « *projet.sitemap* » et « *projet.rules* » (si nécessaire) afin de créer une nouvelle fonctionnalité.

[3] La liste des types d'éléments pouvant être utilisés est décrite dans ce lien : https://github.com/openhab/openhab/wiki/Explanation-of-Sitemaps