|  |
| --- |
| Université du québec en outaouais |
| INF4173 - Projet Synthèse |
| Rapport de progrès |
| Présenté à M. Michal Iglewski  Par |
| **Vincent Crispin Emond & Kossi Ahlin Cornelus Madjri** |
| **3/1/2017** |

Table des matières

[Introduction 3](#_Toc476145223)

[Apprentissage, Analyse et préparation de l’équipement 3](#_Toc476145224)

[Rencontres avec le superviseur 3](#_Toc476145225)

[Analyse de la documentation existantes 3](#_Toc476145226)

[Mise en place de l’environnement de travail 4](#_Toc476145227)

[Analyse et documentation du code 4](#_Toc476145228)

[Pratique de la programmation sur microcontrôleur 4](#_Toc476145229)

[Test de l’architecture existante 4](#_Toc476145230)

[Implémentation d’ISN 4](#_Toc476145231)

[Définition et construction des trames 4](#_Toc476145232)

[Reconnaissance des trames iSN 5](#_Toc476145233)

[Design et implémentation d’une classe pour filtrer les trames sur le réseau 5](#_Toc476145234)

[Ajouter iSN à la boucle principale 5](#_Toc476145235)

[Évaluation des objectifs 5](#_Toc476145236)

[Revue de la méthodologie 6](#_Toc476145237)

[Conclusion 6](#_Toc476145238)

[Glossaire 7](#_Toc476145239)

[Annexe 8](#_Toc476145240)

# Introduction

Ce document rend compte de l’avancement de notre projet synthèse. Nous rappelons que notre projet vise à mettre au point une serre électronique automatisée dont les paramètres environnementaux sont configurables par exemple : la température, l’humidité etc. Plus précisément, nous travaillons à améliorer l’architecture réseau de la serre, et ce, non seulement en améliorant et documentant le code existant, mais aussi en faisant le design et l’implémentation de nouveau code. En effet, une de nos plus grandes tâches vise à implémenter iSN (iSerre Sensor Network) qui est un protocole de communication utilisé entre les capteurs et des nœuds appelés « sink ».

## Apprentissage, Analyse et préparation de l’équipement

La majorité de notre temps jusqu’à maintenant a été passé à apprendre et à analyser l’architecture existante. Le codage consiste en de la programmation embarquée sur microcontrôleur ce qui n’est pas enseigné dans le cadre de notre BAC, nous avons donc eu à apprendre énormément de choses en peu de temps.

## Rencontres avec le superviseur

Nous avons eu des rencontres hebdomadaires avec notre superviseur M. Kamel Adi qui nous à petit à petit expliquer toutes les composantes du projet. Nous avons entre autres appris :

* L’architecture générale du projet ainsi que les objectifs de celui-ci.
* Comment programmer sur une carte microcontrôleur Freescale FRDMKL26Z.
* Comment configurer et utiliser les modules radio émetteur/récepteur XBEE.
* Comment fonctionne le protocole MQTT-SN.
* Comment brancher le matériel.

## Analyse de la documentation existantes

Avant de commencer nous avions besoin de plus de connaissance afin de comprendre le code déjà en place. Nous avons donc eu à lire plusieurs documents pour obtenir ces connaissances. Parmi ces documents nous avons entre autres lu :

* Le document de spécification du protocole MQTT-SN.
* Les documents sur l’installation et la fiche technique de la carte microcontrôleur.
* Le document de spécification du protocole MQTT.
* Le rapport et la documentation technique des étudiants ayant précédemment travaillés sur ce projet.

## Mise en place de l’environnement de travail

À l’aide de notre superviseur, nous avons préparé un local ou nous avons déplacer l’équipement nécessaire pour le travail. Nous avons :

* Installer le système d’exploitation sur la machine de travail.
* Installer l’environnement de développement KDS (Kinetis Design Studio)
* Installer les différents outils logiciels nécessaires (Git, XCTU, Coolterm etc.)
* Fais les branchements des capteurs, « sink » et passerelle.

Notre superviseur nous a ensuite montré comment utiliser raspbian sur la raspberry pi pour activer la passerelle. La passerelle est un nœud qui permet aux « sink » de se connecter à l’internet.

## Analyse et documentation du code

Après avoir obtenu ces connaissances nous avons été en mesure de commencer à analyser le code et en insérer des commentaires petit à petit. Le code a été en grande partie adapté d’un code open source existant, il est donc très peu commenté.

## Pratique de la programmation sur microcontrôleur

Nous avons pratiqué la programmation sur la carte freescale pour nous familiariser avec la programmation embarquée. Nous avons par exemple écrit du code pour faire varier les couleurs d’une DEL.

## Test de l’architecture existante

Une fois notre local disposé nous avons effectué des tests pour s’assurer que les « sink » arrivaient à communiquer avec la passerelle. Notre superviseur en a profité pour nous expliquer les différentes causes possibles de problèmes lorsque la communication ne fonctionne pas.

# Implémentation d’ISN

Nous avons utilisé l’ébauche du protocole iSN décrite dans la documentation laisser par les étudiants précédents pour commencer à faire une première implémentation d’ISN. Notre approche pour développer ce protocole sera incrémentale, la première étape consistera à pouvoir ajouté des capteurs automatiquement dans le réseau. Pour ce faire nous avons déjà écrit une bonne partie du code.

## Définition et construction des trames

Nous avons décidé de créer une classe C++ pour chaque message (trame) iSN pouvant être envoyée ou reçue. Elles héritent toute d’une classe IsnMessage qui définit un tampon qui contiendra les octets de la trame. Nous avons déjà ajouté et testé du code pour construire le tampon de certains de ces messages, notamment ceux servants à la connexion qui nous seront utile pour le premier incrément.

## Reconnaissance des trames iSN

Un des défis que nous avons eu était de trouver un moyen de faire la différence entre une trame MQTT-SN et une trame iSN. Si la trame est dans un format valide pour les deux protocoles, alors la trame est ambiguë et on ne peut pas faire la différence. On s’est donc dit que si on bâtit une trame MQTT invalide alors on pourrait détecter une trame iSN. Notre méthode est simple, nous ajoutons deux octets 0x0203 que nous appelons « Magic Number » au début de chaque trame iSN. Dans le protocole MQTT-SN le premier octet spécifie la longueur de la trame dans ce cas la longueur est de deux octet. Le deuxième octet est le type de message MQTT dans ce cas c’est 0x03, or cette valeur est réservée dans le protocole MQTT on élimine donc toute ambiguïté car aucune trame MQTT valide ne peut commencer par ces deux octets. Du même coup on peut détecter une trame iSN facilement en vérifiant les deux premiers octets de l’en-tête.

## Design et implémentation d’une classe pour filtrer les trames sur le réseau

Avant d’ajouter iSN, le réseau recevait seulement des trames de type MQTT-SN il les envoyait donc systématiquement au protocole MQTT-SN pour le traitement. Maintenant que nous ajoutons iSN nous avions besoin d’un mécanisme pour filtrer les trames MQTT-SN des trames iSN. Lors de la réception de données, le réseau appel une fonction appelé « callback » et lui passe les données reçues. Avant, ce « callback » était une fonction dans le client MQTT-SN, maintenant, nous avons codé une nouvelle fonction « callback » à l’extérieur du client MQTT-SN, celle-ci regarde les données, si elle détermine que c’est une trame MQTT-SN elle envoie ces données à « l’ancien » « callback » MQTT-SN sinon elle l’envoi à une fonction « callback » dans le client/serveur iSN.

## Ajouter iSN à la boucle principale

Nous avons aussi ajouté iSN dans la boucle principale, sa méthode « exec » est appelé à chaque itération il peut donc effectuer du travail régulièrement.

# Évaluation des objectifs

Dans cette section il sera question d’évaluer si nos objectifs sont toujours pertinents et réalisables. Nous rappelons que nos objectifs sont :

* Améliorer l’efficacité et la stabilité de l’architecture de communication de la serre.
* Implémenter (coder) le protocole iSerre Sensor Network (iSN) sur les cartes microcontrôleurs.
* Pouvoir ajouter ou retirer de nouveaux capteurs dans la serre sans avoir à les configurer manuellement.
* Obtenir de l’expérience de programmation embarquée avec C++ et de programmation de protocoles de communication.

Nous considérons que nos objectifs sont encore bons et réalisables. Par exemple, en implémentant iSN, nous améliorons l’efficacité et la stabilité de l’architecture de la serre en corrigeant du code lorsque nous trouvons des problèmes. Pour ce qui est d’iSN sont implémentation a déjà été entamé. Notre premier incrément vise à atteindre l’objectif trois et l’objectif quatre est déjà atteint puisque nous avons beaucoup appris sur C++ et la programmation embarquée.

# Revue de la méthodologie

Dans notre plan de travail nous avons écrit plusieurs points pour expliquer comment nous allions atteindre nos objectifs. En avançant dans le projet deux de ces points sont sujets à des changements :

* Nous allons vérifier si une implémentation du protocole de communication ZigBee qui est un standard de l’industrie pourrait être envisageable à la place du protocole propriétaire DigiMesh. Cela pourrait améliorer l’interopérabilité des composantes utilisées dans la serre.

Nous allons abandonner ce point car son impact sur le projet n’est pas très grand et il demanderait une réécriture complète des classes réseaux utilisées, cela n’est pas réalisable avec le temps qu’il nous reste et surtout nous éloignerais de d’autres points plus importants du projet.

* Nous allons tenter de voir s’il est possible d’utiliser un seul module Xbee par sink au lieu de deux.

Nous n’allons pas abandonner ce point mais nous allons le garder comme élément secondaire non prioritaire. La priorité sera tout d’abord d’implémenté iSN une fois qu’iSN est stable nous allons pouvoir regarder ce point si le temps le permet.

# Conclusion

En conclusion, nous avons assez bien suivi notre calendrier, nous avons commencé à coder une semaine plus tard que prévu mais nous considérons que le retard est léger et rattrapable. Nous considérons que le projet avance bien et que nos objectifs sont réalisables. Parmi les choses que nous devons faire bientôt on peut noter qu’il faut parler à notre superviseur pour obtenir plus d’information sur le fonctionnement du protocole Digimesh et obtenir plus de renseignements sur les actions qui doivent être prise par le protocole iSN.

# Glossaire

**Carte Freescale :** La carte microcontrôleur (FRDMKL26Z) utilisé pour implémenter les capteurs et les « sink ».

**DigiMesh :** Protocole réseau propriétaire développé par la compagnie Digi. Il est utilisé dans la serre pour la communication entre les xbee.

**iSN :** iSerre Sensor Network, protocole application utilisé pour communiquer entre les capteurs et les « sink ».

**MQTT-SN :** MQTT pour réseau de capteurs. Protocole application utilisant un modèle « Publish-Subscribe ». Dans la serre un client peut par exemple s’inscrire au « topic » température pour obtenir la température dans la serre.

**Sink :** Microcontrôleur équipé d’un module radio xbee. Communique avec les capteurs pour recueillir les mesures.

**Xbee** : Appareil transmetteur récepteur radio.

**ZigBee :** Protocole réseau basé sur la specification IEEE 802.15.4 pour les petits émetteur-récepteur radio.

# Annexe

Lien github vers le code, regarder les « commits » pour voir le travail éffectué :

<https://github.com/VincentEmond/iSerre-1>