École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
3e année
2012 - 2013

Rapport de projet de Système d'exploitation

Relevé automatique de présence sous système RFID UHF

Encadrants
Mathieu Delalandre
mathieu.delalandre@univ-tours.fr

Université François-Rabelais, Tours

Etudiants
Lin SHEN
lin.shen@etu.univ-tours.fr
Peng BAI
peng.bai@etu.univ-tours.fr

DI3 2012 - 2013

Version of June 12, 2013

Contents

List of Figures

Introduction

Dans le cadre de notre formation en informatique, nous avons eu l'occasion de réaliser un projet de système exploitation sur le thème de la radio identification.La radio-identification est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés radio-étiquettes i.e. RFID tag.Les radio-étiquettes sont des objets de faibles dimensions, tels que des étiquettes autoadhésives ou des puces, qui peuvent être collés ou incorporés dans des objets, produits ou organismes vivants. Les radio-étiquettes sont utilisés dans de nombreuses industries. Cette technologie d'identification peut être utilisée pour identifier les objets,comme un code-barres.Une radio-étiquette qui est attachée à un automobile pendant la production peut être utilisée pour suivre sa progression à travers la ligne de montage.Il peut aussi être attaché à l'habillement, les possessions, ou même implanté dans les gens.Donc il y a des possibilités de lire des informations personnellement.

Dans ce projet on s'intéressera à la problématique de relevé de présence au sein d'un institut d'enseignement. Un module CAEN RFID nous a été fourni afin de développer une application permettant de détecter la présence d'un ensemble d'étiquettes hautes fréquences.Nous utilisons C# pour réaliser un logiciel de ce sujet.C'est divisé en deux parties.D'abord lecteur détecte les radio-étiquettes.Puis nous cherchons des informations des radio-étiquettes dans le base de donnée.Si ça existe,les informations ont été révélé dans une list.Sinon,nous les ajoutons au base de donnée.

Pour réaliser la fonction de ce sujet.Avant tout,il faut comprendre l'utilisation du module CAEN RFID. "Qu'est-ce que le principe de chaque composant?""Comment connecter l'équipement à un ordinateur?"Nous étudions le manuel technique de CAEN ION R4300P.C'est un UHF(Ultra haute fréquence) lecteur avec la portée longue et GPRS.Puis nous traitons les données avec l'ordinateur.Pour plus de détails, nous verrons dans une première partie le cahier des charges qui nous a été fourni par notre professeur.

Cahier des Charges

L'application « présence/absence » qui devra permettre à un utilisateur de lancer une application sur un pc pour avoir la visualisation temps réel de la présence ou non des tags à proximité des antennes. Vous trouverez ci-dessous un schéma de principe du projet :

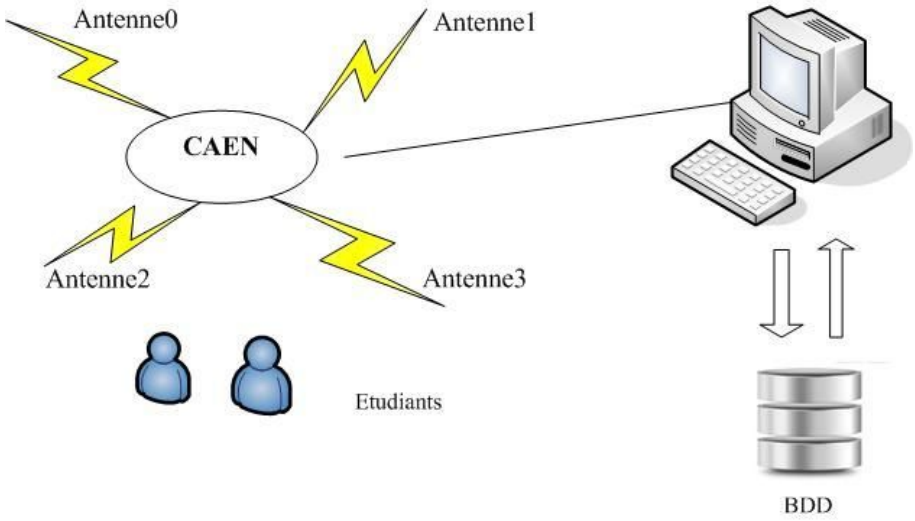

Figure 2.1: figure de la principe

Le main appel trois fonctions :

- Les radio-étiquettes interagissent avec les antennes qui relaient l'information au CAEN.
- Le CAEN gère la réception des informations et envoie les données collectées au serveur. On utilisera le terme de client pour parler du module CAEN ainsi que de ses antennes.
- Le serveur,qui se situera sur un ordinateur,communiquera avec un module CAEN. Il intègrera l'application utilisateur.

Explication de la solution adoptée

3.1 Description matérielle

3.1.1 Schéma de principe

Le schéma suivant décrit notre installation :

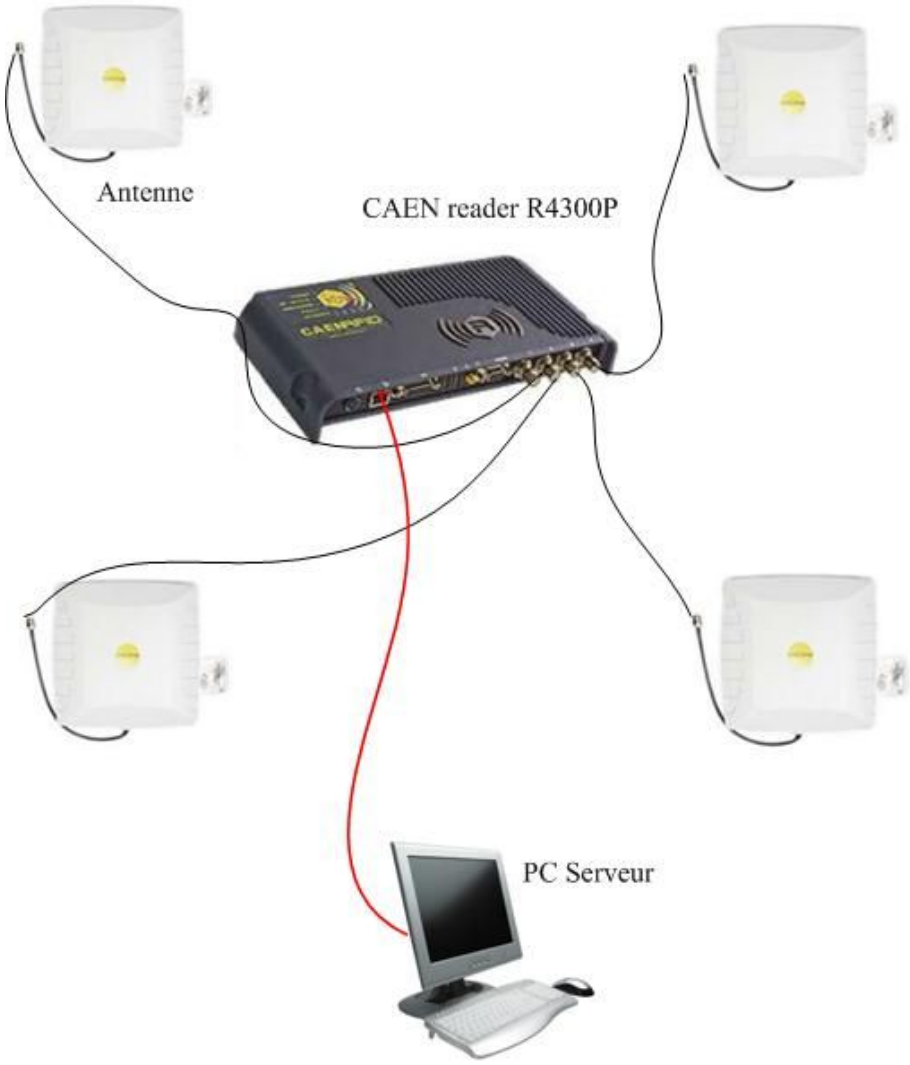

Figure 3.1: schéma de l'instalation

3.1.2 Présentation du matériel

Dans le cadre du projet nous disposons du matériel suivant :

- 4 antennes hautes fréquences WANTENNAX005
- Un module CAEN RFID R4300P
- Un lot de tag RFID
- Un serveur pour l'application client

Les quatre antennes sont branchées directement sur le CAEN RFID R4300P via des câbles coaxiaux de longueur 6 mètres.

Le CAEN RFID R4300P dispose d'un port Ethernet. Nous souhaitons relier le module sur le réseau afin qu'il soit accessible à distance à partir d'un PC sur lequel l'application aura préalablement été installée.

Nous disposons également d'un PC qui servira de serveur où se situera l'application de visualisation et qui sera paramétré de façon à communiquer avec le module CAEN RFID 4300P. L'installation nécessite aussi un switch qui permettra de connecter le client, le serveur et le CAEN RFID R4300P.

Enfin, un lot de tag RFID est mis à notre disposition afin de mettre en œuvre notre projet, il est important de disposer de plusieurs tags afin de pouvoir traiter les différents problèmes de collisions.

3.2 Configuration logiciel et environnement

Dans le cadre de notre projet, nous aurons à utiliser certains logiciels et systèmes d'exploitation, en voici une liste.

3.2.1 Logiciel

Dans cette partie vous trouverez une liste des logiciels que nous allons utiliser afin de mener à bien notre projet.

- L'environnement de développement intégré est Microsoft Visual Studio 2010. On a développé le programme avec la langue C#, par le projet WindowsFormsApplication.
- Tous nos documents seront rédigés sur le même modèle et avec la même charte graphique en utilisant un thème Microsoft Office.

3.2.2 Solution BDD

Access: Il récupérera les données du fichier *.accdb* et les modifiera afin d'avoir une correspondance entre l'utilisateur et le tag détecté. Il ajoutera donc les informations correspondant au prénom, nom, numéro de carte étudiante, formation en fonction de l'ID du tag.

3.3 Modèle de données

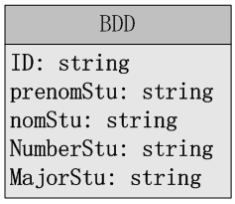

Figure 3.2: Les attributs de base de donnée

Dans la base de donnée, ID est le clé primaire, les autres attributs sont prenomStu, nomsStu, NumberStu, MajorStu. Tous ça sont de type "string". Mais dans le programme, quand il affiche le tag dans le Listview, il existe aussi le data, heure et numéro des antennes.

3.4 Application utilisateur

L'application permet de détecter si un badge est dans la zone de détection. Il peut détecter avec 4 antennes au même temps. L'utilisateur de l'application située sur un PC, pourra visualiser les badges en temps réel.

"Présence/Absence" qui permettra de savoir rapidement si un tag est présent près d'une antenne. Cette partie intégrera donc un plan de l'école où seront positionnées les antennes. Si un tag est présent l'application devra le signaler via un pop-up ou un changement de couleur de l'antenne concernée. Egalement un objet, type bandeau par exemple, devra permettre d'afficher le numéro du tag en question.

"Enregistrement" qui permettra de faire enregistrement de chaque tag dans la base de donnée locale. On a deux façons pour ajouter un nouveau tag au la table "infos" da la BDD. D'abord, on peut faire enregistrer sans faire la connexion avec le CAEN. Dans ce cas, il faut savoir le ID de tag. L'autre façon, quand un tag est détecté et il n'a pas été enregistré dans la BDD, on peut faire enregistrement directement. L'ID de tag est transmis automatiquement au la fenêtre de enregistrement. C'est facile pour enregistrer les informations des étudiants.

3.5 Protocole TCP/IP

Comme dans toutes communications, le CAEN et le PC devront s'échanger de nombreuses informations. Ces informations seront échangées au travers de sockets. Nous utiliserons un protocole en mode connecté: TCP/IP. Bien que celui soit plus lent que le protocole UDP, il est aussi plus fiable.
Echange informations entre les CAEN et PC:

1. Faire la connexion
2. Envoi d'info détection

Synthèse technique

Ce chapitre a pour objectif de présenter la programmation de ce sujet.

4.1 Partie du Front-end(CAEN)

4.1.1 Fonctions de la bibliothèque du CAEN utilisées

Pour la programmation côté client (CAEN ION R4300P), le constructeur du produit met à disposition un SDK 4.4.0(Software eveloppement Kit) disponible en téléchargement sur leur site internet (http://www.caenrfid.it).

Le SDK est disponible pour les langages Java, C, C#. Pour des raisons de commodité, nous avons choisi d'utiliser SDK(CAENRFIDLibrary.dll) la version C# pour Windows XP. Il contient la plupart des méthodes qui peuvent servir dans le cadre de l'utilisation du CAEN. Dans ce SDK, nous avons eu recours aux méthodes et aux classes suivantes :

```
1 CAENRFIDReader myReader = new CAENRFIDReader();
```

Cette ligne permet de créer une instance de la classe CAENRFIDReader. À partir de ce myReader, de nombreuses méthodes sont accessibles.

```
1 myReader.Connect(CAENRFIDPort.CAENRFID_TCP, this.IPtextBox1.Text);
```

Comme par exemple celle-ci, qui est indispensable avant toute opération avec le CAEN (depuis un pc). Cette ligne n'est plus nécessaire à partir du moment où l'on exécute le programme depuis le windows intégré au CAEN. On spécifie à la fenêtre du logiciel également l'adresse IP pour connecter, ce qui permet une certaine modularité concernant l'installation client.

```
1 CAENRFIDReaderInfo info = MyReader.GetReaderInfo();
2 MyReader.GetFirmwareRelease();
```

Ces deux lignes permettent de tester si on réussit de connecter le CAEN ou pas.

```
1 CAENRFIDLogicalSource MySource = MyReader.GetSource("Source_0");
2 CAENRFIDLogicalSource MySource1 = MyReader.GetSource("Source_1");
3 CAENRFIDLogicalSource MySource2 = MyReader.GetSource("Source_2");
4 CAENRFIDLogicalSource MySource3 = MyReader.GetSource("Source_3");
```

Cette méthode ci-dessus permet de créer une instance de LogicalSource et ajoute 4 antennes à l'instance. Utilisée comme ceci, permet de définir MySource comme étant les Logical Sources détectées à ce moment. On peut même utiliser un ou deux parmi ces MySources.

```
CAENRFIDTag[][] MyTags = new CAENRFIDTag[4][];
MyTags[0] = MySource.InventoryTag(new byte[] {},0,0, (short)1);
MyTags[1] = MySource1.InventoryTag(new byte[] {},0,0, (short)1);
MyTags[2] = MySource2.InventoryTag(new byte[] {},0,0, (short)1);
MyTags[3] = MySource3.InventoryTag(new byte[] {},0,0, (short)1);
```

Cette méthode crée une instance de Tag qui donnera les informations telles que le nombre de tags présents (détectés) sur 4 antennes.

```
public CAENRFIDTag[] InventoryTag(byte[] Mask,short MaskLength,
    short Position,short Flag)
```

Un appel à cette méthode va exécuter un cycle de lecture sur chaque point de lecture liée à la source logique.

```
MyTags[i].Length;
```

En les combinant, on obtient ici le nombre de tags détectés avec le résultat dans MyTags.

```
EPC = BitConverter.ToString(MyTags[i][j].GetId());
```

On obtient le ID des tags avec la ligne ci-dessus.

4.1.2 Architecture du programme, principales étapes

Le ordinogramme du programme est ci-dessous:

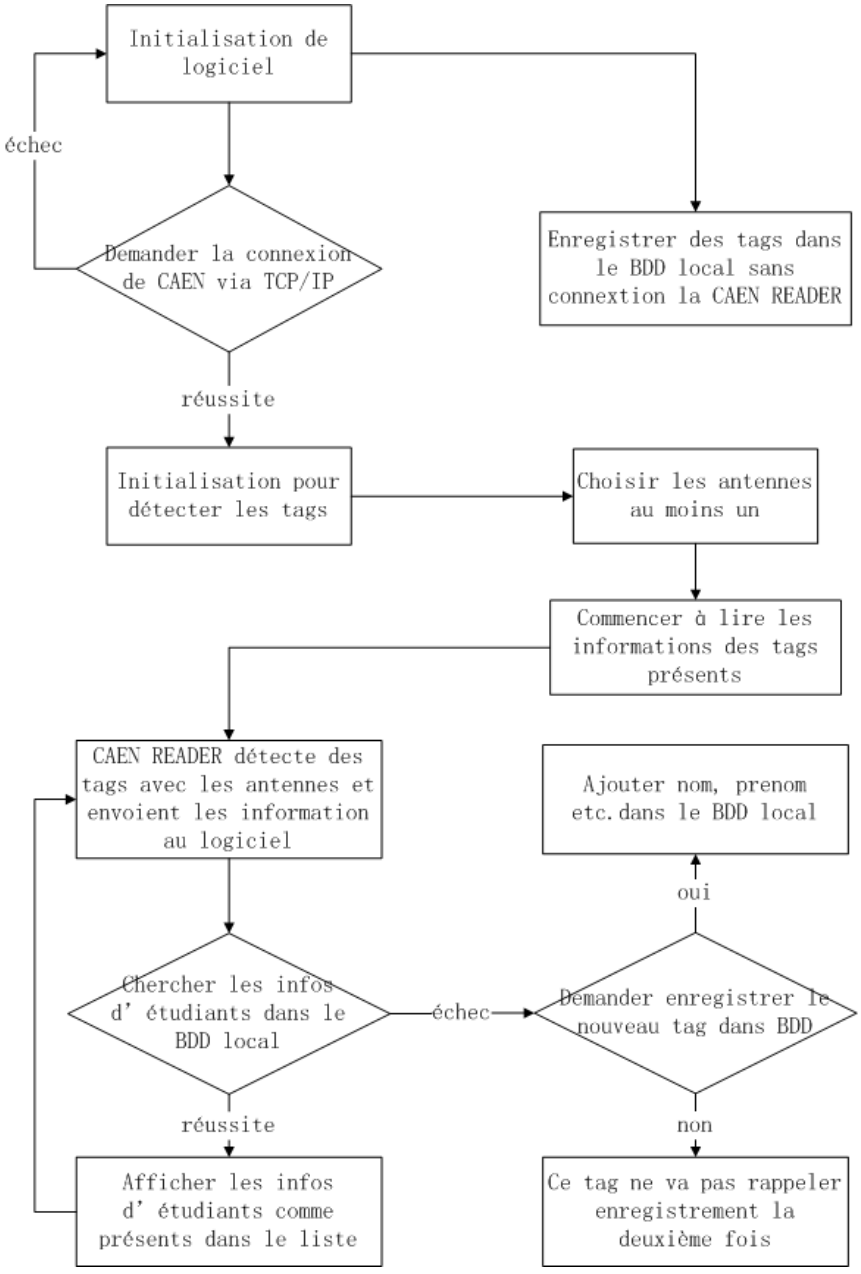

Figure 4.1: Le ordinogramme du programme

Le programme se découpe en deux classes fenêtres principales: MainForm, registerForm

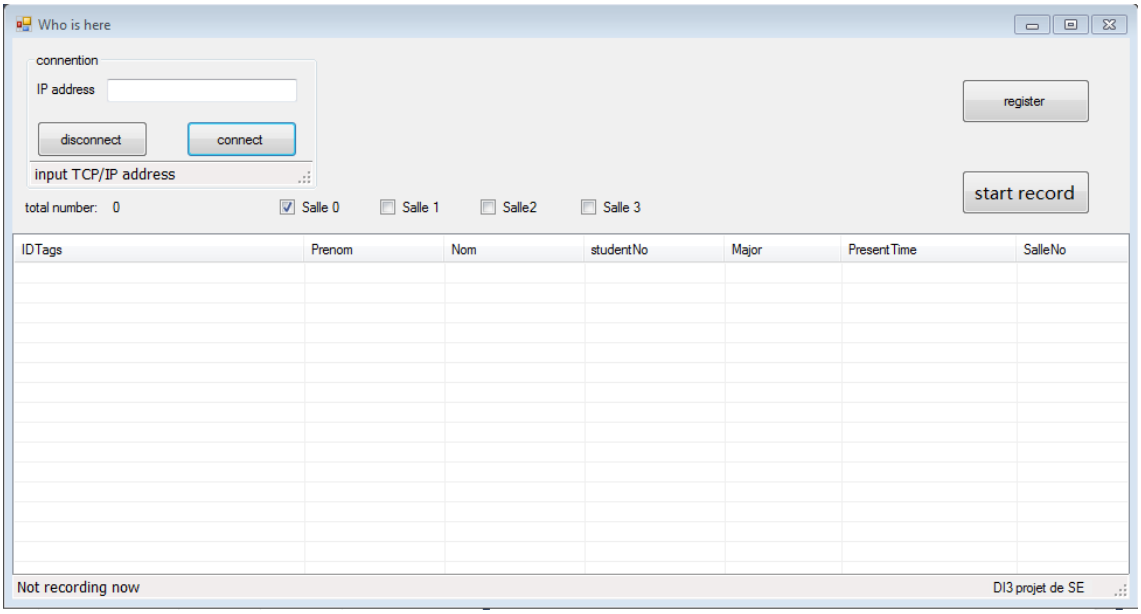

Figure 4.2: MainForm

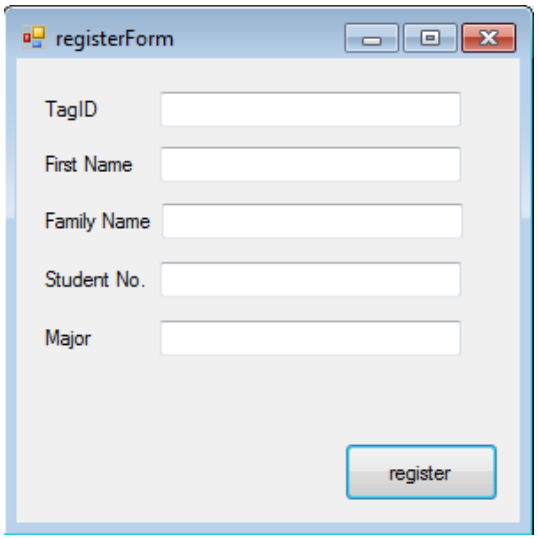

Figure 4.3: registerForm

MainForm

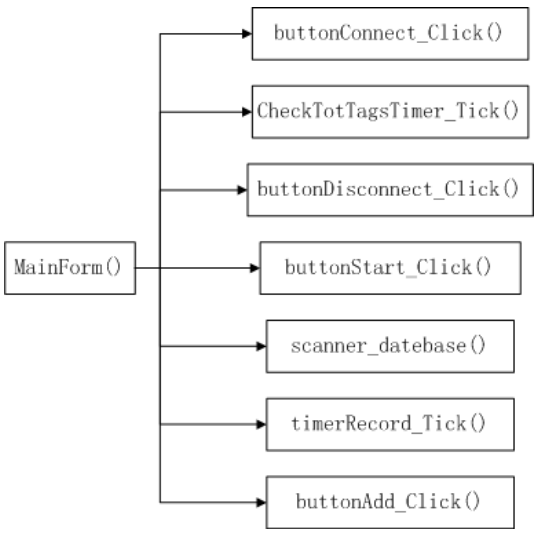

Figure 4.4: structure de la classe MainForm

Dans la classe MainForm, on retouve 7 fonctions principales qu'il faut.

- **buttonConnect_Click()**: Dans cette fonction, on crée la connexion avec le CAEN reader à une adresse IP, si la connexion est réussite, on peut commencer enregistrer les présents. Sinon, ce indique échec de la connexion.
- **CheckTotTagsTimer_Tick()**: Cette fonction est pour compter le nombre de tags présents. C'est le nombre de lignes dans le "Listview".
- **buttonDisconnect_Click()**: Faire la disconnexion de le CAEN reader.
- **buttonStart_Click()**: Après on réussit la connexion avec le CAEN reader, on peut commencer détecter des tags par les antennes. Ici on utilise un outil de C#, "Timer", chaque 100ms, on rappelle la fonction ***timerRecord_Tick()*** et demande les données de CAEN. Et on peut aussi arrêter enregistrer avec cette fonction.
- **scanner_datebase()**: Si un tag est détecté, on cherche les information dans la BDD locale, et on ajoute dans le "Listview" de application. Si on n'y les trouve pas, on rappelle la classe ***FormAlert()*** pour demander untilisateur qu'il veut enregistrer ce tag dans la BDD ou pas. S'il veut, on rappelle la classe registerForm. Pour l'instant le ID de tag est transmis automatiquement dans le fenêtre enregistrer.
- **timerRecord_Tick()**: Cette fonction est utilisée par des threads des antennes, à chaque lancement on a d'abord une partie initialisation de la communication avec les antennes. Une fois la première partie terminée, on rentre dans une boucle qui vient scruter une antenne, vérifier la présence de tags et si un tag est présent, on rappelle la fonction ***scanner_datebase()*** et si ça existe, on rajoute une ligne à "Listview" avec un format de données bien précis. En effet, les quatre threads viennent remplir un seul et unique "Listview".
- **buttonAdd_Click()**: On peut directement ajouter un tag dans la BDD sans faire la connexion du CAEN. Ca se passe en cas de savoir le ID du tag.

RegisterForm

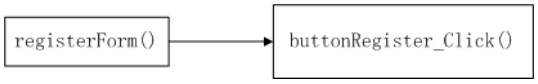

Figure 4.5: structure de la classe registerForm

Cette fenêtre ne fait que la liaison avec la BDD locale, et ajouter les informations dans BDD. Il y a deux façons pour la rappeler: écrire le ID manuel et transmettre le ID par détection.

4.1.3 Gestion de la communication

Pour faire la communication avec fil, il y a deaux choix: par TCP/IP et par RS232. Il ne sera utilisé qu'un seul socket pour la communication des quatre antennes, utilisant le protocole TCP. Une seule et unique connexion est amplement suffisante pour faire transiter les différentes informations voulues.

Le choix du protocole TCP par rapport au protocole UDP s'est fait par soucis de simplicité. En effet la création d'un socket et son utilisation est plus facile et de nombreux tutoriels sont disponibles sur internet. Voici le schéma de communication entre le CAEN et PC:

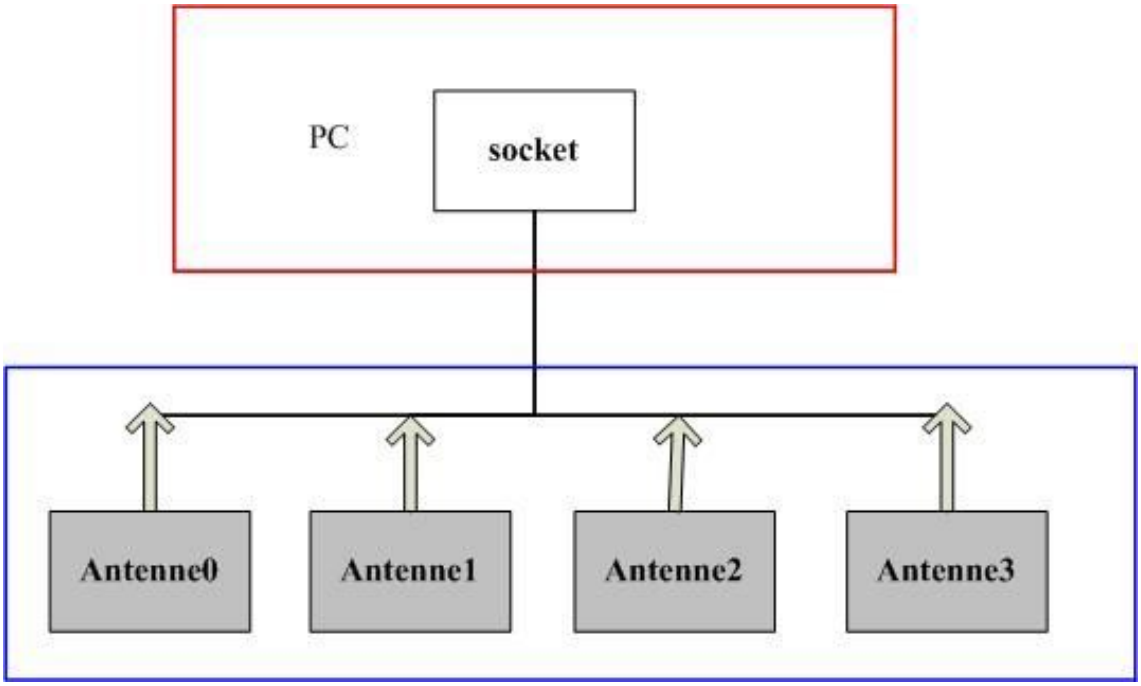

Figure 4.6: communication

4.1.4 Conclusion

L'objectif premier était d'avoir une application pc standard avec une BDD locale joint(voir partie suivante). Nos applications sont fonctionnelles mais le principe est chaque cycle de "Timer" on teste tous les antennes et traite les informations tout de suite.

En effet, on modifie à chaque 250ms, on faire un cycle d'enregistrement. Les antennes détectent les tags au hasard. S'il y a trop de tags présents à le même temps, ça va causer peut-être du manque des tags. Le plus court de laps de temps, le plus lourde de tâche fait par programme. Sur l'énrgie d'antenne, nous ne modifions pas dans le programme.

Parmi les essais, il y a eu une version lourde qui était fonctionnelle, car chaque fois quand il obtient un ID de tag, il va chercher les information dans le base de donnée et faire afficher. Il faut améliorer pour donner une version plus simple, épurée, et avec moins de threads.

4.2 Partie Base de Donnée(Access 2007)

4.2.1 Récupération des données du CAEN

Une fois la communication effectuée et fonctionnelle, les données sont envoyées du CAEN vers le PC au travers d'une chaine de caractères. Voici le format de la chaine de caractère envoyé du CAEN vers le PC: "TagID;Date et Heure;Antenne\n "
Elle contient les données suivantes:

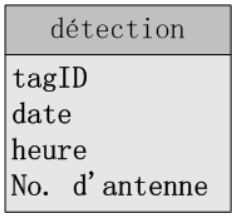

Figure 4.7: donnee de la detection

Exemple de trame envoyée :

TagID	*Date et Heure*	*Antenne*
E2-00-83-19-19-0F-01-22-11-50-A0-61	2013-6-5 17:23:11	Ant0

Table 4.1: Tableau de trame envoyée

Grâce au séparateur ';', les informations sont facilement identifiables et peuvent être manipulées. Une fois la trame reçue, elle est stockée dans un vector de détection. Un vector est un tableau dont la taille est dynamique et croît selon le besoin. De plus, les vectors possèdent des fonctions prédéfinies comme (add, sort ...) ce qui facilite la manipulation de données. La détection est expliqué dans la fonction ***timerRecord_Tick()*** de la partie Front-end ci-dessus.

4.2.2 Traitement des données reçues et gestion de la base de données

Nous allons maintenant détailler les actions effectuées, par le serveur, après la réception des données de détection des tags. Nous allons aussi étudier l'identification d'une personne par un tag. La fonction ***scanner_datebase()*** contient deux paramètres : *string id* et *CAENRFIDTag tag*. Elle représente un utilisateur porteur d'un tag. Elle contient les données suivantes :

Figure 4.8: donnee du porteur badge

Cette fonction permet de faire la jointure entre les données propres à un individu et le ID du tag qui lui est attribué. Les personnes utilisant le système sont enregistrées dans un fichier *.accdb*. Ce fichier permet de stocker les informations des personnes sous un format structuré. Cette structure facilite la lecture des données enregistrées.

La méthode ***MyTags[i][j].GetId()*** de la fonction ***timerRecord_Tick()*** retourne alors les informations de l'utilisateur correspondant au numéro de tag.Les données reçues du programme serveur sont enregistrées au sein du "Listview". Il permet de stocker l'ensemble des données reçues du programme client et les données extraites de la base de données des personnes.

```
1 ListViewItem lvi=new ListViewItem(new string[]{idTags,prenom,nom,
    No,major,DateTime.Now.ToString(),tag.GetReadPoint()});
```

Cette phrase permet de formuler les informations de tag, et afficher tous les attributs dans le “Listview”. Exemple de chaque ligne(lvi ci-dessus) du “Listview”:

IDTags	*Prenom*	*Nom*	*studientNo*	*Major*	*PresentTime*	*SalleNo*
E2-00-83-19-19-0F-01-22-11-50-A0-61	Paul	Martin	20215842	DI3	2012-2-5 9:45:30	Ant 2

Table 4.2: Tableau de trame envoyée

Chaque détection peut alors être affichée sur l'interface du programme. Dans les informations de SalleNo,il permet de détecter 4 salle(présenter par numéro de 0 à 3) au même temps.

4.2.3 Archivage(ne pas fait)

Malheureusement, on ne fait pas les historiques de présents dans ce programme. Mais, c'est mieux de l'ajouter .L'archivage s'effectue dans un fichier au format CSV de manière journalière.

En effet, le fait de générer un fichier par jour offre la possibilité de stocker de manière simple et efficace les données afin que ces fichiers puissent être consultés rapidement par l'utilisateur.

Ces fichiers d'historique seront consultables via un tableur tel que Microsoft Excel, OpenOffice Calc, Google Documents, etc. A chaque démarrage, le programme charge le fichier CSV de la journée et l'affiche dans le tableau de la fenêtre principale.

4.2.4 Conclusion

Afin de conclure cette partie serveur nous allons revenir sur les points qui diffèrent des spécifications originelles. Tout d'abord, la communication entre le CAEN RFID et le PC se fait via uniquement un seul socket.

On prend la façon à cause de étudier le logiciel de CAEN(“CAEN RFID easy controller" téléchargé sur leur site) Cette décision a été prise suite à la rencontre avec le CAEN. Mais, ça va peut-être causer la manque des tags quand il y a trop de tags présents au même temps. On donne une amélioration de cette question(voir la partie suivante).

Pour finir, le point n'ayant pas été réalisé est le système de sélection de période et les historiques de décection.(par exemple sélection de la période du 09/09/12 à 8h00 jusqu'au 10/09/12 à 12h30). Cet aspect du programme n'a pas été retenu du fait des nombreuses possibilités qu'offre la manipulation de données des logiciels tableurs.

4.3 Partie d'amélioration de programme

Pour résoudre le cas de la manque du tag expliqué dessus. On pourra changer la solution de la communication de la base de donnée.

Notre solution : chaque cycle de détection, on traite tous les tags dans le tableu de CAENRFIDTag[] et chercher dans la base de donnée. Si on trouve les informations, on l'ajoute dans le "Listview". On peut modifier le laps de temps de "Timer" pour exécuter le cycle de détection. Le plus grand le laps de temps, le plus possibilité de perdre des tags. Si on le modifie très court, ça fait très lourd de programme.

Solution améliorée : on peut programmer dans le CAEN et d'abord stocker les tags détectés dans une file. Ensuite, On traite les tags dans la file et cherche les informations dans la base de donnée. On sépare l'étape de détection et l'étape de traitement des tags. Comme ça, on peut détecter des tags plus vite que notre solution et perdre moins de tags.

Voici la comparaison des deux solutions :

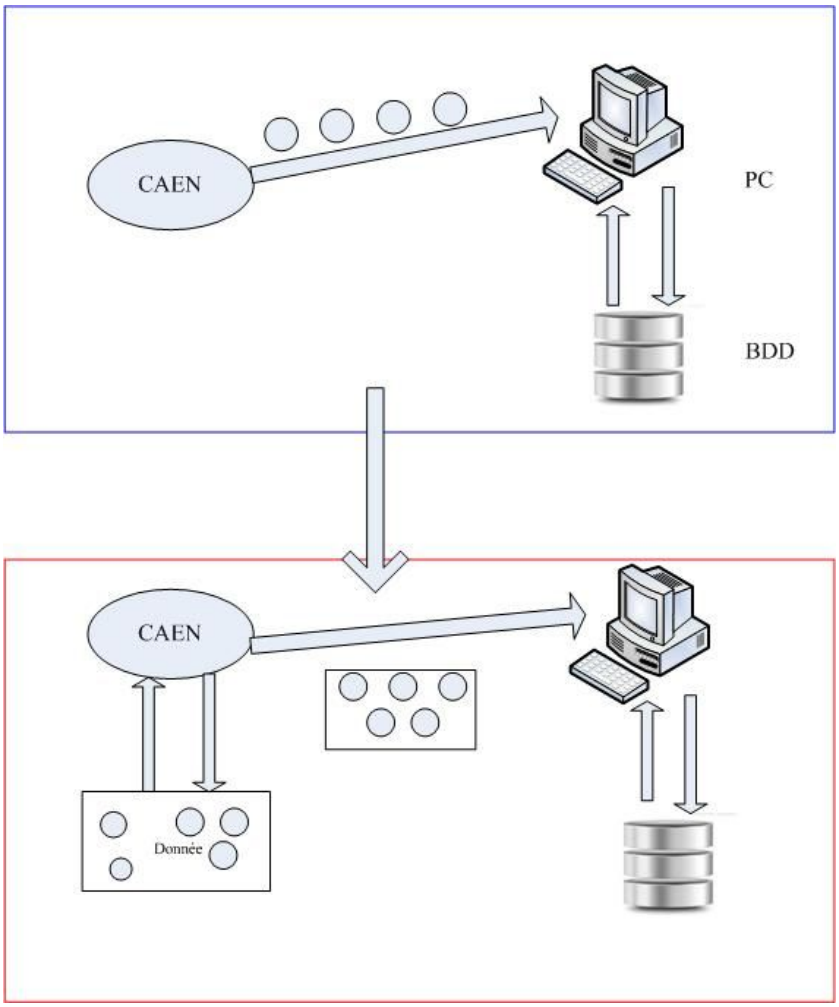

Figure 4.9: comparaison des deux solutions

On n'a pas encore réalisé le solution améliorée dans ce projet.

Conclusion

5.1 Bilan technique

Techniquement nous n'avons pas mis en place tout ce que l'on aurait voulu. Nous avons réalisé la partie de la détection en temps réel avec 4 antennes. Nous pourrons mettre 4 antennes dans 4 salles différentes. Donc en même temps, ça permet de surveiller "présences/absences" de toutes les 4 salles.

Mais s'il y avait trop de tags en même temps, ça pourrait causer la manque du tag. Pour résoudre ce problème, nous avons trouvé une solution améliorée pour le traitement des informations détectées(expliqué ci-dessus).

5.2 Bilan personnels

Lin SHEN:

C'est un projet très intéressant car c'est le premier fois que je fais un projet avec les objets matériels. Je pense qu'on a été un groupe soudé, ce qui nous a permis d'avancer rapidement. On rencontre quelque problèmes dans le processus. Comme tout à bord, on ne réussit pas faire la connexion entre reader et mon ordinateur.

J'ai bénéficié de ce projet beaucoup. Par exemple, j'ai appris le langage C#, écrit la rapport avec Latex. Mais plus important, j'ai appris comment résoudre le problème. Et l'encadrant est très patient de nous aider à expliquer le projet.

Peng BAI:

J'aime beaucoup ce projet. Parce que je peut faire quelques choses pratiques dans ma spécialité. Au niveau de connaissance, j'ai eu beaucoup de problèmes pendant la création de programme. Heureusement, la société CAEN a sorti un logiciel fonctionnel(Easy Controller) et nous donné le code source. Quand j'ai écrit le programme avec le C#, j'ai pu regarder le code et trouver la solution. C'est intéressant de voir les tags détectés apparaître sur mon ordinateur.

Et pour bien faire ce projet et écrire la rapport, je pense que mon niveau de français est aussi augmenté. Merci pour les aides de M.Delalandre, Alice, Florian et Mme. Cylvie pour m'emprunter la clé de casier.

Référence

1. DII4_Geocalisation_RapportV3.pdf(Polytech Tours)
2. ION_R4300P_Technical_Information_Manual_Rev_02.pdf
3. CAENRFID_API_Reference_Manual.pdf
4. CAENRFID_API_User_Manual.pdf
5. CAEN_UHF_RFID_Readers_Communication_Protocol_Manual.pdf
6. Easy Controller_Software_Technical_Information_Manual.pdf

Relevé automatique de présence sous système RFID UHF

Département Informatique
3e année
2012 - 2013

Rapport de projet de Système d'exploitation

Résumé : Ce projet s'intéresse aux technologies de radio indentification i.e. RFID .Dans ce projet on s'intéressera à la problématique de relevé de présence au sein d'un institut d'enseignement.

Mots clefs : RFID UHF -système de présence

Abstract: This project focuses on radio identification technology ie RFID. In this project we will focus on the issue of attendance record within an educational institute.

Keywords: RFID UHF -Presence System

Encadrants
Mathieu Delalandre
mathieu.delalandre@univ-tours.fr

Université François-Rabelais, Tours

Etudiants
Lin SHEN
lin.shen@etu.univ-tours.fr
Peng BAI
peng.bai@etu.univ-tours.fr

DI3 2012 - 2013