

Projet SEVESO



Arrad Youssef
Ahmed Kassim Nail
Debaa Nadir
Baiche Othmane

Sommaire

<u>INTRODUCTION</u>	3
<u>PRESENTATION DU PROJET</u>	5
<u>REPARTITION DES TACHES</u>	7
<u>ANALYSE UML</u>	8
1. DIAGRAMME DE DEPLOIEMENT :	8
2. DIAGRAMME DES CAS D'UTILISATIONS :	10
 <u>ARRAD YOUSSEF</u>	 13
I - Analyse UML	
a - Diagramme de déploiement	
b - Diagramme des cas d'utilisation	
c - Diagramme des séquences	
d - Diagramme des classes	
II - Modélisation de la base de données	
a - Présentation de la base de données	
III - IHM	
a - Aspect ergonomique	
b - Onglet ADD	
c - Onglet DELETE	
d - Onglet MODIFY	
IV - Protocole d'échange de données	
V - Annexe	
 <u>AHMED KASSIM NAIL</u>	 42
I - Présentation du matériel	
II - Mise en place du matériel	
a - Dispositif	
b - Diagramme de déploiement personnel	
III - Contexte	
a - Objectifs	
b - Diagramme des cas d'utilisations général et personnel	
c - Diagramme de séquence	
IV - Modules	
a - Protocole d'échange entre les sous réseaux	
b - Diagramme des classes	
c - Annexe	
V - Démonstration	

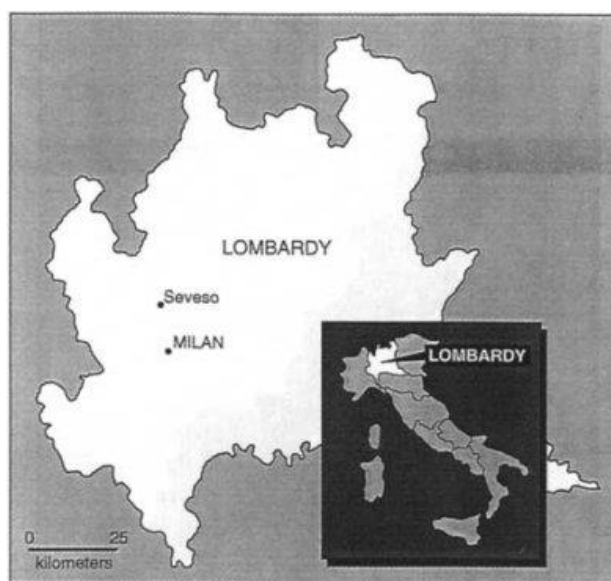
<u>DEBAA NADIR</u>	76
I - Analyse uml	
II -Matériel utilisé	
III - Protocole d'échange	
IV - Lecteur Vellman et raspberry	
a - Configuration	
b - Lecture	
V - LCD	
a - Configuration	
VI - Socket	
a - Création	
b - Envoyé et recevoir des données	
c - Serveur	
VII - Wireshark	
VIII - Annexe	
<u>BAICHE OTHMANE</u>	107
I - Analyse uml	
II -Matériel utilisé	
III - Protocole d'échange	
IV - Lecteur MD-003	
a - Configuration	
b - Lecture	
V - LCD	
a - Ouverture et fermeture	
VI - Socket	
a - Création	
b - Envoyé et recevoir des données	
c - Serveur	
VII - Wireshark	
VIII - Annexe	

Introduction

La **directive Seveso** est le nom générique d'une série de directives européennes qui imposent aux États membres de l'Union européenne d'identifier les sites industriels présentant des risques d'accidents majeurs, appelés « sites SEVESO », et d'y maintenir un haut niveau de prévention. Cette directive tire son nom de la catastrophe de Seveso qui eut lieu en Italie en 1976 et qui a incité les États européens à se doter d'une politique commune en matière de prévention des risques industriels majeurs.



Incendie de l'usine chimique de Seveso (Italie), 1976.



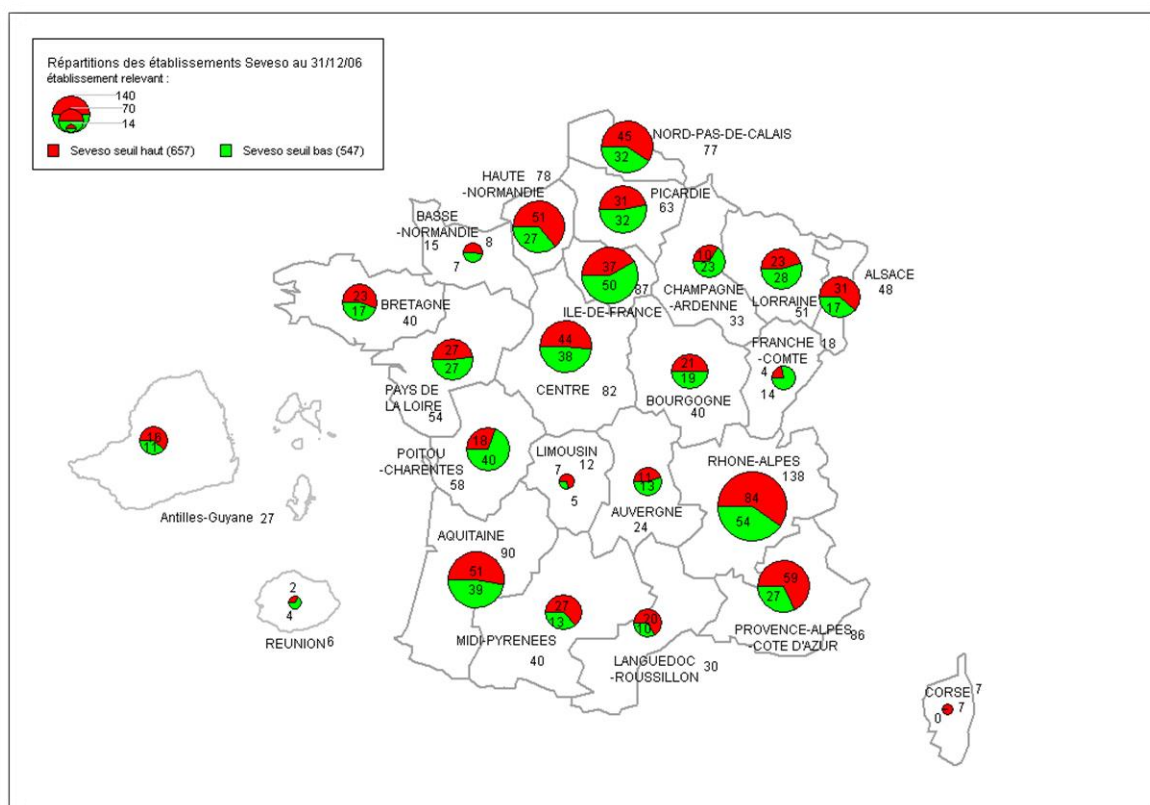
Les sites classés Seveso sont des installations industrielles dangereuses répertoriées selon le degré des risques qu'elles peuvent entraîner.

Toutes les installations Seveso doivent faire l'objet d'une stricte surveillance de la part de l'exploitant et des autorités publiques. Un plan d'urgence interne et un plan d'urgence externe doivent être établis. Des personnes compétentes doivent être capables de prendre immédiatement les bonnes décisions en cas d'accident.

Il s'agit de « *Plans de prévention des risques technologiques* » (PPRT), permettant si nécessaire d'exproprier les habitants dans les zones les plus dangereuses. Et d'obliger les industriels à réduire les risques à la source dans les entreprises. La population doit également être mise au courant des activités de l'usine.

Norme SEVESO

Selon le dernier recensement du 31 décembre 2014, il existe 1 171 sites relevant de la directive Seveso en France. La réglementation introduit deux seuils de classement selon la « *dangerosité* » des sites suivant la quantité de substances dangereuses utilisées: « **Seveso seuil bas** » (risque important – 515 établissements) et « **Seveso seuil haut** » (risque majeur – 656 établissements).



Présentation du projet

Les normes de sécurité des bâtiments SEVESO prévoient maintenant de connaître, à tout moment, le nombre et l'identité de toutes les personnes présentes dans le bâtiment : en cas de catastrophe, les secours et autorités sauront exactement combien de personnes doivent être secourues, leurs identités, et qui prévenir.

Chaque personne du bâtiment se voit remettre un tag RFID à son nom en échange de ses papiers d'identité. Associations stockées dans une base de données.

Le visiteur récupère la pièce d'identité à la sortie.

Le système comporte 3 postes entrant/sortant (2 entrants et un sortant) équipés chacun d'un lecteur RFID, d'un portillon dont l'ouverture est autorisée ou non en fonction du contenu de la carte RFID, Les fonctions 'entrant/sortant' recueillent les informations de la carte et enregistrent l'entrée ou la sortie de l'utilisateur (date, heure...).

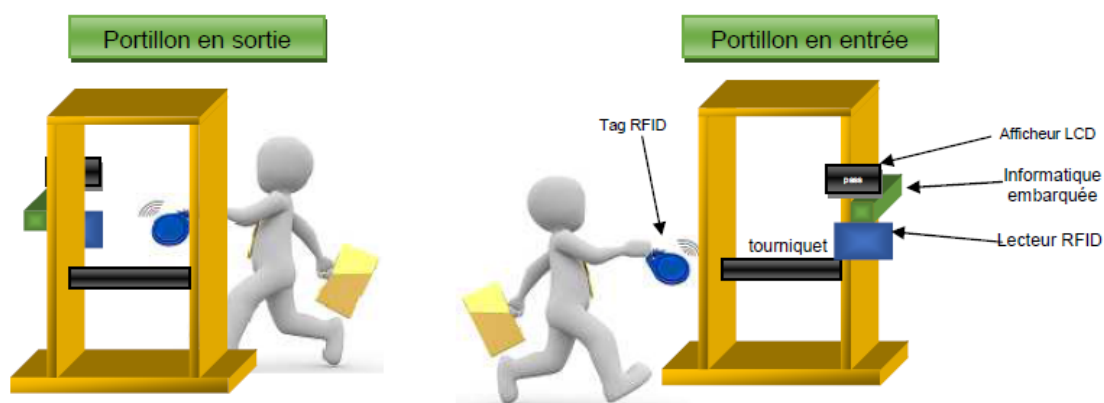
Un poste de supervision enregistre les entrées/sorties des utilisateurs associés aux cartes RFID, il mémorise ces informations dans une base de données et fournit la liste des personnes présentes en temps réel, Ces informations sont accessibles via une application de type IHM.

Le poste de supervision doit pouvoir en cas d'évacuation, autoriser l'ouverture simultanée de tous les postes entrant/sortant.

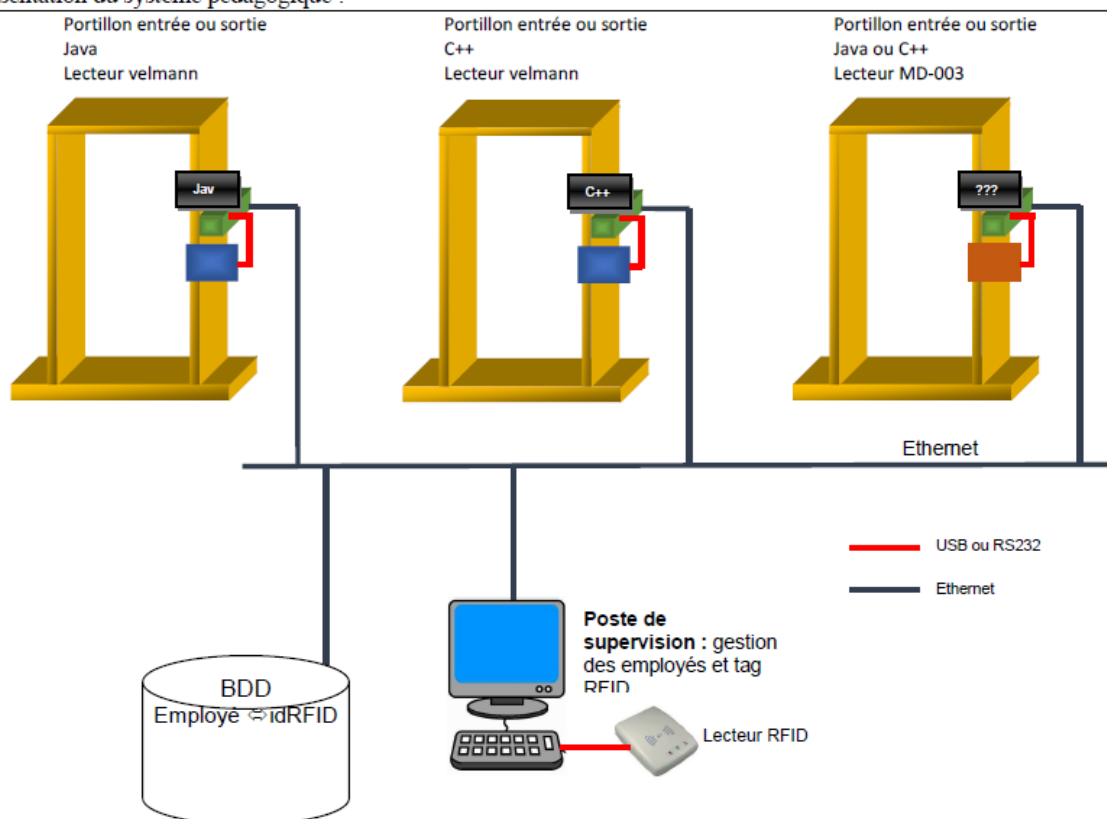
Le poste de supervision doit aussi permettre d'ajouter, ôter de nouvelles associations employé et idRFID dans la base de données.

L'ouverture et la fermeture des tourniquets seront simulées par différents messages sur un afficheur LCD connecté à l'informatique embarquée.

Bâtiment à la norme SEVESO



Présentation du système pédagogique :



Répartition des tâches

Etudiant 1 ARRAD Youssef

- TI-11 : Analyser les données à collecter et stocker.
- TI-12 : Modéliser la base de données.
- TI-13 : Construire et peupler la BDD.
- TI-14 : Coder un module logiciel permettant de gérer la BDD conformément à l'analyse faite en TI-11.
- TI-15 : Définir et coder l'IHM (ergonomie uniquement) permettant à l'opérateur de saisie d'accéder aux services.
- TI-16 : Intégrer le module de TI-14 dans l'IHM.
- TI-17 : Coder un module de communication (serveur) permettant de recevoir les informations issues des tags lors d'un passage (entrée ou sortie).
- TI-18 : Coder le module de communication réseau (client) afin d'envoyer à tous les portillons un ordre d'ouverture ou de fermeture des portillons.
- TI-19 : Coder le module de rafraichissement de l'affichage des personnes présentes dans l'entreprise afin d'avoir un affichage temps réel.

Etudiant 2 AHMED KASSIM Nail

- TI-21 : Mettre en œuvre le matériel RFID Velleman K8019 avec les outils du fournisseur.
- TI-22 : Coder un module logiciel (JAVA) de collecte des informations utiles du tag RFID.
- TI-23 : Coder un module logiciel (JAVA) de commande de l'afficheur LCD.
- TI-24 : Coder le module (JAVA) de communication réseau (client) afin d'envoyer au poste de supervision le tag RFID présenté devant le lecteur.
- TI-25 : Coder un module (JAVA) de communication (serveur) permettant de recevoir les ordres d'ouvertures/fermetures en cas de catastrophe.
- TI-26 : Coder l'application finale (JAVA) en intégrant l'ensemble des modules précédents.

Etudiant 3 DEBAA Nadir

Mêmes tâches que l'étudiant n°2 développées en C++.

Etudiant 4 BAICHE Othmane

Mêmes tâches que l'étudiant n°2 ou 3, développées sur le lecteur RFID MD-003 en C++.

Tâches		Etudiant 1	Etudiant 2	Etudiant 3	Etudiant 4
TC1					
		x	x	x	x
TC2					
		x	x	x	x
TC3					
		x	x	x	x
TC4					
		x	x	x	x
TC5					
		x	x	x	x
TI-11					
		x			
TI-12					
		x			
TI-13					
		x			
TI-14					
		x			
TI-15					
		x			
TI-16					
		x			
TI-17					
		x			
TI-18					
		x			
TI-19					
		x			
TI-21					
			x		
TI-22					
			x		
TI-23					
			x		
TI-24					
			x		
TI-25					
			x		
TI-26					
			x		
TI-31					

[illegible]

			x	
TI-32			x	
TI-33			x	
TI-34			x	
TI-35			x	
TI-36			x	
TI-41				x
TI-42				x
TI-43				x
TI-44				x
TI-45				x
TI-46				x

La technologie RFID est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes ». Ces radio-étiquettes sont des petits objets qui peuvent être incorporés ou collés, ici on va s'en servir sur un badge (ou carte) RFID.

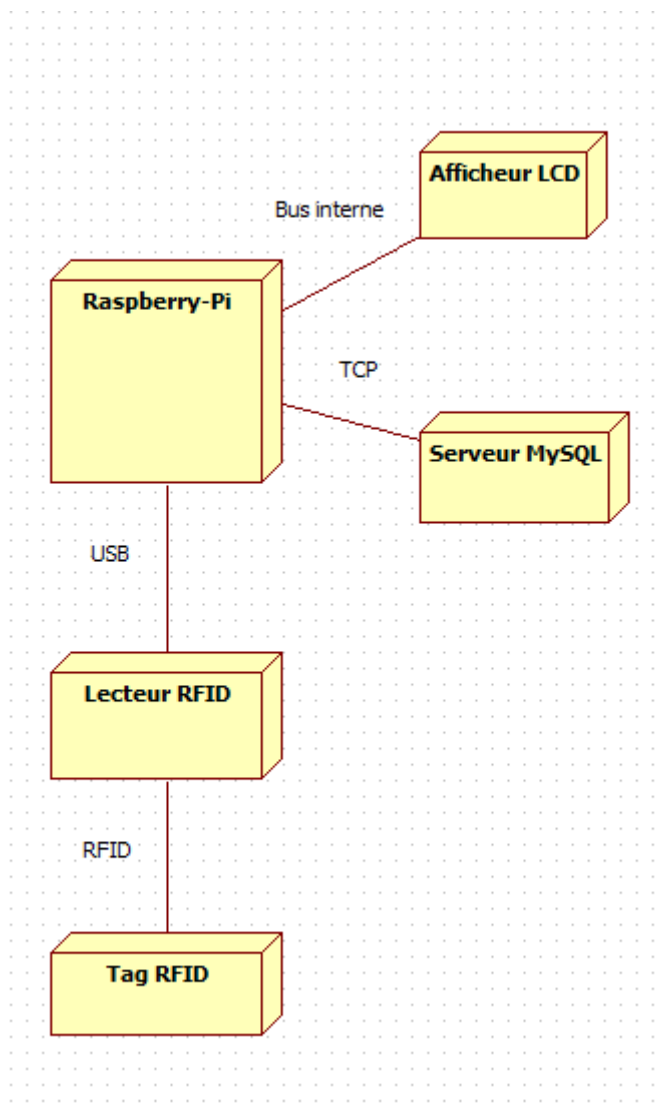
Quand le badge va passer devant le lecteur, celui-ci va envoyer un signal à l'étiquette. L'étiquette retourne alors son identification numérique (Code RFID).

Pour que l'utilisateur puisse accéder au bâtiment, il va passer sa carte devant le lecteur, l'informatique embarquée va récupérer le code RFID et va envoyer ce code au serveur. Le serveur va ensuite interroger la base de données (Base de données MySQL). Ainsi on va pouvoir vérifier que l'utilisateur correspondant à ce code RFID existe réellement.

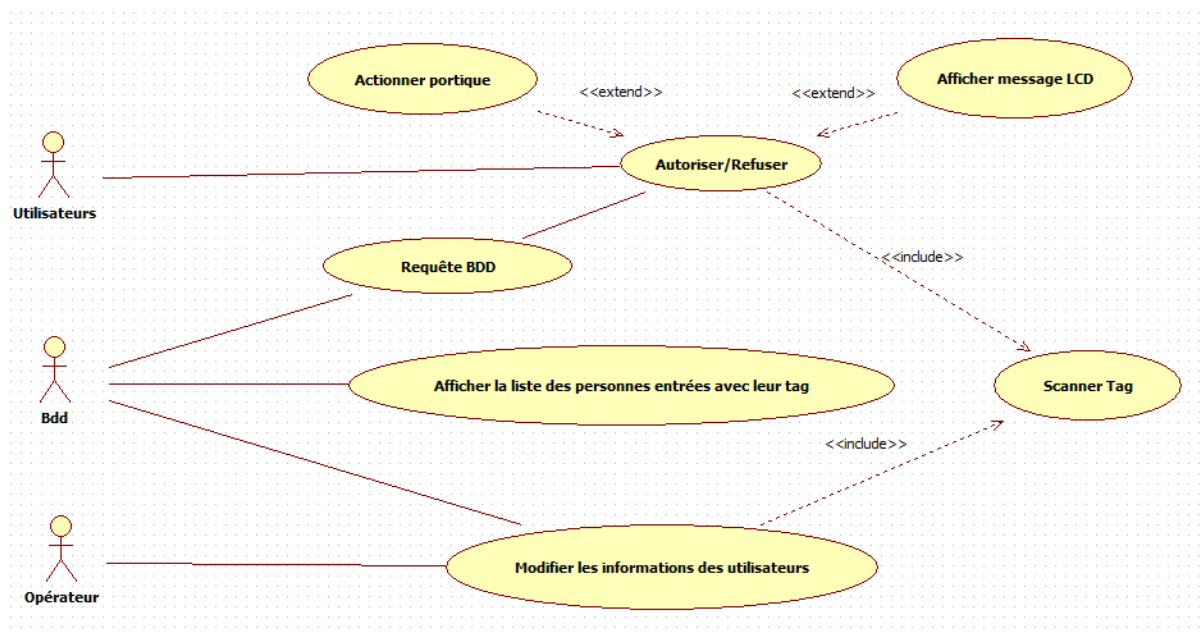
Un afficheur LCD va pouvoir exposer un message, autorisant ou non l'utilisateur à entrer.

Analyse UML

1. Diagramme de déploiement



2. Diagramme de cas d'utilisation



Tâche étudiant n°1

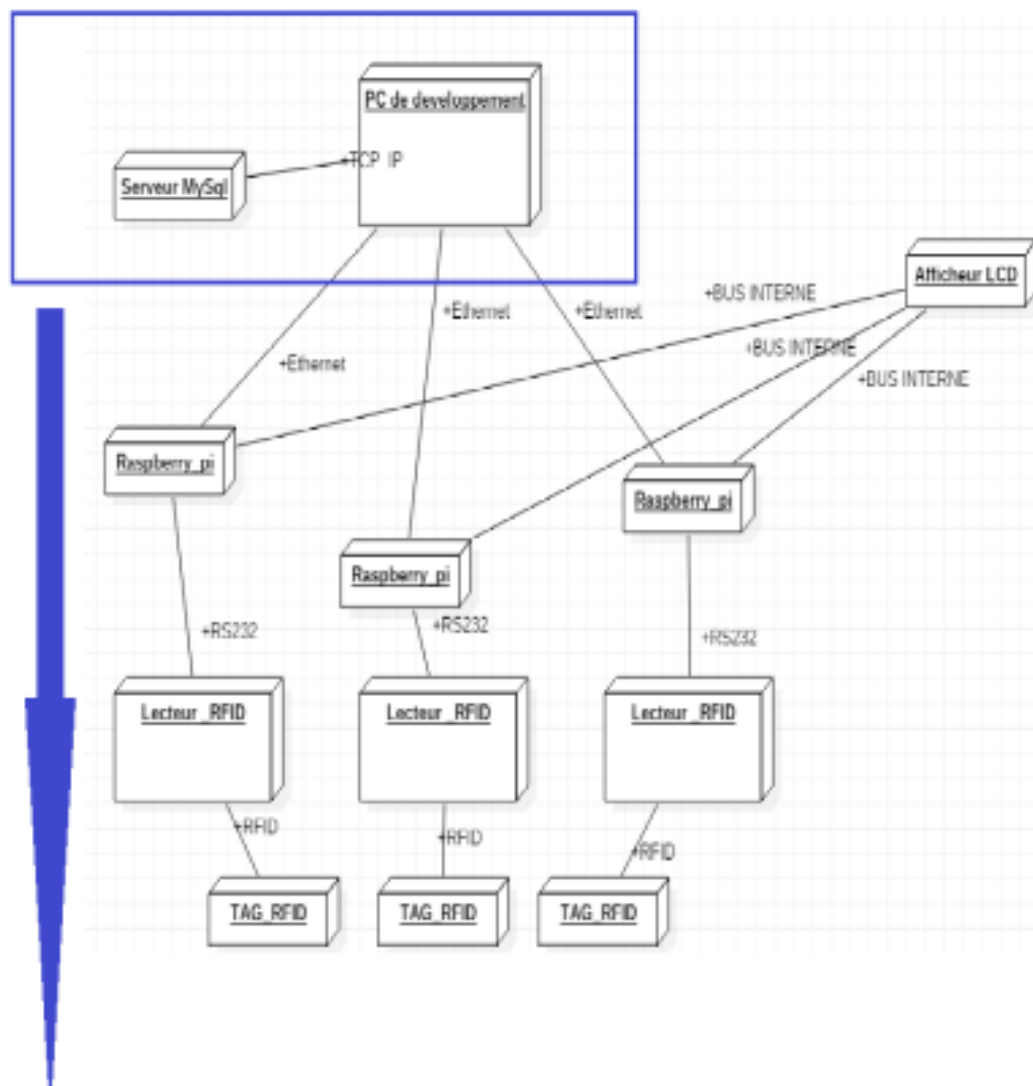
Arrad Youssef

*Base de données
et IHM*

Analyse UML

Suivant les principaux diagrammes UML qui ont permis la conception du système, nous avons :

Le diagramme de déploiement

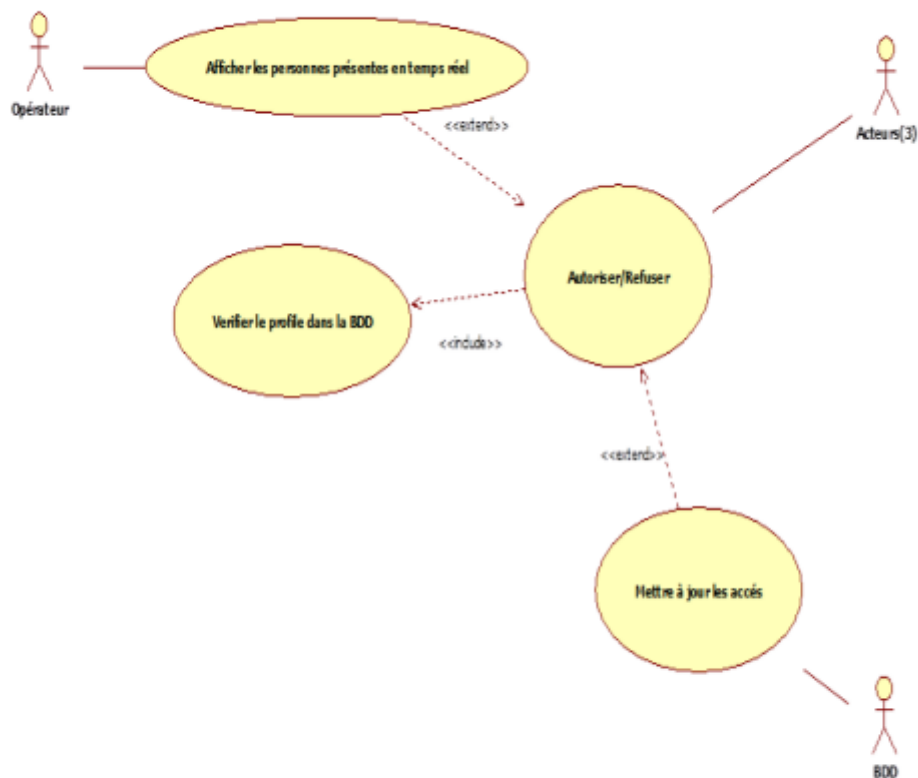


Positionnement
personnel par rapport
à l'étudiant 1

Diagramme des cas d'utilisations

Il y a trois cas d'utilisations possibles le premier cas d'utilisation va concerner les « entrées/sorties », le second concerne les enregistrements de profils ainsi que les modifications de profils. Et enfin le troisième cas d'utilisation représente un éventuel état d'urgence.

Cas d'utilisation 1



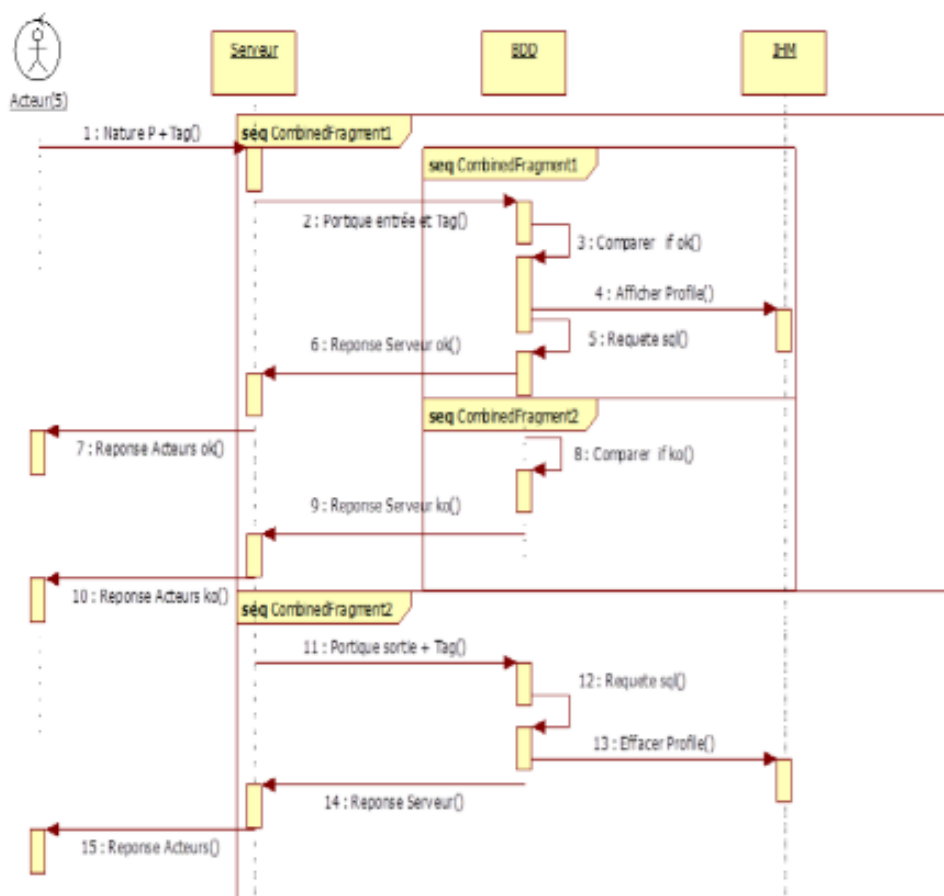
Cas d'utilisation 2



Cas d'utilisation 3

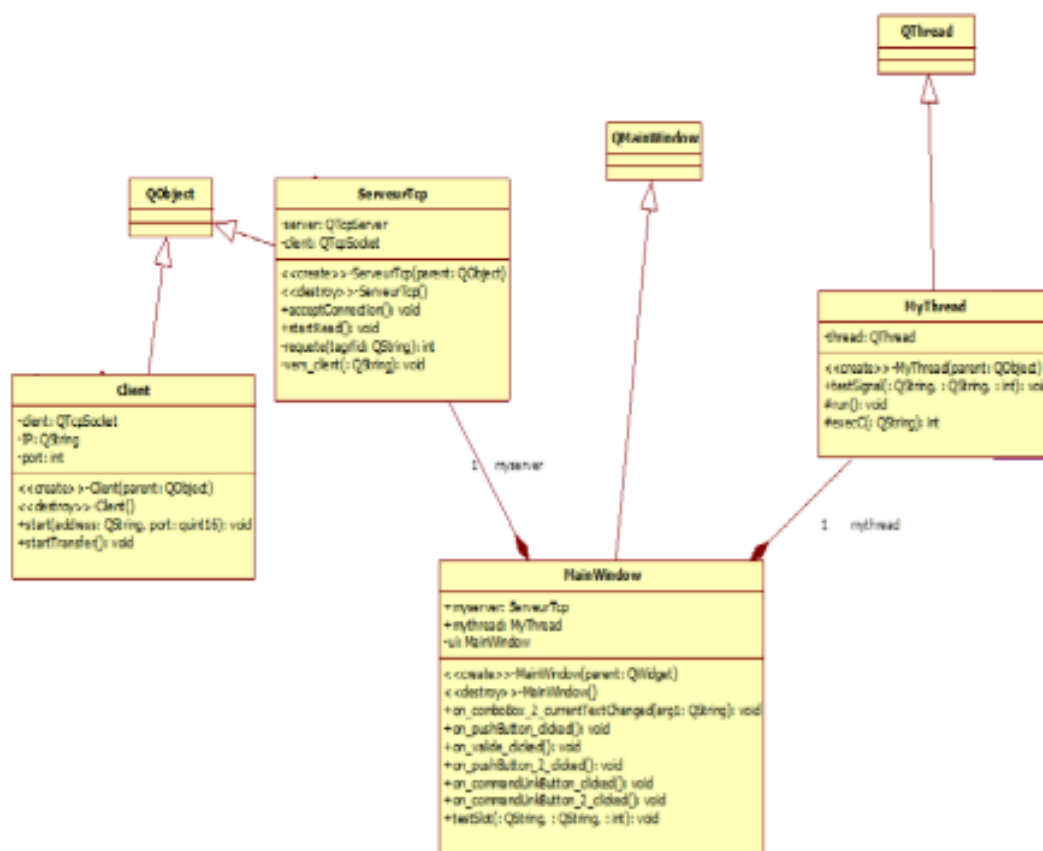


Diagramme des Séquences

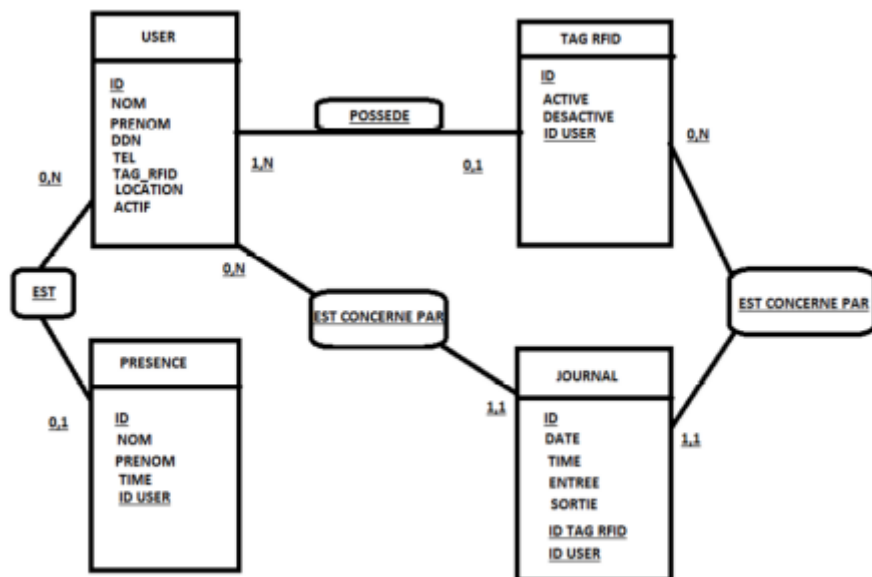


Ce diagramme des séquence est celui du scénario nominal, entre autre l'affichage en temps réel des personnes qui scannent leur tag en entrée.

Diagramme des Classes



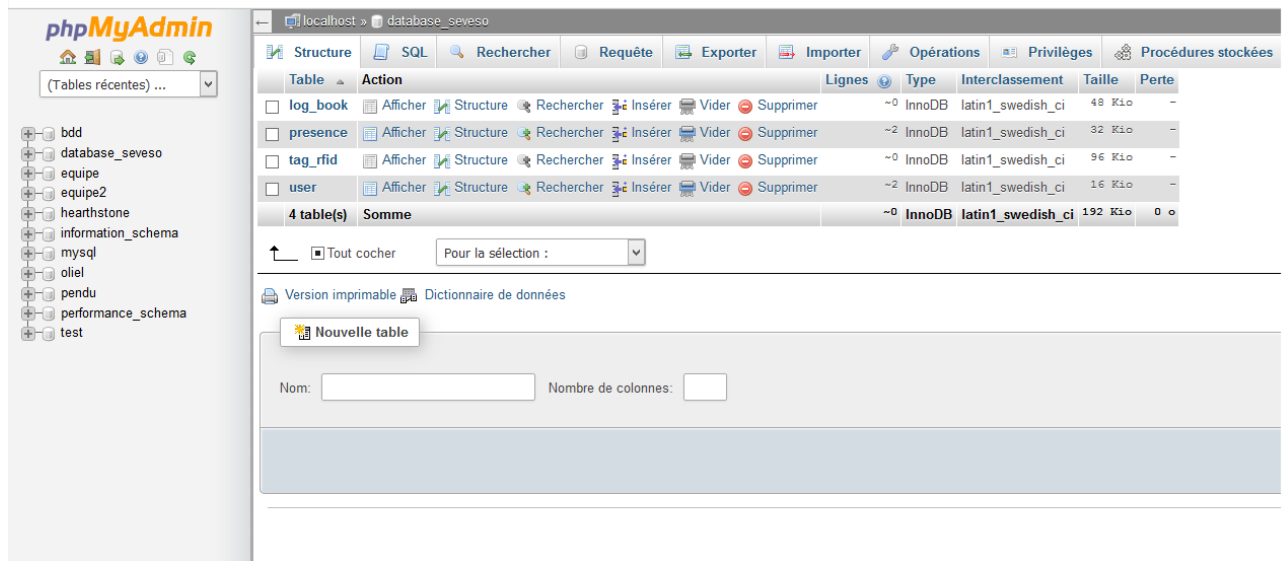
Modélisation de la base de données



La base de données du projet se nomme « database_seveso » et a été créée sur **Wamp Server**. Elle est composée de quatre tables qui gèrent les profils ainsi que la journalisation de ceux-ci.



Ainsi nous avons cet affichage sur « phpMyadmin » :



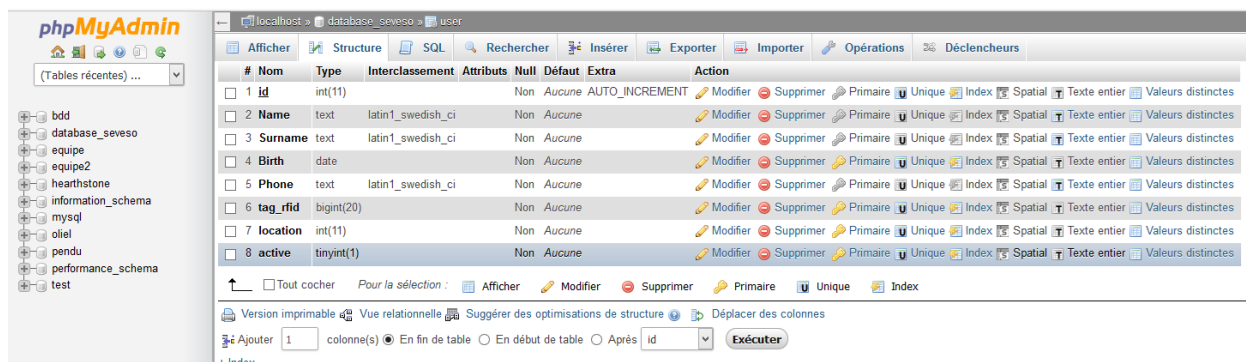
The screenshot shows the phpMyAdmin interface for the 'database_seveso' database. The left sidebar lists various databases including 'bdd', 'database_seveso', 'equipe', 'equipe2', 'hearthstone', 'information_schema', 'mysql', 'oliel', 'pendu', 'performance_schema', and 'test'. The main panel displays the 'Structure' tab for the 'database_seveso' database, showing a list of tables: 'log_book', 'presence', 'tag_rfid', and 'user'. Each table entry includes icons for viewing, structure, search, insert, delete, and update, along with details like 'Lignes' (rows), 'Type', 'Interclassement' (collation), 'Taille' (size), and 'Perte' (loss). Below the table list, there is a section for 'Nouvelle table' (New table) with fields for 'Nom' (Name) and 'Nombre de colonnes' (Number of columns).

L'élaboration de la base de données a été assez simple, cela dit elle a été modifiée à plusieurs reprises.

Une table a été ajoutée (la table « TAG RFID ») qui a permis une lecture plus simple de la base de donnée et ce malgré une redondance calculée.

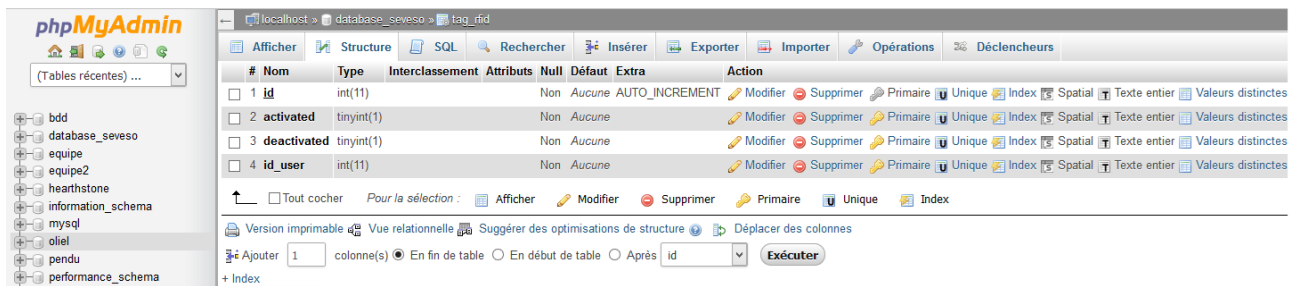
Voici la structure des différentes tables :

La table USER



The screenshot shows the 'Structure' tab for the 'user' table in the 'database_seveso' database. The table structure is displayed with columns: 'id', 'Name', 'Surname', 'Birth', 'Phone', 'tag_rfid', 'location', and 'active'. Each column has a checkbox for selection, a primary key icon, and a unique index icon. The 'id' column is the primary key. The 'tag_rfid' column has a unique index. The 'location' column has a unique index. The 'active' column has a unique index. Below the table structure, there is a section for 'Ajouter' (Add) with a dropdown for 'colonne(s)' (column(s)) and a button 'Exécuter' (Execute).

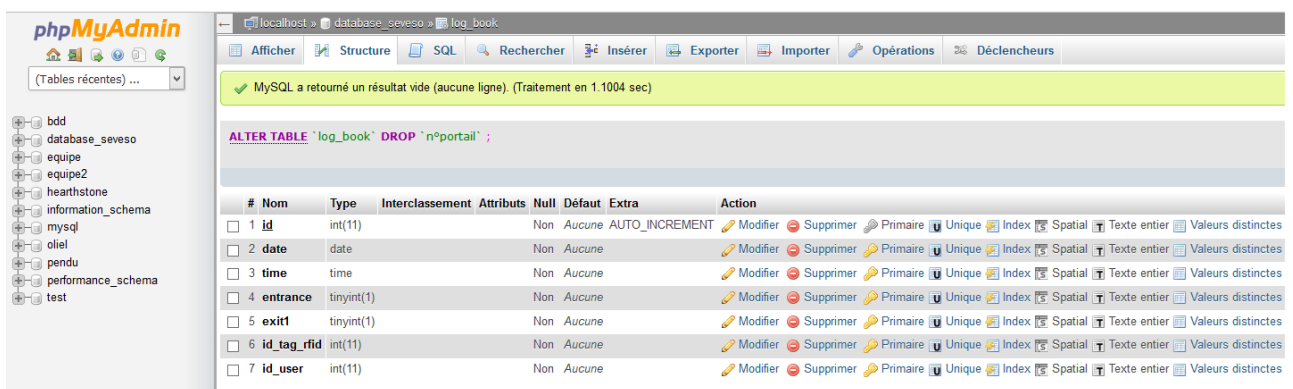
La table TAG RFID



Structure of the **tag_rfid** table:

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
2	activated	tinyint(1)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
3	deactivated	tinyint(1)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
4	id_user	int(11)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes

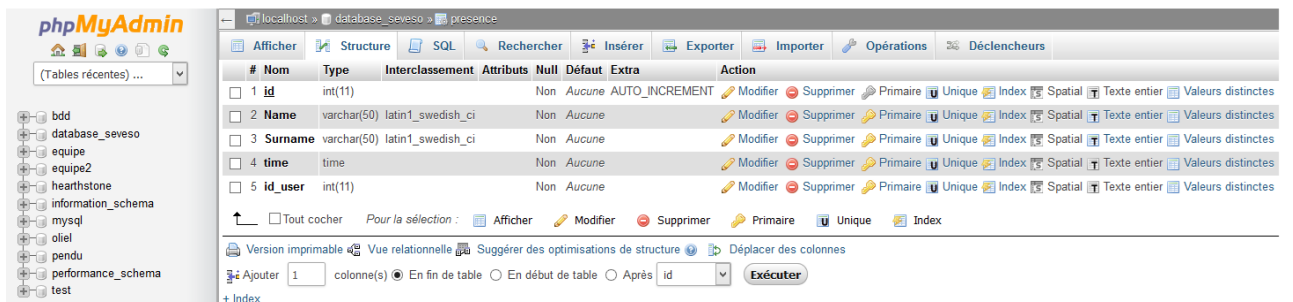
La table LOG BOOK



Structure of the **log_book** table:

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
2	date	date			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
3	time	time			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
4	entrance	tinyint(1)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
5	exit1	tinyint(1)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
6	id_tag_rfid	int(11)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
7	id_user	int(11)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes

La table PRESENCE



Structure of the **presence** table:

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
2	Name	varchar(50)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
3	Surname	varchar(50)	latin1_swedish_ci		Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
4	time	time			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes
5	id_user	int(11)			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier Valeurs distinctes

La table **user** reprend les profiles des diverses personnes qui vont scanner leur Tag.

La table **tag_RFID** va elle rendre compte de l'état du tag (si il est activé ou désactivé).

La table **log-book** va permettre la journalisation des accès.

La table **presence** est sollicitée directement sur l'IHM. En effet, elle permet

l'affichage en temps réel des personnes présentes dans le bâtiment.

L'IHM a été modélisée sur l'IDE **Qt Creator VERSION 5.3.2** avec le langage **C++**.

Le système d'exploitation est **Windows 8**.

La fenêtre est composée de quatre onglets.

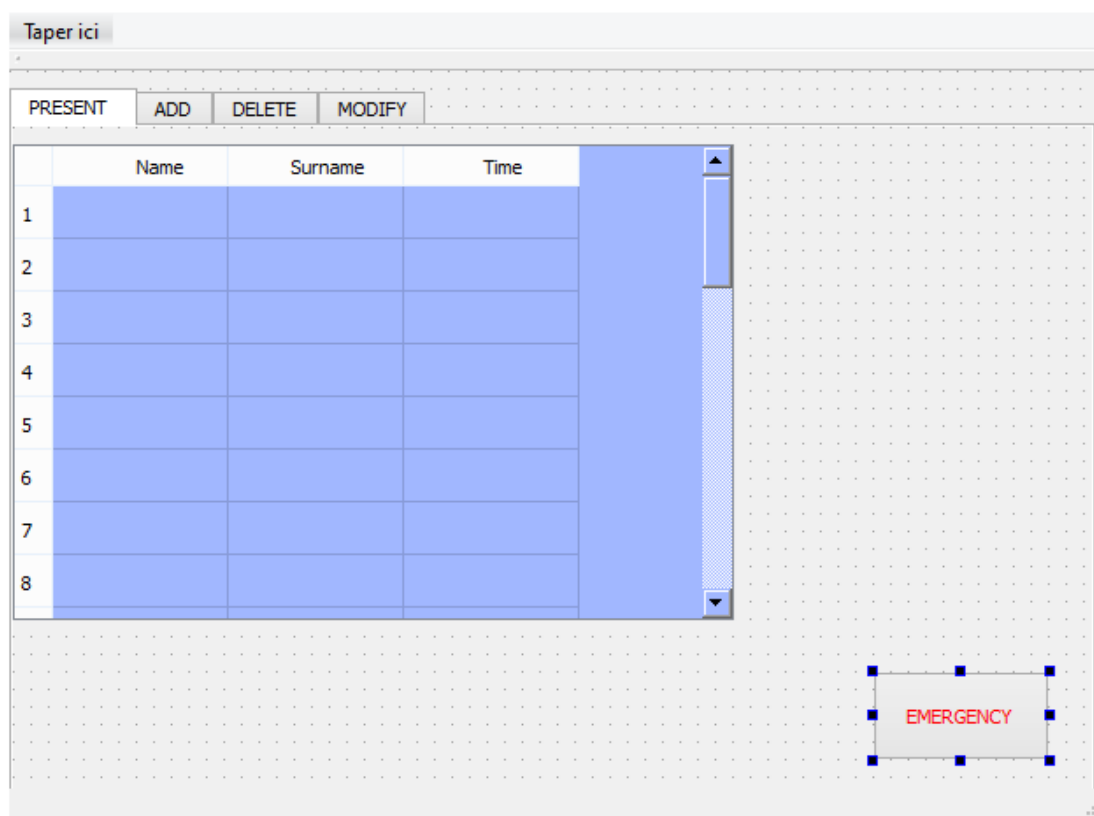
Un onglet **PRESENT** qui affiche les personnes présentes et ce à l'aide d'une requête SQL.

Un onglet **ADD** qui permet d'ajouter un nouveau profile dans la base de données.

Un onglet **DELETE** qui permet de supprimer un profile de la base de données .

Un onglet **MODIFY** qui permet de modifier un profile déjà existant dans la base de données.

Voici l'aspect ergonomique de l'IHM :

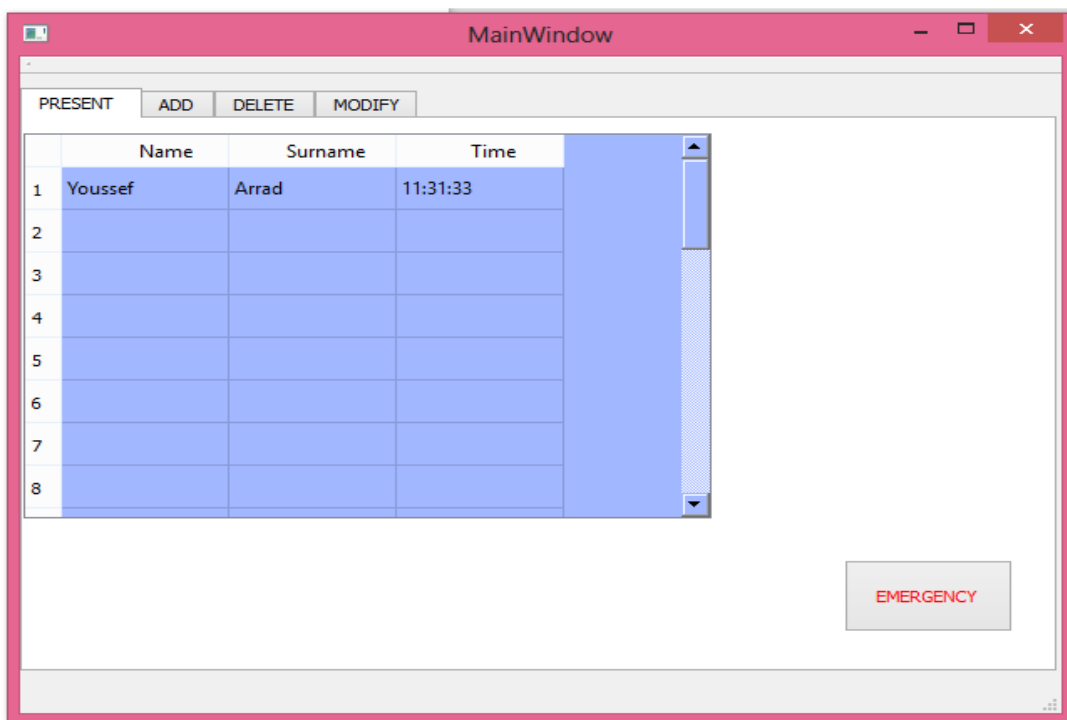


Le bouton **EMERGENCY** a pour but de gérer un état d'urgence,

malheureusement par manque de temps ce cas d'utilisation n'a pas été réalisé.

Le **QtableWidget** affiche les personnes présentes selon les critères (Name, Surname, Time)

ainsi lorsqu'une personne est inscrite dans la table **presence** de la base de données, elle est immédiatement affichée (voir ci-dessous).



Voici le code utilisé pour remplir le **QtableWidget** :

```
//remplissage du qtablewidget

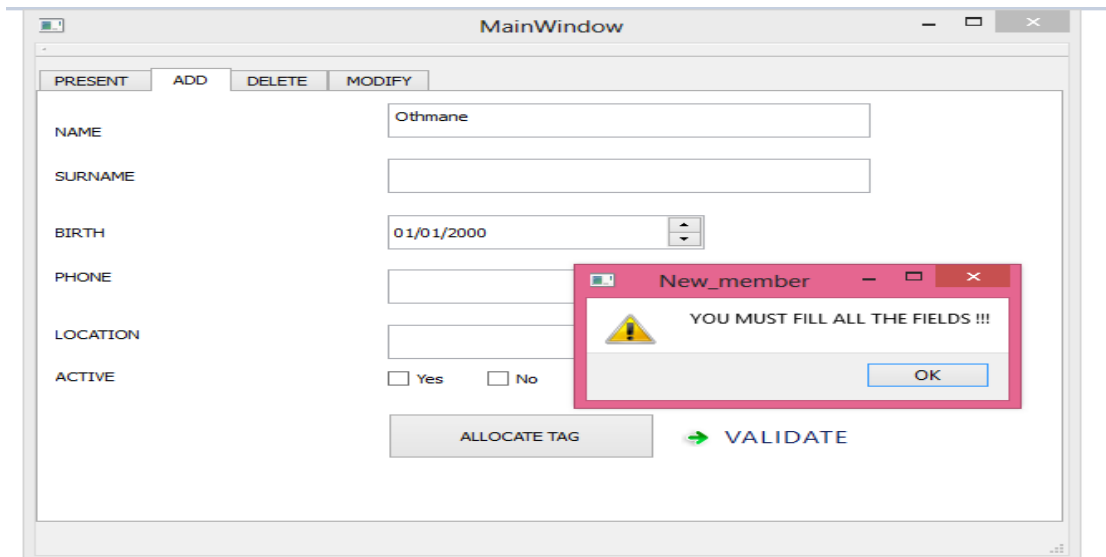
QTableWidgetItem *cubesHeaderItem = new QTableWidgetItem(a);
ui->tableWidget->setItem(n, 0, cubesHeaderItem);

cubesHeaderItem = new QTableWidgetItem(b);
ui->tableWidget->setItem(n, 1, cubesHeaderItem);

cubesHeaderItem = new QTableWidgetItem(c);
ui->tableWidget->setItem(n, 2, cubesHeaderItem);
```

Onglet ADD

Il est impératif de remplir tout les champs autrement la validation ne peut pas s'effectuer .



Le bouton **ALLOCATE TAG** permet d'attribuer un tag RFID au futurs profiles mais je n'ai pu le coder par manque de temps.
la requête utilisée pour enregistrer les profiles saisis est la suivante :

Une fois le profile entré on clique sur le bouton **validate** et le profile est aussi tôt enregistrer dans la table **user** de la base de données

```
void MainWindow::on_valide_clicked()
{
    //on recupere la date
    QDate date = ui->dateEdit->date();
    //on recupere les données de type string
    QString N = ui->textEdit->toPlainText();
    QString S = ui->textEdit_2->toPlainText();
    QString P = ui->textEdit_4->toPlainText();
    QString l = ui->textEdit_9->toPlainText();
    bool a = ui->checkBox_3->isChecked();
    bool b = ui->checkBox_4->isChecked();
    int c = 1;
    int d = 0;

    //on recupere la valeur cochée dans la check box

    //si tout les champs sont remplis on enregistre sinon non
    if(date,N,S,P,l,a || b == true){

        QSqlQuery qry;
        //une fois le texte recuperé on insère les données dans la BDD
        qry.prepare("INSERT INTO user(Name,Surname,Birth,Phone,location,active) "
                    "VALUES (:Name, :Surname, :Birth, :Phone, :location, :active)");
        qry.bindValue(":Name",N);
        qry.bindValue(":Surname",S);
        qry.bindValue(":Birth",date);
        qry.bindValue(":Phone",P);
        qry.bindValue(":location",l);
        if (a == true){
            qry.bindValue(":active",c);
            qDebug() << "a="+a;
        }else{
            qry.bindValue(":active",d);
            qDebug() << "b="+b;
        }
    }
}
```

MainWindow

PRESENT ADD DELETE MODIFY

NAME Othmane

SURNAME Baiche

BIRTH 01/01/2000

PHONE 0145967820

LOCATION 23

ACTIVE ☒ Yes ☐ No

ALLOCATE TAG → VALIDATE

you are registered !
OK

Onglet DELETE

MainWindow

PRESENT ADD DELETE MODIFY

nadir

→ Delete

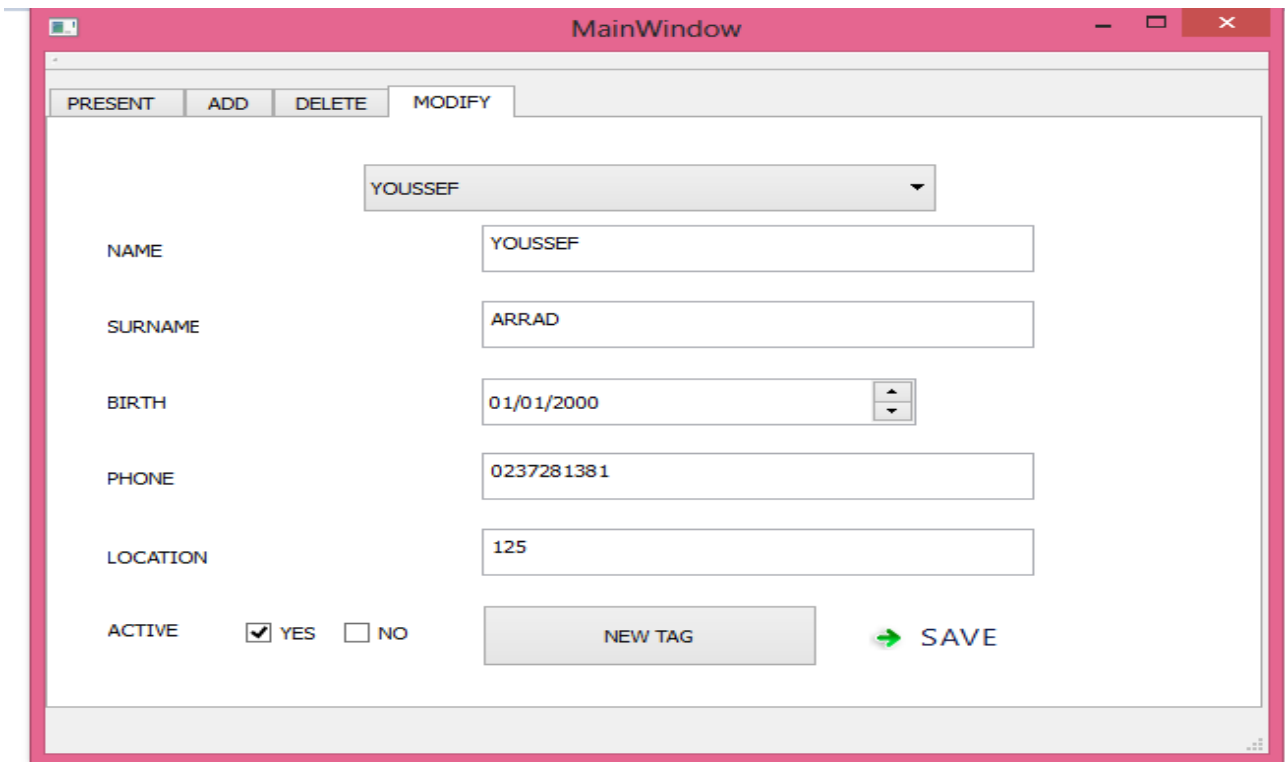
Profile
Are you sure you want to delete this profile ?
Yes No

Cet onglet est muni d'un **ComboBox** qui reprend les profils enregistrés dans la table **user**.

Voici la requête SQL permettant d'effacer un profil de la table **user** :

```
QSqlQuery req;  
QString cb = ui->comboBox->currentText();  
qDebug() << "Select="+cb;  
  
QString requete ="DELETE FROM user WHERE Name='"+cb+"'";  
bool A = req.exec(requete);  
qDebug() << "A=" << A;
```

Onglet MODIFY



On choisi le nom dans le **ComboBox** et il s'affiche les données civils du profile que l'on va pouvoir modifier.

Cependant il y a eu une anomalie au niveau de la requête **Update** .

Module client serveur

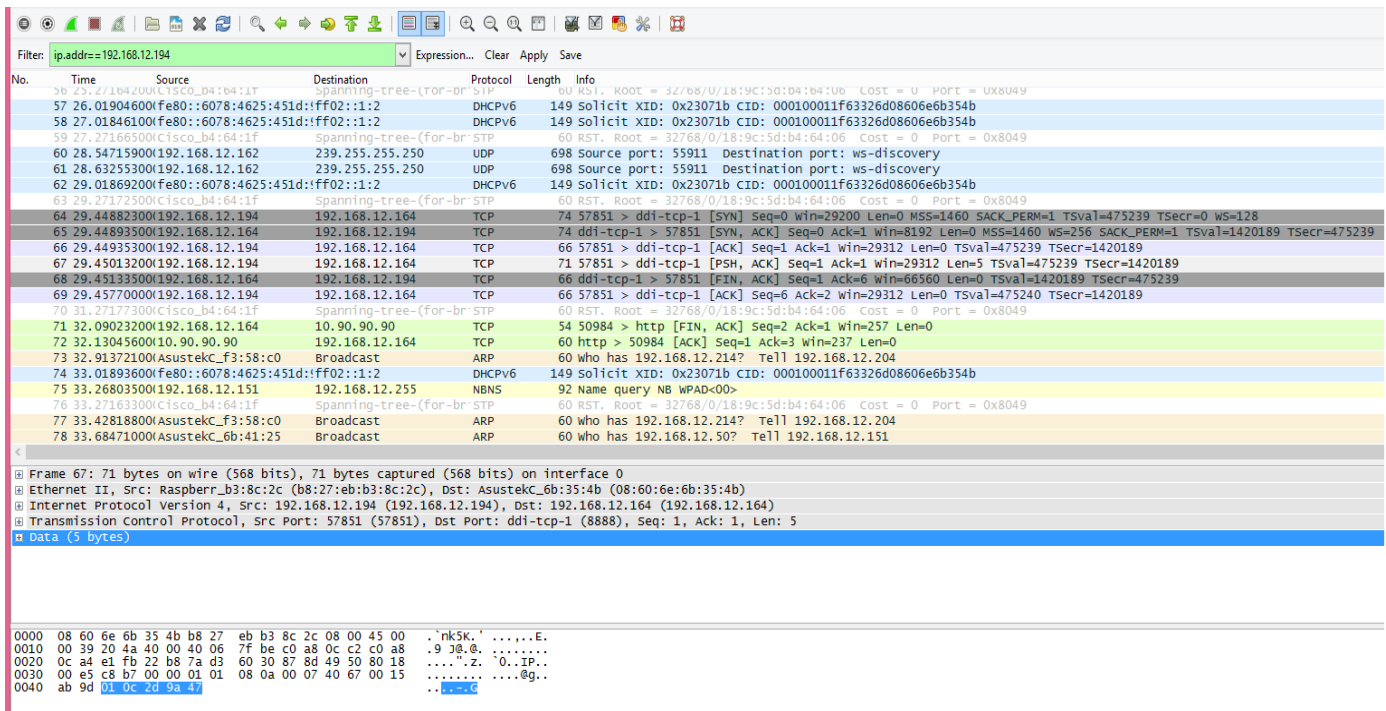
Il est possible d'établir une connexion entre le Client sur le Raspberry_pi et le Serveur qui communique avec la base de données.

Ainsi on peut envoyer des trames TCP/IP selon un protocole d'échange de données spécifique (comme ci-dessous).



Ceci est la trame « aller », le code du tag est envoyé sur 5 octets.

Voici la capture qui montre l'échange de données entre le client TCP et le serveur TCP on voit bien que cinq octets sont transmis.



Filter: ip.addr==192.168.12.194

No.	Time	Source	Destination	Protocol	Length	Info
57	26.01904600	fe80::6078:4625:451d::ff02::1:2	ff02::1:2	DHCPv6	149	Solicit XID: 0x23071b CID: 000100011f63326d0860e6b354b
58	27.01846100	fe80::6078:4625:451d::ff02::1:2	ff02::1:2	DHCPv6	149	Solicit XID: 0x23071b CID: 000100011f63326d0860e6b354b
59	27.27166500	cisco_b4:64:1f	Spanning-tree-(for-br) STP	60	RST, Root = 32768/0/18:9c:5d:b4:64:06 Cost = 0 Port = 0x8049	
60	28.54719000	192.168.12.162	239.255.255.250	UDP	698	Source port: 55911 Destination port: ws-discovery
61	28.63255300	192.168.12.162	239.255.255.250	UDP	698	Source port: 55911 Destination port: ws-discovery
62	29.01869200	fe80::6078:4625:451d::ff02::1:2	ff02::1:2	DHCPv6	149	Solicit XID: 0x23071b CID: 000100011f63326d0860e6b354b
63	29.27172500	cisco_b4:64:1f	Spanning-tree-(for-br) STP	60	RST, Root = 32768/0/18:9c:5d:b4:64:06 Cost = 0 Port = 0x8049	
64	29.44882300	192.168.12.194	192.168.12.164	TCP	74	57851 > ddi-tcp-1 [SYN] Seq=0 win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=475239 TSecr=0 WS=128
65	29.44893500	192.168.12.164	192.168.12.194	TCP	74	ddi-tcp-1 > 57851 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=1420189 TSecr=475239
66	29.44935300	192.168.12.194	192.168.12.164	TCP	66	57851 > ddi-tcp-1 [ACK] Seq=1 Ack=1 win=29312 Len=0 TSval=475239 TSecr=1420189
67	29.45013200	192.168.12.194	192.168.12.164	TCP	71	57851 > ddi-tcp-1 [PSH, ACK] Seq=1 Ack=1 win=29312 Len=5 TSval=475239 TSecr=1420189
68	29.45133300	192.168.12.164	192.168.12.194	TCP	66	ddi-tcp-1 > 57851 [FIN, ACK] Seq=1 Ack=6 win=66360 Len=0 TSval=1420189 TSecr=475239
69	29.45770000	192.168.12.194	192.168.12.164	TCP	66	57851 > ddi-tcp-1 [ACK] Seq=6 Ack=2 win=29312 Len=0 TSval=475240 TSecr=1420189
70	31.27177300	cisco_b4:64:1f	Spanning-tree-(for-br) STP	60	RST, Root = 32768/0/18:9c:5d:b4:64:06 Cost = 0 Port = 0x8049	
71	32.09023200	192.168.12.164	10.90.90.90	TCP	54	50984 > http [FIN, ACK] Seq=2 Ack=1 win=257 Len=0
72	32.13045600	10.90.90.90	192.168.12.164	TCP	60	http > 50984 [ACK] Seq=1 Ack=3 win=237 Len=0
73	32.91372100	AsustekC_f3:58:c0	broadcast	ARP	60	who has 192.168.12.214? Tell 192.168.12.204
74	33.01893600	fe80::6078:4625:451d::ff02::1:2	ff02::1:2	DHCPv6	149	Solicit XID: 0x23071b CID: 000100011f63326d0860e6b354b
75	33.26803500	192.168.12.151	192.168.12.255	NBNS	92	Name query NB WPAD=00
76	33.27163300	cisco_b4:64:1f	Spanning-tree-(for-br) STP	60	RST, Root = 32768/0/18:9c:5d:b4:64:06 Cost = 0 Port = 0x8049	
77	33.42818800	AsustekC_f3:58:c0	broadcast	ARP	60	who has 192.168.12.214? Tell 192.168.12.204
78	33.68471000	AsustekC_b6:41:25	broadcast	ARP	60	who has 192.168.12.50? Tell 192.168.12.151

Frame 67: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0

Ethernet II, Src: Raspberr_b3:8c:2c (b8:27:eb:b3:8c:2c), Dst: AsustekC_b6:35:4b (08:60:6e:6b:35:4b)

Internet Protocol version 4, Src: 192.168.12.194 (192.168.12.194), Dst: 192.168.12.164 (192.168.12.164)

Transmission Control Protocol, Src Port: 57851 (57851), Dst Port: ddi-tcp-1 (8888), Seq: 1, Ack: 1, Len: 5

Data (5 bytes)

```

0000  08 60 6e 6b 35 4b b8 27  eb b3 8c 2c 08 00 45 00  .`nk5K. ....E.
0010  00 10 20 4a 40 00 40 06  7f b6 c0 a8 02 c2 c0 a8  .9 38.8. ....
0020  0c a4 e1 f6 22 b8 7a d3  60 30 87 8d 45 50 80 18  .....Z. 0..IP.
0030  00 e5 c8 b7 00 00 01 01  08 0a 00 07 40 67 00 15  .....8g..
0040  ab 9d 01 0c 2d 9a 47      ..v..G
  
```

La trame réponse n'a cependant pas pu être réalisée du à des problèmes liés à l'interprétation des octets envoyés sur le buffer.

ANNEXES

Mainwindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QApplication>
#include <QSqlDatabase>
#include <QSqlDriver>
#include <QThread>

#include <QtSql/QtSql>
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlDriver>
#include <QtSql/QtSqlQuery>
#include "serveurtcp.h"
#include "mythread.h"

namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    //objet serveurtcp
    ServeurTcp *myserver;
    MyThread *mythread;
private slots:
    void on_comboBox_2_currentTextChanged(const QString &arg1);
    void on_pushButton_clicked();
    void on_valide_clicked();
    void on_pushButton_2_clicked();
    void on_commandLinkButton_clicked();
    void on_commandLinkButton_2_clicked();
    void testSlot(QString,QString,QString,int);
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H

```

Mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "serveurtcp.h"
#include "mythread.h"
#include "client.h"

#include <string>

#include "mainwindow.h"
#include <QApplication>
#include <QtSql>
#include <iostream>
#include <QTableWidgetItem>

#include "QTimer"
#include "QTime"
#include "QThread"
#include "QGraphicsView"
#include <QDebug>
#include <sstream>
#include <iostream>
#include <QComboBox>
#include <QCheckBox>
#include <QPushButton>
#include <QPalette>
#include <QColor>

#include <QtSql/QtSql>
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlDriver>
#include <QtSql/QtSqlQuery>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{

    ui->setupUi(this);

    //permet pour l'onglet MODIFY de mettre les attributs invisibles des la compilation
    ui->textEdit_5->setVisible(false);
    ui->textEdit_6->setVisible(false);
    ui->dateEdit_2->setVisible(false);
    ui->textEdit_8->setVisible(false);
    ui->textEdit_10->setVisible(false);
    ui->commandLinkButton_2->setVisible(false);
    ui->pushButton_2->setVisible(false);
    ui->label_5->setVisible(false);
    ui->label_6->setVisible(false);
    ui->label_7->setVisible(false);
    ui->label_8->setVisible(false);
    ui->label_10->setVisible(false);
    ui->label_11->setVisible(false);
    ui->checkBox->setVisible(false);
    ui->checkBox_2->setVisible(false);

    //connexion à la base de données

    QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName("localhost");
    db.setDatabaseName("database_seveso");
    db.setUserName("root");
    db.setPassword("");
    bool ok = db.open();
    if(ok) qDebug() << "you are now connected to Database SEVESO !!!" ;
    else qDebug() << "An error has occurred while trying to connect !!!" ;

```



```

//ajouter dans les combobox les noms saisis
QSqlQuery q;

bool R = q.exec("SELECT Name FROM user");
if(R == true){ qDebug() << " Select ok";//envoi le message d'erreur
    //int i = 1 ;
    while(q.next())
    {
        QString n = q.value(0).toString();
        ui->comboBox->addItem(n);
        ui->comboBox_2->addItem(n);
        //i++;
    }
}else qDebug() << "Error Select";//envoi le message d'erreur

/////////////////////////////////////////////////////////////////

//on instancie un objet serveur tcp
myserver = new ServeurTcp();
/////////////////////////////////////////////////////////////////

//on instancie un objet mythread
mythread = new MyThread();

connect(mythread, SIGNAL(testSignal(QString,QString,QString,int)), this, SLOT(testSlot(QString,QString,QString,int)));
qDebug ()<< "signal ok";

mythread->start();
/////////////////////////////////////////////////////////////////
}

void MainWindow::testSlot(QString a,QString b,QString c,int n){
qDebug ()<< "recep signal:";

/////////////////////////////////////////////////////////////////
//remplissage du qtablewidget

QTableWidgetItem *cubesHeaderItem = new QTableWidgetItem(a);
ui->tableWidget->setItem(n, 0, cubesHeaderItem);

cubesHeaderItem = new QTableWidgetItem(b);
ui->tableWidget->setItem(n, 1, cubesHeaderItem);

cubesHeaderItem = new QTableWidgetItem(c);
ui->tableWidget->setItem(n, 2, cubesHeaderItem);

/////////////////////////////////////////////////////////////////
}

MainWindow::~MainWindow()
{
    delete ui;
}

```

```

//onglet ADD

//bouton permettant de scanner le tag "ALLOCATE TAG"
void MainWindow::on_pushButton_clicked()
{
    QMessageBox* msgBox = new QMessageBox();
    msgBox->setWindowTitle("RFID_SEQUENCE");
    msgBox->setText("Please scan your Tag to end this !");
    msgBox->setIcon(QMessageBox::Information);
    msgBox->setWindowFlags(Qt::WindowStaysOnTopHint);
    msgBox->show();
    msgBox->setStandardButtons(0); //enleve le bouton "ok" par default
}

//Bouton valide

void MainWindow::on_valide_clicked()
{
    //on recupere la date
    QDate date = ui->dateEdit->date();
    //on recupere les données de type string
    QString N = ui->textEdit->toPlainText();
    QString S = ui->textEdit_2->toPlainText();
    QString P = ui->textEdit_4->toPlainText();
    QString l = ui->textEdit_9->toPlainText();
    bool a = ui->checkBox_3->isChecked();
    bool b = ui->checkBox_4->isChecked();
    int c = 1;
    int d = 0;

    //on recupere la valeur cochée dans la check box

    //si tout les champs sont remplis on enregistre sinon non
    if(date,N,S,P,l,a || b == true){

        QSqlQuery qry;
        //une fois le texte recuperé on insère les données dans la BDD
        qry.prepare("INSERT INTO user (Name,Surname,Birth,Phone,location,active) "
            "VALUES (:Name,:Surname,:Birth,:Phone,:location,:active)");
        qry.bindValue(":Name",N);
        qry.bindValue(":Surname",S);
        qry.bindValue(":Birth",date);
        qry.bindValue(":Phone",P);
        qry.bindValue(":location",l);
        if (a == true){
            qry.bindValue(":active",c);
            qDebug() << "a="+a;
        }else{
            qry.bindValue(":active",d);
            qDebug() << "b="+b;
        }
    }

    if( !qry.exec() )
        qDebug() << qry.lastError().text();//envoi le message d'erreur
    else{
        QMessageBox* msgBox = new QMessageBox();
        msgBox->setWindowTitle("New_member");
        msgBox->setText("you are registered !");
        msgBox->setWindowFlags(Qt::WindowStaysOnTopHint);
        msgBox->show();
    }
    qDebug() << "passage insert";
}

```

```

//onglet DELETE

//liste deroulante permettant de supprimer un profile
void MainWindow::on_commandLinkButton_clicked()
{
    if(ui->comboBox->currentText() != "Select Name")
    {
        QMessageBox msgBox;

        msgBox.setWindowTitle("Profile");
        msgBox.setText("Are you sure you want to delete this profile ?");
        msgBox.setIcon(QMessageBox::Question);
        msgBox.show();
        msgBox.setStandardButtons(0); //enleve le bouton "ok" par default
        msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No );

        //Quand on clique sur le bouton yes
        int del = msgBox.exec();

        //Quand on clique sur le bouton yes
        int del = msgBox.exec();
        switch(del)
        {
            case QMessageBox::Yes:
            {
                QSqlQuery req;
                QString cb = ui->comboBox->currentText();
                qDebug() << "Select="+cb;

                QString requete ="DELETE FROM user WHERE Name='"+cb+"'";
                bool A = req.exec(requete);
                qDebug() << "A=" << A;

                //Boite de dialogue qui confirme la suppression du profile
                QMessageBox* msBox = new QMessageBox();
                msBox->setWindowTitle("Profile");
                msBox->setText("The PROFILE has been erased !");
                msBox->show();

                ui->comboBox->removeItem(1);
                ui->comboBox_2->removeItem(1);
                break;
            }
            case QMessageBox::No:

                QMessageBox* m = new QMessageBox();
                m->setWindowTitle("Profile");
                m->setText("Profile Saved !");
                m->setWindowFlags(Qt::WindowStaysOnTopHint);
                m->show();
                break;
        }
    }
}

```

```
//onglet MODIFY

void MainWindow::on_comboBox_2_currentTextChanged(const QString &arg1)
{
    if(ui->comboBox_2->currentText() == "Choose a name")
    {
        ui->textEdit_5->setVisible(false);
        ui->textEdit_6->setVisible(false);
        ui->dateEdit_2->setVisible(false);
        ui->textEdit_8->setVisible(false);
        ui->textEdit_10->setVisible(false);
        ui->commandLinkButton_2->setVisible(false);
        ui->pushButton_2->setVisible(false);
        ui->label_5->setVisible(false);
        ui->label_6->setVisible(false);
        ui->label_7->setVisible(false);
        ui->label_8->setVisible(false);
        ui->label_10->setVisible(false);
        ui->label_11->setVisible(false);
        ui->checkBox->setVisible(false);
        ui->checkBox_2->setVisible(false);
    }else{
        ui->textEdit_5->setVisible(true);
        ui->textEdit_6->setVisible(true);
        ui->dateEdit_2->setVisible(true);
        ui->textEdit_8->setVisible(true);
        ui->textEdit_10->setVisible(true);

        ui->commandLinkButton_2->setVisible(true);
        ui->pushButton_2->setVisible(true);
        ui->label_5->setVisible(true);
        ui->label_6->setVisible(true);
        ui->label_7->setVisible(true);
        ui->label_8->setVisible(true);
        ui->label_10->setVisible(true);
        ui->label_11->setVisible(true);
        ui->checkBox->setVisible(true);
        ui->checkBox_2->setVisible(true);

    }

}

//bouton clic "NEW TAG"
void MainWindow::on_pushButton_2_clicked()
{
    QMessageBox* msgBox = new QMessageBox();
    msgBox->setWindowTitle("RFID_SEQUENCE");
    msgBox->setText("Please scan your Tag to end this !");
    msgBox->setWindowFlags(Qt::WindowStaysOnTopHint);
    msgBox->show();
    msgBox->setStandardButtons(0); //enleve le bouton "ok" par default
}

//bouton de sauvegarde "save"
void MainWindow::on_commandLinkButton_2_clicked()
{
    //requete qui supprime le profile avant d'insérer les modifs
    QString cb = ui->comboBox_2->currentText();
    QSqlQuery dl;
    QString r ="DELETE FROM user WHERE Name='"+cb+"'";
    bool A = dl.exec(r);
    qDebug() << "dl="+r;

    //on recupere la date
    QDate date = ui->dateEdit_2->date();
    //on recupere les données de type string
    QString N = ui->textEdit_5->toPlainText();
    QString S = ui->textEdit_6->toPlainText();
    QString P = ui->textEdit_8->toPlainText();
    QString l = ui->textEdit_10->toPlainText();
    bool a = ui->checkBox->isChecked();
    bool b = ui->checkBox_2->isChecked();
    int c = 1;
    int d = 0;

    qDebug() << "a="<a;
    //on recupere la valeur cochée dans le check box

```

```

if(ui->comboBox_2->currentText() != "Choose a name"){
    QSqlQuery qy;
    QString z = ui->comboBox_2->currentText();
    qDebug() << "Select="+z;

    //req permettant l'affichage des données civil apres avoir selectionné le nom

    QString qey ="SELECT * FROM user WHERE Name='"+z+"'";
    bool A = qy.exec(qey);
    qDebug() << "A="<<qey;

    while(qy.next())
    {
        ui->textEdit_5->setText(qy.value(1).toString());
        ui->textEdit_6->setText(qy.value(2).toString());
        ui->dateEdit_2->setDate(qy.value(3).toDate());
        ui->textEdit_8->setText(qy.value(4).toString());
        ui->textEdit_10->setText(qy.value(6).toString());
    }

    //requete qui va gérer le coche des checkboxes
    QSqlQuery ch;
    QString check ="SELECT * FROM user WHERE active =1";
    bool B = ch.exec(check);
    qDebug() << "B="<<check;

    while(ch.next()){
        if (B == true){
            ui->checkBox->setChecked(true);
        }else if(B == false){
            ui->checkBox_2->setChecked(true);
        }
    }

}

//si tout les champs sont remplis on enregistre sinon non
QSqlQuery qry;

qry.prepare("INSERT INTO user (Name,Surname,Birth,Phone,location,active) "
            "VALUES (:Name, :Surname, :Birth, :Phone, :location, :active)");
qry.bindValue(":Name",N);
qry.bindValue(":Surname",S);
qry.bindValue(":Birth",date);
qry.bindValue(":Phone",P);
qry.bindValue(":location",l);
if (a == true){
    qry.bindValue(":active",c);
    qDebug() << "c="<<c;
}else{
    qry.bindValue(":active",d);
    qDebug() << "d="<<d;
}

if( !qry.exec() )
    qDebug() << qry.lastError().text();//envoi le message d'erreur
else{
    QMessageBox* msgBox = new QMessageBox();
    msgBox->setWindowTitle("New member");
    msgBox->setText("The modifications have been taken into account!");
    msgBox->setWindowFlags(Qt::WindowStaysOnTopHint);
    msgBox->show();
    qDebug() << "insert ok";
}
}

```

ServeurTcp.h

```
#ifndef SERVEURTCP_H
#define SERVEURTCP_H
#include <QtNetwork>
#include <QObject>
#include <QTcpServer>
#include <QTcpSocket>

class ServeurTcp : public QObject
{
    Q_OBJECT
public:

    ServeurTcp(QObject * parent = 0);
    ~ServeurTcp();

public slots :

    void acceptConnection();
    void startRead();

private:

    QTcpServer server;
    QTcpSocket* client;

    //requete qui va comparer le tag
    int requete(QString tagrfid);

signals:
    void vers_client(QString);
};
#endif // SERVEURTCP_H
```

ServeurTcp.cpp

```
#include "serveurtcp.h"
#include <iostream>
#include <QByteArray>
#include <QtSql/QtSql>
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlDriver>
#include <QtSql/QtSqlQuery>

using namespace std;

//constructeur
ServeurTcp::ServeurTcp(QObject* parent): QObject(parent)
{
    connect(&server, SIGNAL(newConnection()),
           this, SLOT(acceptConnection()));

    if(!server.listen(QHostAddress::Any, 8888)){
        qDebug() << " Erreur Connexion " ;
    } else
        qDebug() << " Serveur ouvert sur le port 8888 : " ;
}

//destructeur
ServeurTcp::~ServeurTcp()
{
    server.close();
}

void ServeurTcp::acceptConnection()
{
    client = server.nextPendingConnection();

    connect(client, SIGNAL(readyRead()),
           this, SLOT(startRead()));
    qDebug() << " connexion ok" ;
}
```



```

void ServeurTcp::startRead()
{
    //on recoit 5 octet or long long int fait 8 octets donc il faut au moins 8 octets
    char buffer[]={0,0,0,0,0,0,0,0,0};

    //lire les données du buffer
    int b = client->read(buffer, 20);
    qDebug() <<"Recu: " << buffer << "b="<<b<< endl;
    for(int n=0;n<b;n++){
        qDebug() <<" " << (int)buffer[n];
    }
    //interpretation en entier des données reçus dans le buffer
    long long int* l = (long long int *)buffer;
    qDebug() << "l= " << *l << endl;

    cout << endl << "taille= " << l << endl;

    //appel de la fonction requete pour comparer le code rfid

    QSqlQuery comparer;
    QString dd = "";
    dd.setNum(*l);

    QString req = "SELECT tag_rfid FROM user WHERE tag_rfid = '"+dd+"'";
    bool a = comparer.exec(req);
    qDebug() << " requete " << req << endl;

    client->close();
}

int ServeurTcp::requete(QString tagrfid){
}

```

MyThread.h

```
#ifndef MYTHREAD_H
#define MYTHREAD_H

#include <QThread>
#include <QtSql/QSql>
#include <QtSql/QSqlDatabase>
#include <QtSql/QSqlDriver>
#include <QtSql/QSqlQuery>

class MyThread : public QThread
{
    Q_OBJECT

public:
    MyThread(QObject *parent = 0);
signals:
    void testSignal(QString,QString,QString,int);
protected:
    void run();
public slots:
    int execC(QString);
private:
    QThread thread;
};
#endif // MYTHREAD_H
```

MyThread.cpp

```
#include "mainwindow.h"
#include "mythread.h"
#include <QThread>
#include <QtSql/QtSql>
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlDriver>
#include <QtSql/QtSqlQuery>

using namespace std;
void MyThread::run()
{
    while(true)
    {
        //requete permettant de recuperer les personnes presentes et de les rafraechir l'ihm
        QSqlQuery a;
        int n = 0;
        bool R = a.exec("SELECT Name,Surname,Time FROM presence");
        if (R == true){

            //tant qu'il y aura une autre personne presente
            while(a.next()){

                //on emet le signal
                emit(testSignal(a.value(0).toString(),a.value(1).toString(),a.value(2).toString(), n));
                n++;
            }
            sleep(3);
        }
    }
}

MyThread::MyThread(QObject* parent): QThread(parent)
{

}

int MyThread::execC(QString V)
{
}
```

Main.cpp

```
#include "mainwindow.h"
#include "serveurtcp.h"
#include <QApplication>
#include <QtSql>
#include <iostream>

#include "QTimer"
#include "QTime"
#include <QDebug>
#include <sstream>
#include <iostream>

#include <QtSql/QtSql>
#include <QtSql/QtSqlDatabase>
#include <QtSql/QtSqlDriver>
#include <QtSql/QtSqlQuery>

int main(int argc, char *argv[])
{

    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();

}
```

Tâche étudiant n°2

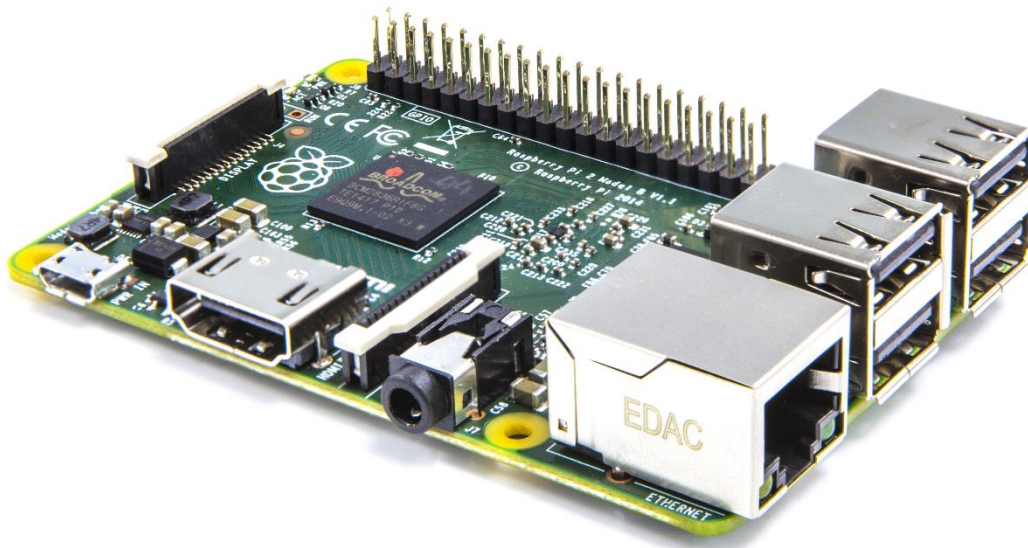
Ahmed Kassim Naïl

I- Présentation du matériel

Afin de mettre en place des portiques qui permettront l'accès des individus dans le bâtiment nous avons à disposition : un Raspberry pi , un écran LCD , un lecteur velleman k8010 ainsi que des cartes RFID

Raspberry pi : Le Raspberry Pi est un nano-ordinateur mono carte à processeur ARM conçu par le créateur de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi2.

Cet ordinateur, qui a la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique ; il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles. Il est fourni nu (carte mère seule, sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.



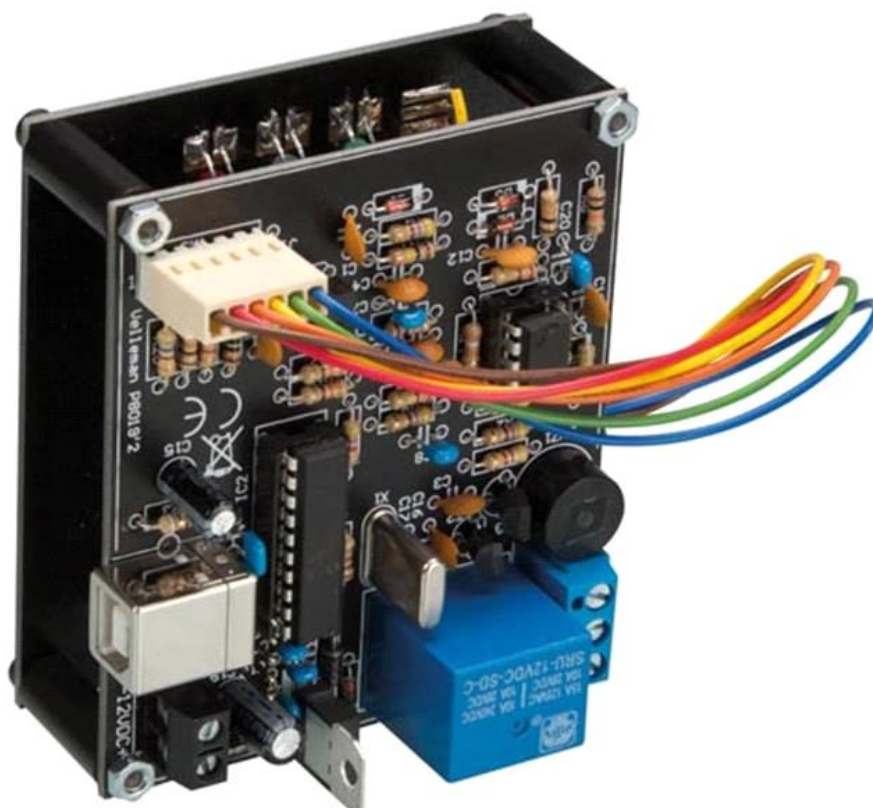
Dans notre projet , l'application s'exécute dans le Raspberry.

Ecran LCD : Un écran LCD est directement lié au Raspberry pi



Lecteur Velleman k8010 :

Le lecteur utilise la technologie RFID (Radio Fréquence Identification)



Il permet de récupérer un tag lorsque l'on scanne une carte dessus.

II- Mise en place du matériel

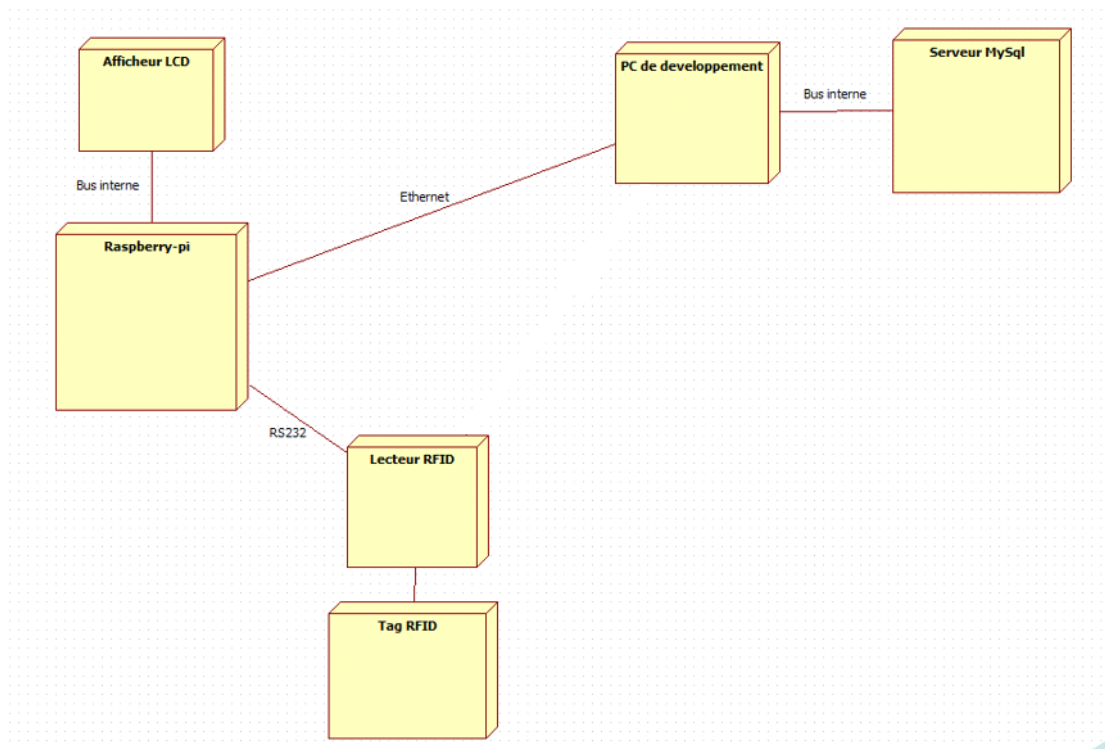
a-dispositifs

L'ensemble de l'application est intégré dans le Raspberry. Le lecteur velleman k8019 est relié par

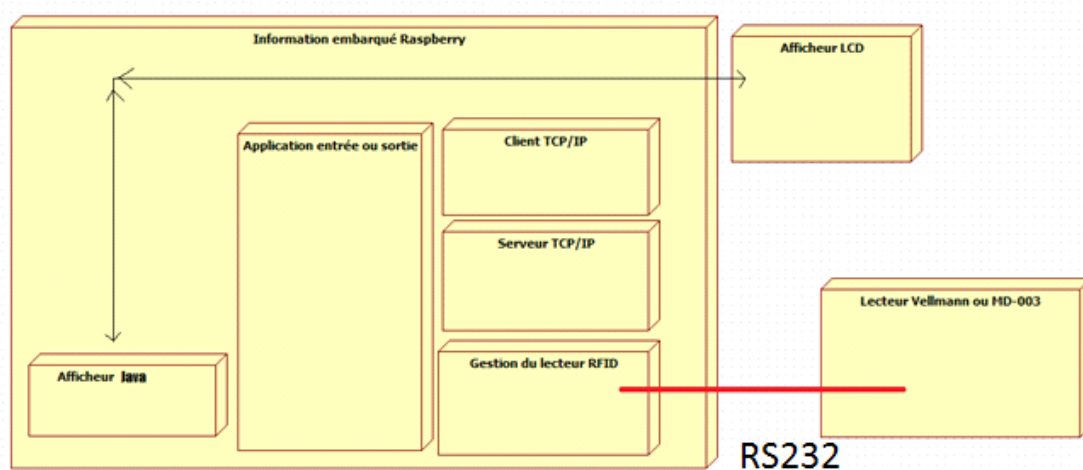
liaison RS232.

l'écran LCD est directement lié au Raspberry par un bus interne.

Le Raspberry est lui branché au poste de supervision par un câble Ethernet.



b- Diagramme de déploiement personnel



L'ensemble de l'application est intégré dans le Raspberry. Le lecteur velleman k8019 est relié par liaison RS232.

III- Contexte

a-Objectifs

Le but de ce projet est le contrôle d'accès et le comptage du personnel dans un bâtiment.

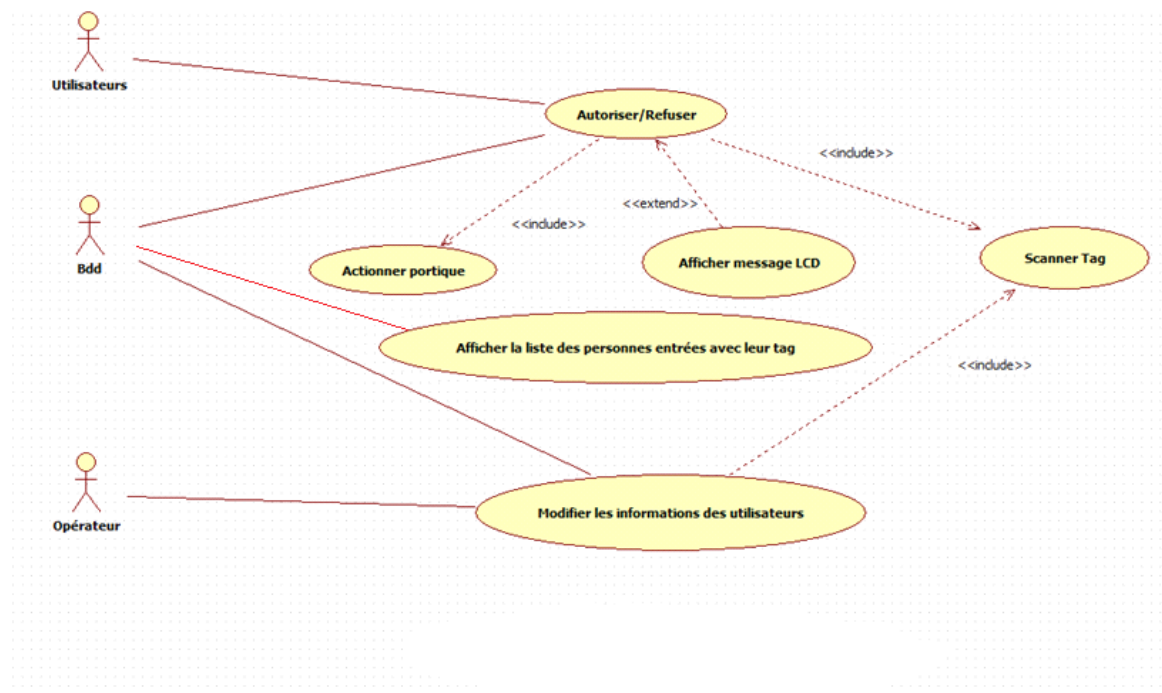
Pour ce faire chaque individu se voit attribuer un badge contenant un tag unique. Lorsque scanne son badge dans le lecteur l'entrée est alors autorisée ou refusée.

les données de chacun ainsi que le tag attribué est stocké dans une base de données.

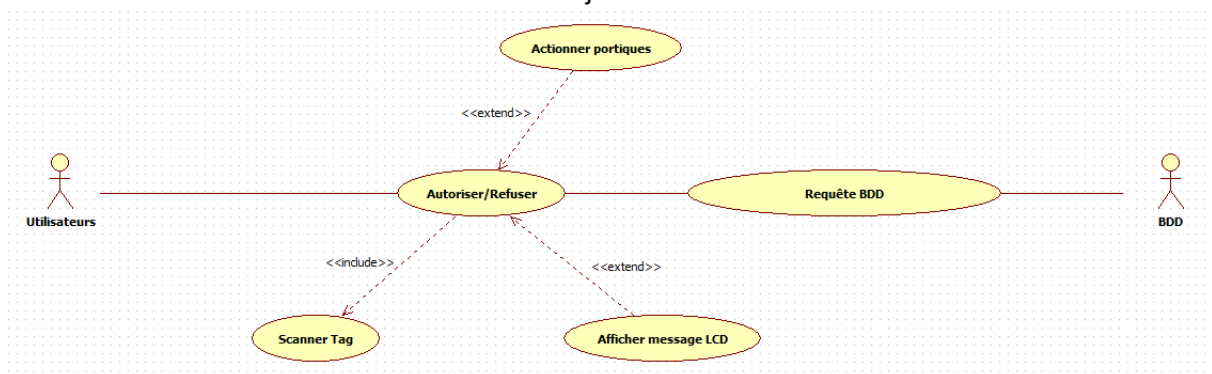
Quand une carte est passée sur le lecteur on envoie le tag au poste de supervision qui va comparer avec ceux stockés dans la base de données et va nous renvoyer une réponse. Ouverture ou fermeture.

Dans un premier temps il a fallu créer un module qui permet la récupération du tag ensuite, un module client et serveur permettant la communication et un module commandant le lecteur LCD

b-Diagramme des cas d'utilisations général et personnel



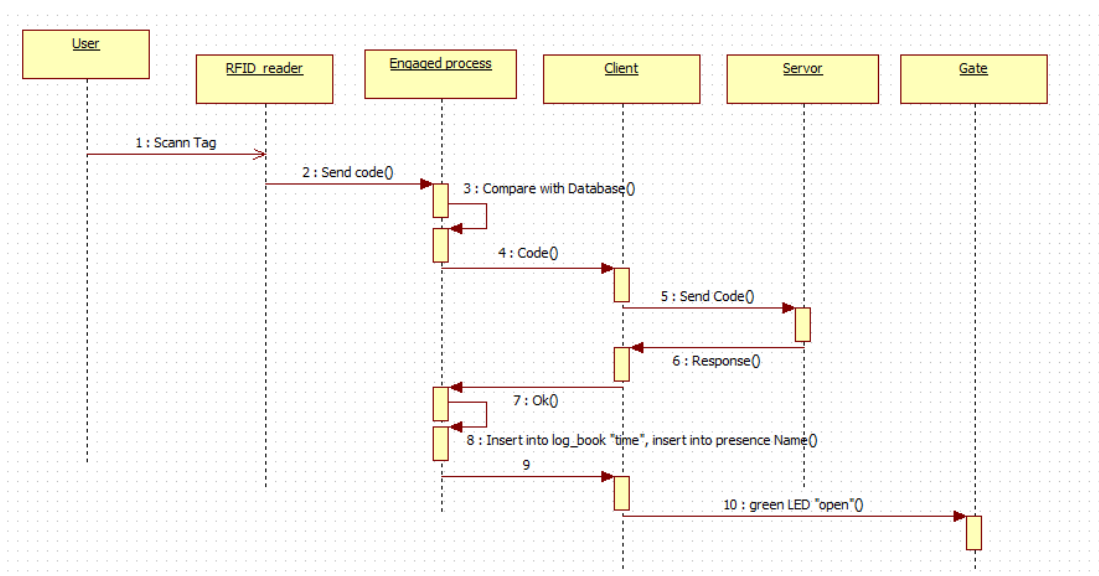
Projet SNIR



On peut déterminer 3 acteurs : utilisateurs (celui qui va badger) , la base de données (car elle permet l'actionner du portique et l'opérateur qui a travers le poste de supervision va pouvoir ajouter, supprimer et modifier des données.

Dans ma partie je n'interviens qu'au niveau de l'utilisateur.

c- Diagramme de séquence



Les évènements se déroulent de la sorte ; l'utilisateur scanne le tag , le lecteur lis et envoi le tag au poste de supervision , une comparaison est faite avec la base de données , qui va ensuite renvoyé une réponse si nous pouvons ou non ouvrir le portique.

III - Modules

a- Protocole d'échange entre les sous réseaux

Afin de communiquer entre les sous réseaux nous avons décidé d'utiliser le protocole TCP.

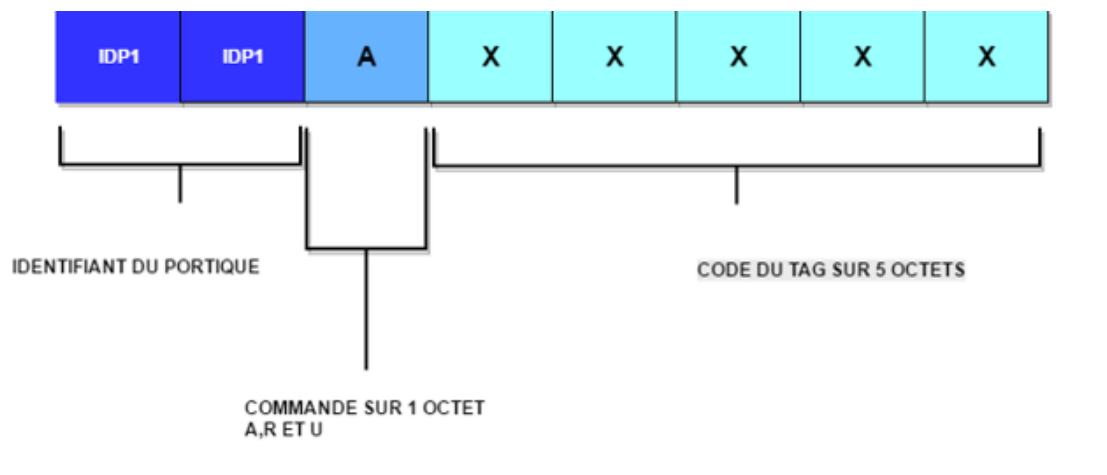
Grâce au protocole TCP, les applications peuvent communiquer de façon sûre (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est

réalisé par la couche transport (plus particulièrement par le protocole TCP).

Lors d'une communication à travers le protocole TCP, les deux machines doivent établir une connexion. La machine émettrice (celle qui demande la connexion) est appelée client, tandis que la machine réceptrice est appelée serveur. On dit qu'on est alors dans un environnement Client-Serveur. Les machines dans un tel environnement communiquent en mode connecté, c'est-à-dire que la communication se fait dans les deux sens.

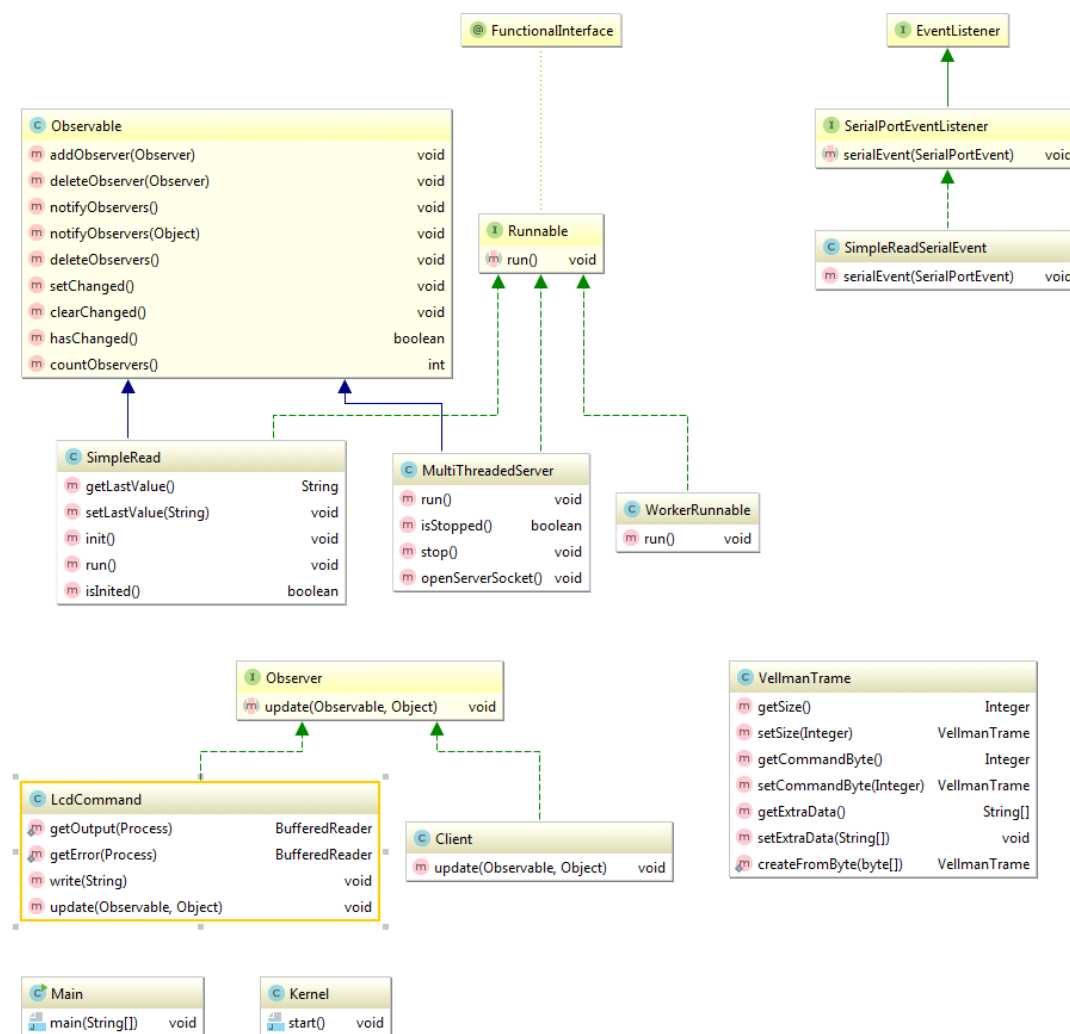
Pour permettre le bon déroulement de la communication et de tous les contrôles qui l'accompagnent, les données sont encapsulées, c'est-à-dire qu'on ajoute aux paquets de données un en-tête qui va permettre de synchroniser les transmissions et d'assurer leur réception.

Format de la Trame envoyé :



La trame est composée de 9 octets . Les deux premiers représente l'identifiant du portique , le 3e indique une commande (A = Autoriser , R= Refuser , U = Urgence) , les cinq derniers octets représentent l'identifiant du tag.

b- Diagramme des classes



C - Annexe

```
File - C:\Users\admin\IdeaProjects\Nall\src\Main.java
1 import core.Kernel;
2
3 /**
4  * Point d'entrée principal du programme
5  */
6 public class Main {
7
8
9
10     //methode d'entrée
11     public static void main(String[] args) {
12         //initialisation du noyau d'execution
13         Kernel kernel = new Kernel();
14         //démarrage du noyau
15         kernel.start();
16     }
17 }
18
19
```

File - C:\Users\admin\IdeaProjects\Nail\src\lcd\LcdCommand.java

```

1 package lcd;
2
3 import java.io.BufferedReader;
4 import java.io.InputStreamReader;
5 import java.util.Observable;
6 import java.util.Observer;
7
8 /**
9  *
10 */
11 public class LcdCommand implements Observer {
12
13     private String pythonCommandName;
14
15     public LcdCommand(String pythonCommandName){
16         this.pythonCommandName = pythonCommandName;
17     }
18
19
20
21     private static BufferedReader getOutput(Process p) {
22         return new BufferedReader(new InputStreamReader(p.getInputStream()));
23     }
24
25     private static BufferedReader getError(Process p) {
26         return new BufferedReader(new InputStreamReader(p.getErrorStream()));
27     }
28
29
30     /**
31      * Méthode permettant d'afficher un texte dans le LCD
32      * @param text
33      */
34     private void write(String text){
35
36         String s;
37         Process p;
38
39         String[] command = {"python", "mytest.py", text};
40         try {
41             //On exécute la commande encapsulée dans mytest.py
42             p = Runtime.getRuntime().exec(command);
43
44             //Gestion des retours
45             BufferedReader output = getOutput(p);
46             BufferedReader error = getError(p);
47
48             String ligne = "";
49
50             while ((ligne = output.readLine()) != null) {
51                 System.out.println(ligne);
52             }
53
54             while ((ligne = error.readLine()) != null) {
55                 System.out.println(ligne);
56             }
57
58             p.waitFor();
59             p.destroy();
60         } catch (Exception e) {

```

Page 1 of 2

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\lcd\LcdCommand.java

```
61         System.out.println(e.getMessage());
62     }
63 }
64
65
66 @Override
67 public void update(Observable o, Object arg) {
68     String msg = (String) arg;
69     this.write(msg);
70 }
71 }
72
```

```

File - C:\Users\admin\IdeaProjects\Nail\src\core\Kernel.java
1 package core;
2
3 import lcd.LcdCommand;
4 import networking.Client;
5 import networking.MultiThreadedServer;
6 import purejavacomm.CommPortIdentifier;
7 import rs232.SimpleRead;
8
9 import java.net.InetAddress;
10 import java.net.UnknownHostException;
11 import java.util.Enumeration;
12
13 /**
14  *
15  */
16 public class Kernel {
17
18     public void start() {
19         CommPortIdentifier portId;
20         Enumeration portList;
21
22
23         MultiThreadedServer server;
24
25
26         //On initialise le client
27         Client client = null;
28         try {
29             client = new Client(25000, InetAddress.getLocalHost());
30         } catch (UnknownHostException e) {
31             e.printStackTrace();
32         }
33         //On initialise le client LCD
34         LcdCommand lcd = new LcdCommand("~/mytest.py");
35
36
37         //On initialise le serveur
38         server = new MultiThreadedServer(26889);
39         //On rajoute le LCD aux observateurs du serveur, Le LCD de son côté
40         //affichera tout retour depuis le serveur
41         server.addObserver(lcd);
42         new Thread(server).start();
43
44         System.out.println("Recherche de ports... ");
45         //On récupère la liste des ports
46         portList = CommPortIdentifier.getPortIdentifiers();
47
48         //On parcourt la liste des ports avec un while
49         while (portList.hasMoreElements()) {
50
51             //on récupère le port identifier
52             portId = (CommPortIdentifier) portList.nextElement();
53
54             //On ne garde que les ports Serial
55             if ((portId.getPortType() == CommPortIdentifier.PORT_SERIAL) /*%%
56                 portId.getName().equals("ttyACM0")*/) {
57
58                 SimpleRead reader;

```


Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\core\Kernel.java

```
59
60         try {
61
62             //On initialise un reader sur le port courant
63             reader = new SimpleRead(portId);
64
65             if (reader.isInitiated()) {
66                 //Nous inscrivons le client en observer du reader
67                 reader.addObserver(client);
68
69                 //On démarre un thread pour le reader
70                 Thread readThread = new Thread(reader);
71                 readThread.start();
72
73             }
74
75         } catch (Exception e) {
76             System.out.println(e.getClass());
77         }
78     }
79 }
80 }
81 }
82 }
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\mock\Client.java

```
1 package mock;
2
3 import java.io.BufferedWriter;
4 import java.io.IOException;
5 import java.io.OutputStream;
6 import java.io.OutputStreamWriter;
7 import java.net.InetAddress;
8 import java.net.Socket;
9 import java.net.UnknownHostException;
10 import java.util.Observable;
11 import java.util.Observer;
12
13
14 /**
15  * Created by admin on 21/01/2017.
16  */
17 public class Client {
18
19
20     private final Integer port;
21     private final InetAddress address;
22
23     public Client(Integer port, InetAddress address) {
24         this.port = port;
25         this.address = address;
26     }
27
28
29
30     public void open() {
31         Socket socket;
32         OutputStream out;
33         OutputStreamWriter osw;
34         BufferedWriter bw;
35         try {
36
37             socket = new Socket(this.address, this.port);
38             out = socket.getOutputStream();
39             osw = new OutputStreamWriter(out);
40             bw = new BufferedWriter(osw);
41             bw.write("Ouverture");
42             bw.flush();
43             socket.close();
44
45         } catch (UnknownHostException e) {
46
47             e.printStackTrace();
48         } catch (IOException e) {
49
50             e.printStackTrace();
51         }
52     }
53
54     public void close(){
55         Socket socket;
56         OutputStream out;
57         OutputStreamWriter osw;
58         BufferedWriter bw;
59         try {
60
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\mock\Client.java

```
61         socket = new Socket(this.address, this.port);
62         out = socket.getOutputStream();
63         osw = new OutputStreamWriter(out);
64         bw = new BufferedWriter(osw);
65         bw.write("Fermeture");
66         bw.flush();
67         socket.close();
68
69     } catch (UnknownHostException e) {
70
71         e.printStackTrace();
72     } catch (IOException e) {
73
74         e.printStackTrace();
75     }
76 }
77
78
79 }
80
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\mock\Server.java

```
1 package mock;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
7 import java.net.InetAddress;
8 import java.net.ServerSocket;
9 import java.net.Socket;
10
11 public class Server {
12
13     public static void main(String[] zero) {
14
15         ServerSocket socketserver;
16         Socket socket;
17         BufferedReader in;
18
19         try {
20
21             socketserver = new ServerSocket(25000);
22
23             while (true) {
24                 //Reading the message from the client
25                 socket = socketserver.accept();
26                 InputStream is = socket.getInputStream();
27                 InputStreamReader isr = new InputStreamReader(is);
28                 BufferedReader br = new BufferedReader(isr);
29
30                 String message = br.readLine();
31
32                 Client client = new Client(26889, InetAddress.getLocalHost());
33                 client.open();
34                 System.out.println("Message received from client is " + message
35 );
36
37             }
38
39             //
40             //         socketserver.close();
41             //         socketduserveur.close();
42             } catch (IOException e) {
43                 e.printStackTrace();
44             }
45         }
46
47     }
48 }
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\rs232\SimpleRead.java

```

1  package rs232;
2
3  import purejavacomm.*;
4
5  import java.util.*; //import gnu.io.*;
6
7  public class SimpleRead extends Observable implements Runnable {
8
9      private boolean initied;
10     private CommPortIdentifier portId;
11     SerialPort serialPort;
12     String lastValue;
13
14
15     /**
16      * @return
17      */
18     public String getLastValue() {
19         return lastValue;
20     }
21
22     /**
23      * @param lastValue
24      */
25     public void setLastValue(String lastValue) {
26         this.lastValue = lastValue;
27         setChanged();
28         notifyObservers(lastValue);
29     }
30
31     /**
32      *
33      * @param portId
34      */
35     public SimpleRead(CommPortIdentifier portId) {
36         super();
37         this.portId = portId;
38         this.initied = false;
39         this.init();
40     }
41
42     /**
43      * On initialise le reader
44      */
45     private void init() {
46         try {
47
48             //On initialise un portCom
49             serialPort = (SerialPort) portId.open("SimpleReadAppl111", 500);
50             //On lui associe un Event listener qui est un SimpleReadSerialEvent
51             serialPort.addEventListener(new SimpleReadSerialEvent(serialPort,
52 this));
53
54             //On demande à être notifié qu'en cas d'arrivée de données
55             serialPort.notifyOnDataAvailable(true);
56             //On initialise des paramètres de base
57             serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8,
58 SerialPort.STOPBITS_1, SerialPort.PARITY_EVEN);
59
60             // no handshaking or other flow control
61             serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);

```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\rs232\SimpleRead.java

```
60
61         // On met un timer de 500 millisecond
62         serialPort.enableReceiveTimeout(500);
63
64         this.inited = true;
65
66         } catch (PortInUseException e) {
67             System.out.println("Port in use Exception");
68         } catch (TooManyListenersException e) {
69             System.out.println("Too many Listener exception");
70         } catch (UnsupportedCommOperationException e) {
71             System.out.println("UnSupported comm operation");
72         } catch (Exception e) {
73             System.out.println(e.getMessage());
74         }
75     }
76 }
77
78 public void run() {
79
80     try {
81         while (true) {
82             Thread.sleep(500);
83         }
84         // System.out.println();
85     } catch (InterruptedException e) {
86         System.out.println("Interrupted Exception in run() method");
87     }
88 }
89
90
91 public boolean isInited() {
92     return inited;
93 }
94 }
95
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\rs232\SimpleReadSerialEvent.java

```
1 package rs232;
2
3 import purejavacomm.SerialPort;
4 import purejavacomm.SerialPortEvent;
5 import purejavacomm.SerialPortEventListener;
6 import vellman.VellmanTrame;
7
8 import java.io.IOException;
9 import java.io.InputStream;
10 import java.util.Arrays;
11
12 /**
13  *
14  */
15 public class SimpleReadSerialEvent implements SerialPortEventListener {
16
17     private final SerialPort serialPort;
18     private final SimpleRead simpleRead;
19
20
21     private InputStream inputStream;
22
23
24     public SimpleReadSerialEvent(SerialPort serialPort, SimpleRead simpleRead)
25     {
26         this.serialPort = serialPort;
27         this.simpleRead = simpleRead;
28         try {
29             inputStream = serialPort.getInputStream();
30             System.out.println(" Input Stream... " + inputStream);
31         } catch (IOException e) {
32             System.out.println("IO Exception");
33         }
34     }
35
36     /**
37      * Methode qui est appelée lorsqu'un événement sur le port sérial se
38      * produit
39      * @param event
40      */
41     public void serialEvent(SerialPortEvent event) {
42
43         //On vérifie le type d'événement
44         switch (event.getEventType()) {
45
46             //Lorsque il y'a des données
47             case SerialPortEvent.DATA_AVAILABLE:
48
49                 //ON récupère une trame de 9 bytes
50                 byte[] readBuffer = new byte[9];
51
52                 try {
53
54                     while (inputStream.available() > 0) {
55
56                         int numBytes = inputStream.read(readBuffer);
57                     }
58
59                     //On crée une trame vellman
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\rs232\SimpleReadSerialEvent.java

```
59         VellmanTrame velTr = VellmanTrame.createFromByte(  
        readBuffer);  
60  
61         //On informe le reader  
62         this.simpleRead.setLastValue(Arrays.toString(velTr.  
        getExtraData()));  
63  
64  
65         } catch (IOException e) {  
66             System.out.println("IO Exception in SerialEvent()");  
67         }  
68         break;  
69     }  
70  
71 }  
72  
73  
74 }  
75
```


Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\vellman\VellmanTrame.java

```
1 package vellman;
2
3 import java.util.*;
4 import java.lang.System.*;
5
6 import static java.lang.System.arraycopy;
7
8 /**
9  * Created by admin on 11/03/2017.
10 */
11 public class VellmanTrame {
12
13     private Integer size;
14     private Integer commandByte;
15     private String[] extraData;
16
17     public Integer getSize() {
18         return size;
19     }
20
21     public VellmanTrame setSize(Integer size) {
22         this.size = size;
23         return this;
24     }
25
26     public Integer getCommandByte() {
27         return commandByte;
28     }
29
30     public VellmanTrame setCommandByte(Integer commandByte) {
31         this.commandByte = commandByte;
32         return this;
33     }
34
35     public String[] getExtraData() {
36         return extraData;
37     }
38
39     public void setExtraData(String[] extraData) {
40         this.extraData = extraData;
41     }
42
43     public static VellmanTrame createFromByte(byte[] trameF){
44
45         List<String> parts = new ArrayList<String>();
46         for (byte aTrameF : trameF) {
47             parts.add(Integer.toString((aTrameF & 0xff) + 0x100, 16).substring(1
48 ));
49         }
50         VellmanTrame vel = new VellmanTrame();
51         vel.setSize(Integer.parseInt(parts.get(1)));
52         vel.setCommandByte(Integer.parseInt(parts.get(2)));
53         String[] datas = new String[5];
54         vel.setExtraData(parts.subList(4, 9).toArray(datas));
55
56         return vel;
57     }
58 }
59
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\vellman\VellmanTrame.java

```

1 package vellman;
2
3 import java.util.*;
4 import java.lang.System.*;
5
6 import static java.lang.System.arraycopy;
7
8 /**
9  * Created by admin on 11/03/2017.
10 */
11 public class VellmanTrame {
12
13     private Integer size;
14     private Integer commandByte;
15     private String[] extraData;
16
17     public Integer getSize() {
18         return size;
19     }
20
21     public VellmanTrame setSize(Integer size) {
22         this.size = size;
23         return this;
24     }
25
26     public Integer getCommandByte() {
27         return commandByte;
28     }
29
30     public VellmanTrame setCommandByte(Integer commandByte) {
31         this.commandByte = commandByte;
32         return this;
33     }
34
35     public String[] getExtraData() {
36         return extraData;
37     }
38
39     public void setExtraData(String[] extraData) {
40         this.extraData = extraData;
41     }
42
43     public static VellmanTrame createFromByte(byte[] trameF){
44
45         List<String> parts = new ArrayList<String>();
46         for (byte aTrameF : trameF) {
47             parts.add(Integer.toString((aTrameF & 0xFF) + 0x100, 16).substring(1
48         ));
49         }
50         VellmanTrame vel = new VellmanTrame();
51         vel.setSize(Integer.parseInt(parts.get(1)));
52         vel.setCommandByte(Integer.parseInt(parts.get(2)));
53         String[] datas = new String[5];
54         vel.setExtraData(parts.subList(4, 9).toArray(datas));
55
56         return vel;
57     }
58 }
59

```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\META-INF\MANIFEST.MF

```
1 Manifest-Version: 1.0
2 Main-Class: Main
3
4
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\networking\Client.java

```
1 package networking;
2
3 import java.io.*;
4 import java.nio.ByteBuffer;
5 import java.nio.CharBuffer;
6 import java.nio.channels.SocketChannel;
7 import java.nio.charset.Charset;
8 import java.util.Observable;
9 import java.util.Observer;
10 import java.net.InetAddress;
11 import java.net.Socket;
12 import java.net.UnknownHostException;
13
14
15 /**
16  * Class client
17  * Cette classe peremt lors d'une notification d'envoyer un
18  */
19 public class Client implements Observer {
20
21
22     private final Integer port;
23     private final InetAddress address;
24
25     public Client(Integer port, InetAddress address) {
26         this.port = port;
27         this.address = address;
28     }
29
30
31     @Override
32     public void update(Observable o, Object arg) {
33
34
35         String msg = (String) arg;
36
37         try {
38
39             //On initialise une nouvelle socket vers le serveur cible
40             Socket socket = new Socket(this.address, this.port);
41             //ON récupère la sortie standard de la socket
42             OutputStream out = socket.getOutputStream();
43             //On initialise un writer sur la sortie standard
44             OutputStreamWriter osw = new OutputStreamWriter(out);
45             BufferedWriter bw = new BufferedWriter(osw);
46             //On écrit le message reçu depuis le reader vers le serveur
47             bw.write(msg);
48             bw.flush();
49             socket.close();
50
51
52         } catch (IOException exp) {
53
54             System.out.println(exp.getMessage());
55         } catch (Exception exp) {
56
57             System.out.println("Erreur : " + exp.getClass());
58         }
59
60     }
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\networking\Client.java

```
61  
62  
63 }  
64
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\networking\WorkerRunnable.java

```
1 package networking;
2
3 import java.io.InputStream;
4 import java.io.OutputStream;
5 import java.io.IOException;
6 import java.net.Socket;
7
8 /**
9
10 */
11 public class WorkerRunnable implements Runnable{
12
13     protected Socket clientSocket = null;
14     protected String serverText = null;
15
16     public WorkerRunnable(Socket clientSocket, String serverText) {
17         this.clientSocket = clientSocket;
18         this.serverText = serverText;
19     }
20
21     public void run() {
22         try {
23             InputStream input = clientSocket.getInputStream();
24             OutputStream output = clientSocket.getOutputStream();
25             long time = System.currentTimeMillis();
26             output.write(("HTTP/1.1 200 OK\n\nWorkerRunnable: " +
27                 this.serverText + " - " +
28                 time +
29                 "\n").getBytes());
30             output.close();
31             input.close();
32             System.out.println("Request processed: " + time);
33         } catch (IOException e) {
34             //report exception somewhere.
35             e.printStackTrace();
36         }
37     }
38 }
```

Projet SNIR

File - C:\Users\admin\IdeaProjects\Nail\src\networking\MultiThreadedServer.java

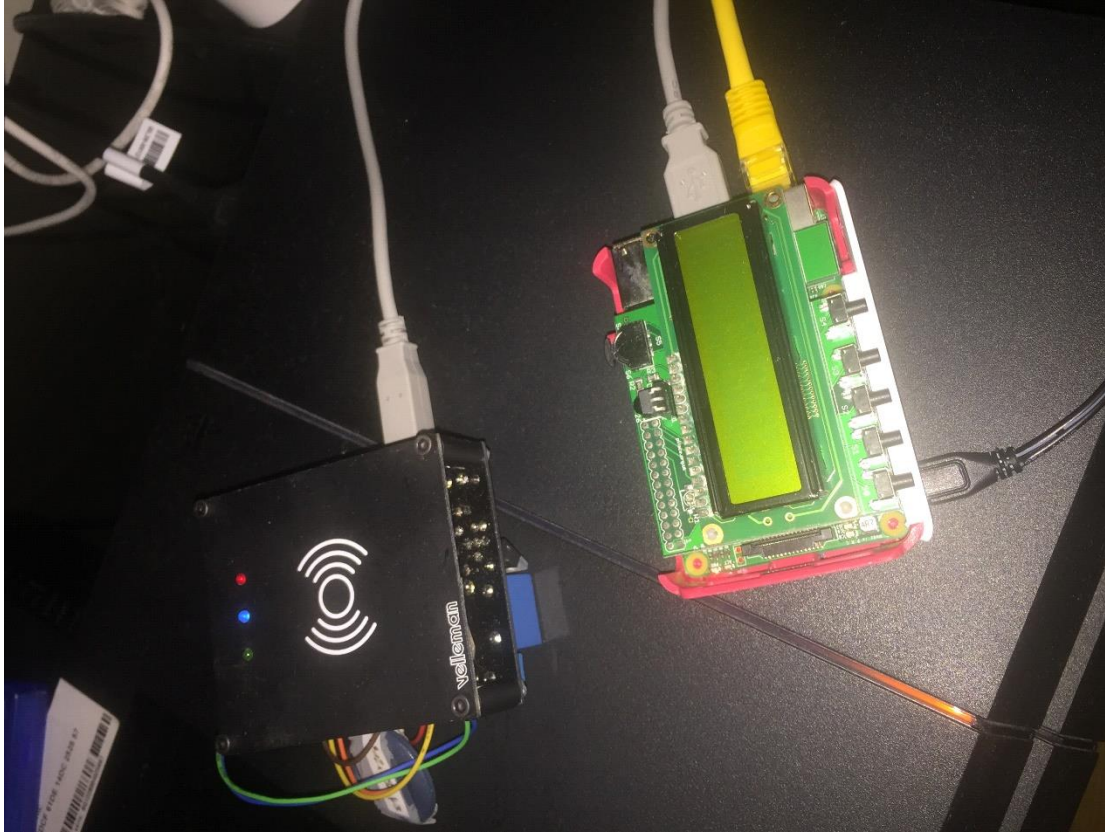
```
1 package networking;
2
3 import java.io.BufferedReader;
4 import java.io.InputStream;
5 import java.io.InputStreamReader;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8 import java.io.IOException;
9 import java.util.Observable;
10
11 public class MultiThreadedServer extends Observable implements Runnable{
12
13     protected int      serverPort      = 8080;
14     protected ServerSocket serverSocket = null;
15     protected boolean   isStopped      = false;
16     protected Thread    runningThread= null;
17
18     public MultiThreadedServer(int port){
19         this.serverPort = port;
20     }
21
22     public void run(){
23         synchronized(this){
24             this.runningThread = Thread.currentThread();
25         }
26         openServerSocket();
27         while(! isStopped()){
28             Socket clientSocket = null;
29             try {
30                 clientSocket = this.serverSocket.accept();
31                 InputStream is = clientSocket.getInputStream();
32                 InputStreamReader isr = new InputStreamReader(is);
33                 BufferedReader br = new BufferedReader(isr);
34                 String message = br.readLine();
35                 setChanged();
36                 //On informe les observers du message reçu
37                 notifyObservers(message);
38
39             } catch (IOException e) {
40                 if(isStopped()) {
41                     System.out.println("Server Stopped.") ;
42                     return;
43                 }
44                 throw new RuntimeException(
45                     "Error accepting client connection", e);
46             }
47             new Thread(
48                 new WorkerRunnable(
49                     clientSocket, "Multithreaded Server")
50             ).start();
51         }
52         System.out.println("Server Stopped.") ;
53     }
54
55     private synchronized boolean isStopped() {
56         return this.isStopped;
57     }
58
59 }
60
```

Projet SNIR

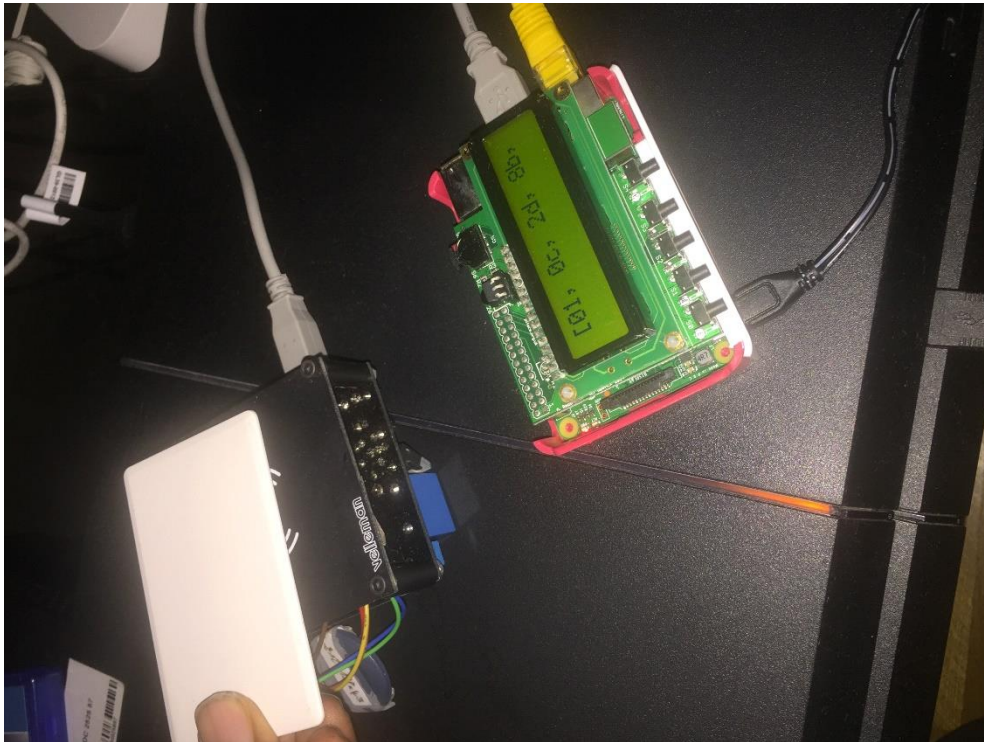
File - C:\Users\admin\IdeaProjects\Nail\src\networking\MultiThreadedServer.java

```
61     public synchronized void stop(){
62         this.isStopped = true;
63         try {
64             this.serverSocket.close();
65         } catch (IOException e) {
66             throw new RuntimeException("Error closing server", e);
67         }
68     }
69
70     private void openServerSocket() {
71         try {
72             this.serverSocket = new ServerSocket(this.serverPort);
73         } catch (IOException e) {
74             throw new RuntimeException("Cannot open port 8080", e);
75         }
76     }
77
78
79
80 }
```

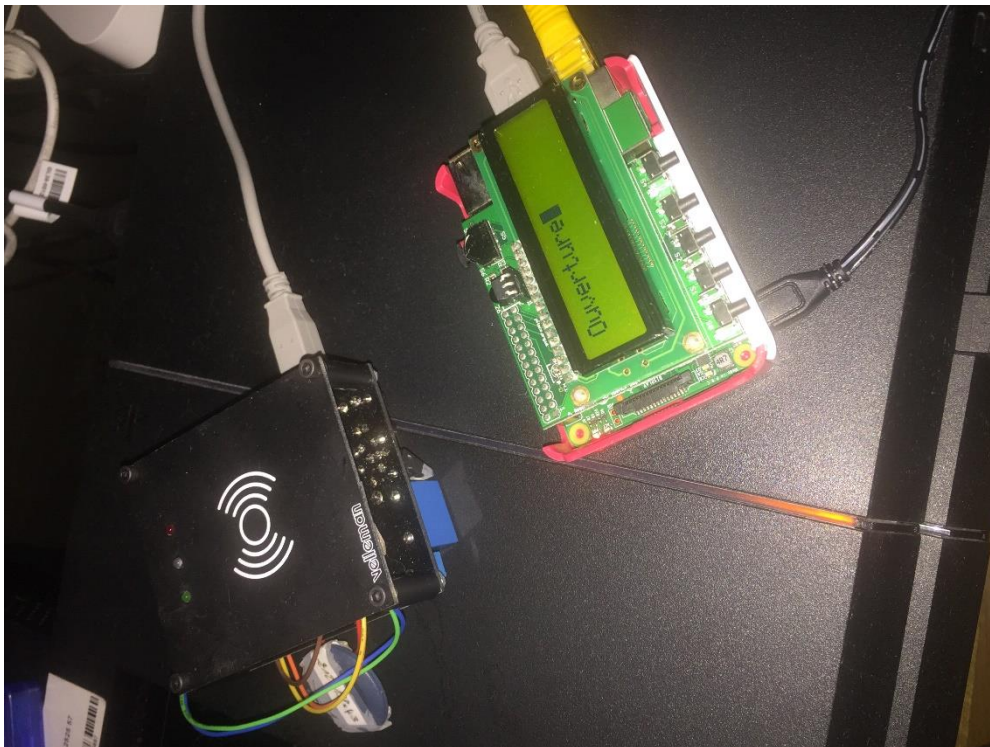

V – Démonstration



Lecture de la carte :



Ouverture :



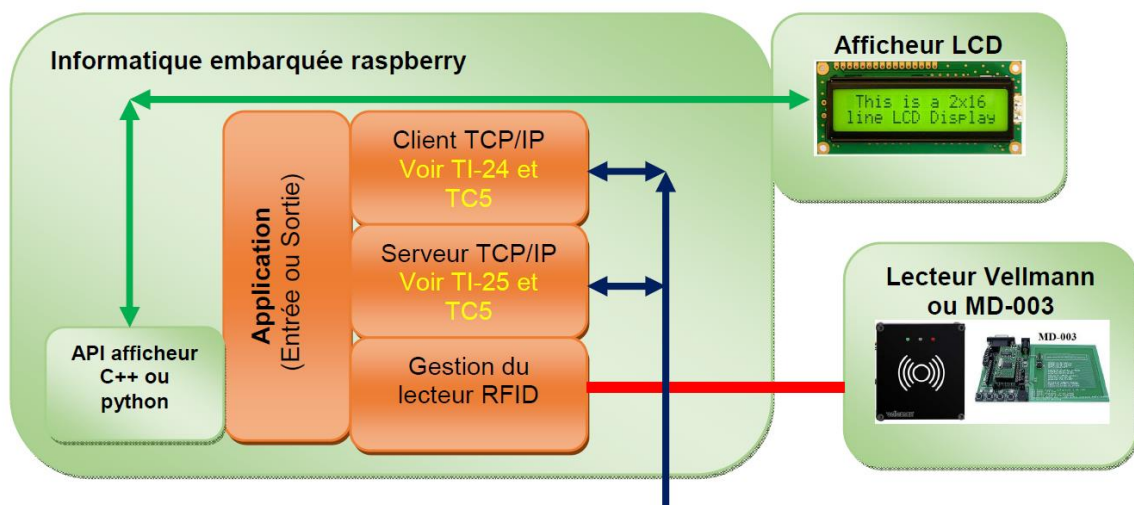
Tâche étudiant n°3

Debaa Nadir

Présentation de la partie personnelle

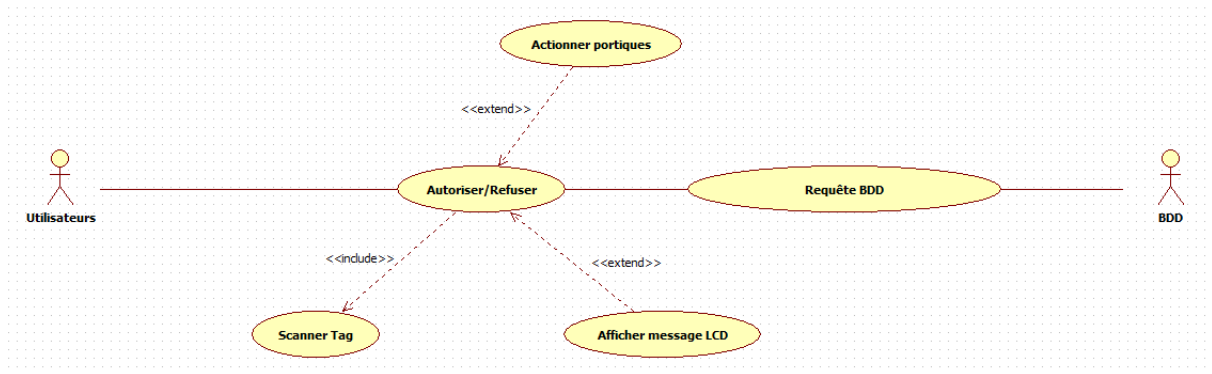
Pour que l'utilisateur puisse accéder au bâtiment, il va passer sa carte devant le lecteur, l'informatique embarquée va récupérer le code RFID et va envoyer ce code au serveur. Le serveur va ensuite interroger la base de données (Base de données MySQL). Ainsi on va pouvoir vérifier que l'utilisateur correspondant à ce code RFID existe réellement.

Un afficheur LCD va pouvoir exposer un message, autorisant ou non l'utilisateur à entrer.



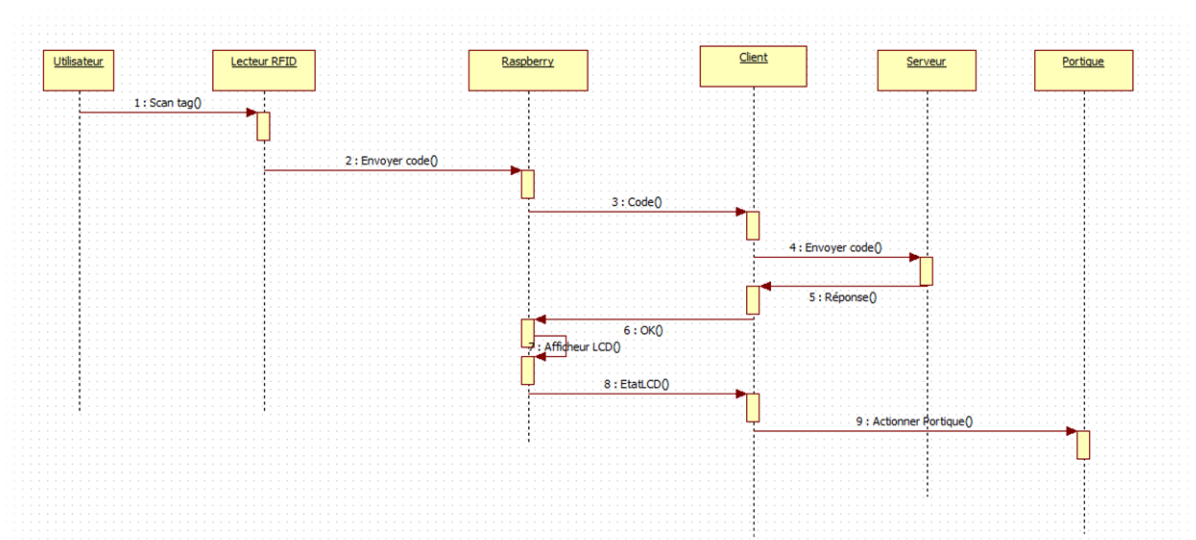
Analyse UML

1. Diagramme des cas d'utilisations personnel



Ici, l'utilisateur passe son badge. Il est donc autorisé ou non selon la base de données (car elle permet l'actionner du portique et l'opérateur qui, à travers le poste de supervision va pouvoir ajouter, supprimer et modifier des données.)

2. Diagramme des séquences personnel



L'utilisateur passe son badge RFID sur le lecteur.

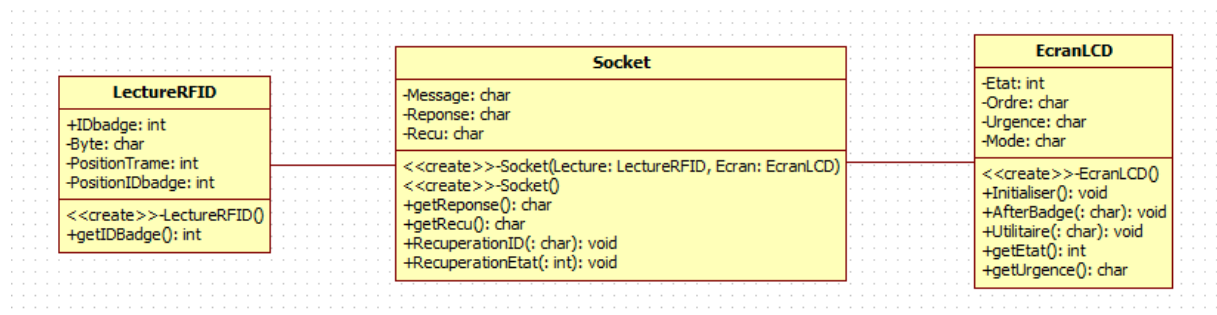
Le code du badge passe par l'informatique embarquée (Raspberry) et est envoyé au serveur.

Le serveur fait la comparaison avec la base de données, et retourne sa réponse au Raspberry.

Selon la réponse, l'afficheur LCD autorisera ou non l'utilisateur.

L'Etat de l'écran LCD est envoyé, et permet donc d'actionner le portique.

3. Diagramme des classes



Choix OS, IDE et langage

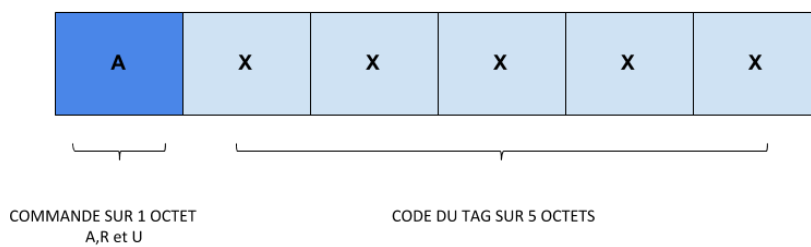
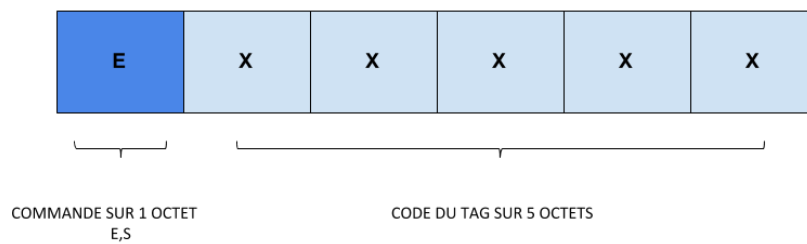
- Système d'Exploitation : Linux
- IDE : CodeBlocks
- Langage de programmation : C++

Spécificités matérielles et logicielles

- PC de développement (IDE C++)
- Lecteur RFID Velleman K8019
- Tag RFID
- Raspberry
- Afficheur LCD compatible avec le Raspberry

Protocole d'échange de données

Protocole TCP (unicast) : Permet la communication entre 2 machines et dispose d'un système d'accusés de réception.



Présentation du lecteur Velleman

CARACTERISTIQUES :

- 250 badges programmables
- Avec interface USB pour la gestion de la configuration
- Application gratuite pour la gestion des cartes RFID pour PC
- Protocole complètement documenté, créez votre propre application
- Il est possible d'ajouter des cartes avec une carte maître
- Sortie relais avec contact de commutation ou de pulsation (NO/NC)
- Temps de pulsation réglable : de 1 à 47 min
- 3 LEDs d'état et ronfleur
- Livré avec 2 cartes
- Fonctionnement autonome

SPECIFICATIONS :

- EM4100 compatible: EM4100 compatible
- Contact relais: 3A/24VCC
- Alimentation: 12VCC ou 5VCC (USB)
- Puissance: max. 100mA
- Dimensions: 69 x 80 x 47 mm



Le port série

L'interface série est une interface asynchrone, ce qui signifie que le signal de cette interface n'est pas synchronisé avec celui d'un bus quelconque. C'est à dire que les bits des données sont envoyés les uns après les autres. Un caractère est composé d'un ensemble de bit. C'est généralement une matrice de 8x8 bits codé par une valeur. Cette valeur est comprise entre 0 et 255 et elle est stockée sur 1 octets c'est à dire 8 bits. Chaque caractère est délimité par un signal de début qui est un bit à 0 et par signal de fin standard qui peut correspondre à un ou deux bits de fin, cela permet d'indiquer que le caractère a été envoyé.

L'interface asynchrone est orienté caractère, c'est à dire que l'on doit utiliser les signaux de début et de fin pour identifier un caractère. L'inconvénient de ce processus c'est qu'il augmente la durée des transferts de presque 25 %. En effet pour chaque ligne de 8 bits il faut au minimum 2 bits.

Le terme "série" vient du fait que les bits sont envoyés les uns après les autres sur un seul fil pour l'émission et un autre fil pour la réception, comme pour le téléphone. Il existe de nombreuses cartes d'extension permettant d'avoir plusieurs ports séries ou port parallèle.

Configuration du port série

La commande **stty** permet de modifier les paramètres de l'interface.

```
eleve@localhost:~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
[eleve@localhost ~]$ stty  
speed 38400 baud; line = 0;  
eol = M-^?; eol2 = M-^?; swtch = M-^?;  
ixany iutf8  
[eleve@localhost ~]$
```

Protocole K8019

Une étiquette RFID est composée de 5 octets.

Les 4 derniers octets sont utilisés pour valider la carte et représentent le numéro unique gravé sur la carte.

Le premier octet représente le numéro d'identification unique du fabricant.

<42> = always 42 hex

<??> = size (in bytes) of the entire packet

<??> = command byte

<ff> = always ff hex

<??>...<??> = optional extra data

Ouverture/Configuration

```
//ouverture driver
int USB = open("/dev/ttyACM0", O_RDWR);
if(USB == -1)
{
    perror("\n Ouverture impossible !! ");
    close(USB);
}
```

On ouvre le port série (/dev/ttyACM0) et on affiche un message d'erreur s'il n'est pas ouvert.

```
termios config;

//acquisition de la configuration par défaut
tcgetattr(USB , &config);
//38400 bauds 8 bits de données
config.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
config.c_lflag = 0;
config.c_oflag = 0;
//pas de parité
config.c_iflag = IGNPAR;
//mode non canonique
config.c_iflag &= ~(ICANON);
//minuteur = 0
config.c_cc[VTIME] = 0;
//read retourne dès la réception d'un caractère
config.c_cc[VMIN] = 1;
//mise à jour de la configuration
tcsetattr(USB , TCSANOW , &config);
tcflush(USB , TCIOFLUSH);
```

On fait toute la configuration : acquisition de la configuration, vitesse (38400 bauds), 8 bits de données, pas de parité, mode non canonique, minuteur = 0, read retourne dès la réception d'un caractère, et mise à jour de la configuration.

Lecture RFID

```
printf("Passez votre carte\n");

for(PositionTrame=0;PositionTrame<9;PositionTrame++)
{
    if (read(USB,&Byte,1)>0)
    {
        printf("%x ",Byte);
        if(PositionTrame>=4)
        {
            IDbadge[PositionIDbadge] = Byte&0xff;
            PositionIDbadge++;
        }
    }
}
```

On fait la lecture RFID (taille de la trame totale : 9) et on l'affiche.

Ensuite, on récupère le code du badge RFID à partir du 4ème octet grâce à PositionIDbadge.

```
printf("\nTaille de L'ID badge: %d\n",sizeof(IDbadge));

printf("L'ID du badge est :");

for (PositionIDbadge=0; PositionIDbadge<TAILLENUM;PositionIDbadge++)
{
    printf(" %x ", IDbadge[PositionIDbadge]&0xff);
}
```

On affiche la taille de l'ID Badge.

On affiche ID Badge.

Présentation du Raspberry

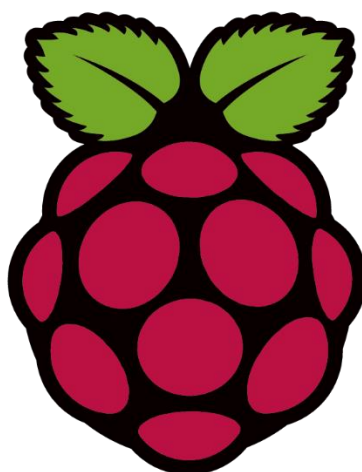
Le Raspberry Pi est un ordinateur dont la particularité est la très petite.

Il a été créé par l'anglais David Braben, dans le cadre de sa fondation Raspberry Pi, dans le but d'encourager l'apprentissage de la programmation informatique.

La fondation Raspberry Pi recommande d'utiliser [Raspbian](#), une distribution GNU/Linux optimisée pour le matériel du Raspberry Pi. Raspbian est basée sur Debian, embarquant l'environnement de bureau LXDE et le navigateur web Midori.

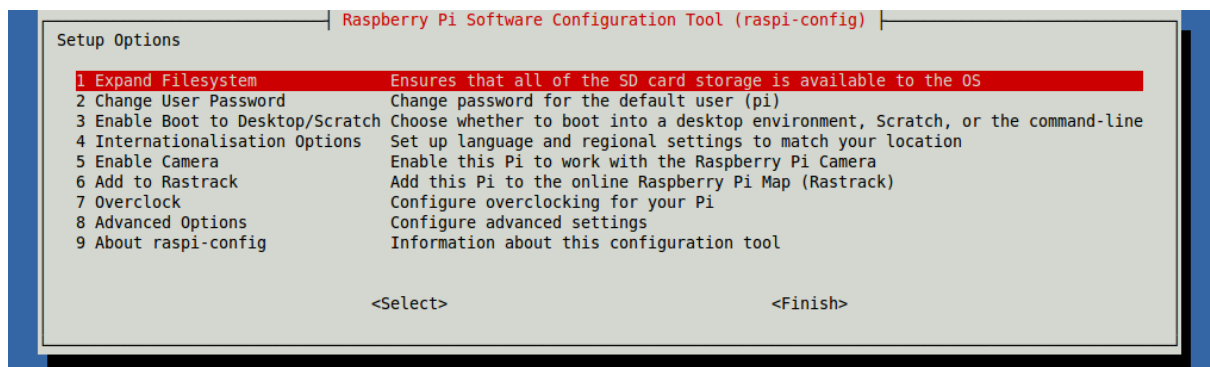
CARACTERISTIQUES :

- 1 lecteur micro SD / SDHC
- 4 sorties USB 2.0
- 1 port RJ45 (Ethernet 10/100)
- 1 port HDMI
- 1 audio Jack 3,5 mm
- GPIO 40 broches



Gestion de l'afficheur LCD (Configuration)

- Mise à jour du système : Sudo apt-get update / Sudo apt-get upgrade
- Configuration raspberry : Sudo raspi-config
- Adresse IP raspberry : 192.168.12.194
Adresse IP du PC : 192.168.12.164



Librairies utilisées

- PiFace Control and Display :
 - Libpifacecad
 - Libmcp23s17



Afficheur LCD (Initialisation)

```
void EcranLCD::Initialiser()  
{  
    Etat=0;  
    Urgence[0]='?';  
    Urgence[1]='F';  
    Urgence[2]='U';  
  
    printf("Initialisation LCD...\n");  
    pifacecad_open();  
    pifacecad_lcd_blink_off();  
    pifacecad_lcd_cursor_off();  
    if(Etat==0)  
        pifacecad_lcd_write("Fermer\n");  
}
```

Initialisation de l'afficheur LCD : Etat = 0
Ecran éteint et pas de curseur.
Fermer.

Afficheur LCD (Autoriser/Refuser)

```
pifacecad_lcd_clear();  
pifacecad_lcd_backlight_on();  
pifacecad_lcd_write("Autoriser\n");  
pifacecad_lcd_write("Ouvert\n");  
sleep(5);  
pifacecad_lcd_backlight_off();  
pifacecad_lcd_clear();  
if(Etat==0)  
    pifacecad_lcd_write("Mode Entre\n");  
else  
    pifacecad_lcd_write("Mode Sortie\n");  
  
pifacecad_lcd_write("Fermer\n");
```

Si l'utilisateur est autorisé, on allume l'afficheur : Autoriser/Ouvert et on éteint l'afficheur.
On précise ensuite le Mode : Entré/Sortie.
Fermer

```
pifacecad_lcd_clear();  
pifacecad_lcd_backlight_on();  
pifacecad_lcd_write("Refuser\n");  
pifacecad_lcd_write("Fermer\n");  
sleep(5);  
pifacecad_lcd_backlight_off();  
pifacecad_lcd_clear();  
if(Etat==0)  
    pifacecad_lcd_write("Mode Entre\n");  
else  
    pifacecad_lcd_write("Mode Sortie\n");  
pifacecad_lcd_write("Fermer\n");
```

Si l'utilisateur est refusé, on allume l'afficheur : Refuser/Fermer et on éteint l'afficheur.
On précise ensuite le Mode : Entré/Sortie.
Ferme

Socket

Les sockets ont été mises au point en 1984, lors de la création des distributions BSD (*Berkeley Software Distribution*).

Apparues pour la première fois dans les systèmes [UNIX](#), les sockets sont des points de terminaison mis à l'écoute sur le [réseau](#), afin de faire transiter des données logicielles. Celles-ci sont associées à un numéro de [port](#).

Les ports sont des numéros allant de 0 à 216-1 inclus (soit 65535). Chacun de ces ports est associé à une application (à savoir que les 1024 premiers ports sont réservés à des utilisations bien précises).

Les sockets sont aussi associées à un [protocole](#). Dans notre cas nous utiliserons le protocole TCP/IP. Les sockets servent à établir une transmission de flux de données ([octets](#)) entre deux machines ou applications.

Création/Configuration (Client)

```
SOCKET sock;  
SOCKADDR_IN sin;  
  
// Création de la socket  
sock = socket(AF_INET, SOCK_STREAM, 0);  
  
// Configuration de la connexion  
sin.sin_addr.s_addr = inet_addr(ADDRIP);  
sin.sin_family = AF_INET;  
sin.sin_port = htons(PORT);  
  
char* p = Ecran.getUrgence();  
char urg[3];  
  
for (int i=0;i<3;i++)  
{  
    urg[i]=*p;  
    p++;  
}  
  
if(urg[0]=='?' && urg[1] == 'F' && urg[2] =='U')  
{  
    printf("L'urgence est désactivée : %s \nConnexion autorisée \n",urg);  
  
    // Si le client arrive à se connecter  
    if(connect(sock, (SOCKADDR*)&sin, sizeof(sin)) != SOCKET_ERROR)  
    {  
        printf("Connexion à %s sur le port %d\n", inet_ntoa(sin.sin_addr), htons(sin.sin_port));  
    }
```

Création de la socket : `sock = socket(AF_INET, SOCK_STREAM, 0);`

Configuration de la connexion : Adresse IP / Port

Récupération de l'état d'urgence et Autorisation de la connexion.

Envoi/Réponse

On récupère l'ID du Badge

On envoie l'ID Badge au serveur et on récupère la réponse dans Socket::Reponse

```
char *myMessage = Lecture.getIDBadge();

Socket::RecuperationEtat(Ecran.getEtat());

for (int i=0;i<TAILLENUM;i++)
    printf("%x ",myMessage[i]&0xff);

//Envoyer Message
if( send(sock , myMessage , 5 , 0) < 0)
{
    perror("Send failed \n");
}
cout<<"\nData send\n";

//Recevoir la reponse du serveur
if( recv(sock , Socket::Reponse , sizeof(Socket::Reponse) , 0) < 0)
{
    puts("recv failed\n");
}
printf("Serveur : %s\n",Socket::Reponse);
}
else
{
    printf("Impossible de se connecter\n");

    // On ferme la socket précédemment ouverte
    close(sock);
}
```

Création/Configuration/Réponse (Serveur)

```
SOCKET sockU;  
SOCKADDR_IN sinU;  
  
// Création de la socket  
sockU = socket(AF_INET, SOCK_STREAM, 0);  
  
// Configuration de la connexion  
sinU.sin_addr.s_addr = inet_addr(ADDRIP);  
sinU.sin_family = AF_INET;  
sinU.sin_port = htons(PORT1);  
  
// Si le client arrive à se connecter  
if(connect(sockU, (SOCKADDR*)&sinU, sizeof(sinU)) != SOCKET_ERROR)  
{  
    printf("Connexion du client utilitaire à %s sur le port %d\n", inet_ntoa(sinU.sin_addr), htons(sinU.sin_port));
```

Création de la socket : sock = socket(AF_INET, SOCK_STREAM, 0) ;

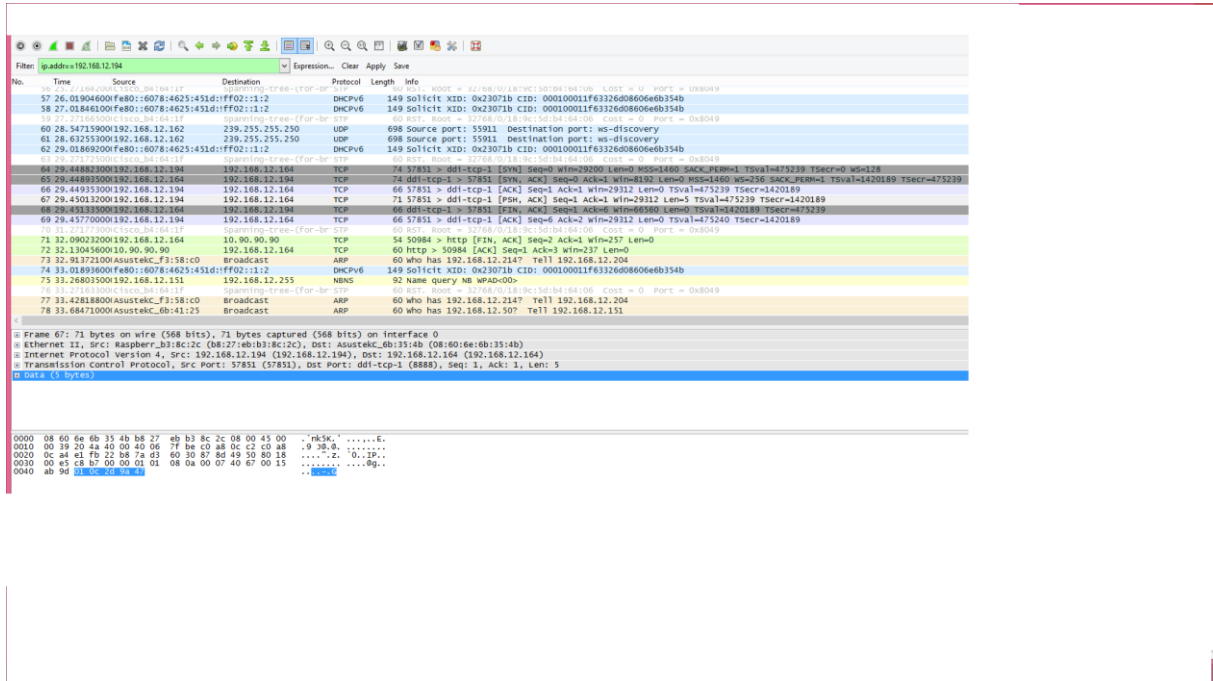
Configuration de la connexion : Adresse IP / Port

Récupération de l'état d'urgence et Autorisation de la connexion.

```
//Recevoir reponse du serveur utilitaire  
if( recv(sockU , Socket::Recu, sizeof(Socket::Recu) , 0) < 0)  
{  
    puts("Recv failed");  
}  
  
printf("Serveur utilitaire : %s\n", Socket::Recu);
```

On attend la réponse du serveur (Socket::Recu)

WireShark



The screenshot shows a Wireshark capture of network traffic on interface 0. The packet list displays various protocols including DHCP, ARP, and TCP. The packet details pane shows the structure of the selected packet (Frame 67: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0). The packet bytes pane shows the raw data in hexadecimal and ASCII.

Vérification de l'envoi de l'ID Badge.

ANNEXE

```
#ifndef LECTURERFID_H
#define LECTURERFID_H
#define TAILLENUM 5

class LectureRFID
{
public:
    char IDbadge[TAILLENUM];
    LectureRFID();
    char* getIDBadge();

protected:
private:
    char Byte;
    int PositionTrame, PositionIDbadge;
};

#endif // LECTURERFID_H
```

```
#include "LecteurRFID.h"
#include <iostream>
#include <unistd.h> // UNIX standard function definitions
#include <fcntl.h> // File control definitions
#include <errno.h> // Error number definitions
#include <termios.h> // POSIX terminal control definitions
#include <stdio.h>
#include <stdlib.h>

LectureRFID::LectureRFID()
{
    //ouverture driver /dev/ttyUSB0 O_RDWR
    int USB = open("/dev/ttyACM0", O_RDWR);
    if(USB == -1)
    {
        perror("\n Ouverture impossible !! ");
        close(USB);
    }

    termios config;

    PositionIDbadge=0;

    //acquisition de la configuration par défaut
    tcgetattr(USB, &config);
    //9600 bauds 8 bits de données
    config.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
    config.c_lflag = 0;
    config.c_oflag = 0;
    //pas de parité
    config.c_iflag = IGNPAR;
    //mode non canonique
    config.c_iflag &= ~(ICANON);
    //minuteur = 0
    config.c_cc[VTIME] = 0;
    //read retourne dès la réception d'un caractère
    config.c_cc[VMIN] = 1;
    //mise à jour de la configuration
    tcsetattr(USB, TCSANOW, &config);
}
```

```

tcf flush(USB , TCIOFLUSH);

printf("Passez votre carte\n");

for(PositionTrame=0;PositionTrame<9;PositionTrame++)
{
    if (read(USB,&Byte,1)>0)
    {
        printf("%x ",Byte);
        if(PositionTrame>=4)
        {
            IDbadge[PositionIDbadge] = Byte;
            PositionIDbadge++;
        }
        } // if new data is available on the serial port, print it out
    }

printf("\nTaille de L'ID badge: %d\n",sizeof(IDbadge));

printf("L'ID du badge est :");

for (PositionIDbadge=0; PositionIDbadge<5;PositionIDbadge++)
{
    printf(" %x ", IDbadge[PositionIDbadge]);
}

printf("\n");

close(USB);
}

char * LectureRFID::getIDBadge()
{
    return &(LectureRFID::IDbadge[0]);
}

```

```
#ifndef EcranLCD_H
#define EcranLCD_H

class EcranLCD
{
public:
    EcranLCD();
    void Initialiser();
    void AfterBadge(char *);
    void Utilitaire(char *);
    int getEtat();
    char* getUrgence();

protected:
private:
    int Etat;
    char Ordre[2];
    char Urgence[3];
    char Mode[2];
};

#endif // EcranLCD
```

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "pifacecad.h"
#include "mcp23s17.h"
#include "EcranLCD.h"

EcranLCD::EcranLCD()
{
    Initialiser();
}

void EcranLCD::Initialiser()
{
    Etat=0;
    Urgence[0]='?';
    Urgence[1]='F';
    Urgence[2]='U';

    pifacecad_open();
    pifacecad_lcd_blink_off();
    pifacecad_lcd_cursor_off();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    pifacecad_lcd_write("Fermer\n");
}

void EcranLCD::AfterBadge(char * PRep)
{
    for (int i=0;i<2;i++)
    {
        Ordre[i]=*PRep;
        PRep++;
    }
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        if(Ordre[0] == '?' && Ordre[1]=='A')
        {
```

```

    pifacecad_lcd_clear();
    pifacecad_lcd_backlight_on();
    pifacecad_lcd_write("Autoriser\n");
    pifacecad_lcd_write("Ouvert\n");
    sleep(5);
    pifacecad_lcd_backlight_off();
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");

    pifacecad_lcd_write("Fermer\n");
}

if(Ordre[0] == '?' && Ordre[1]=='R')
{
    pifacecad_lcd_clear();
    pifacecad_lcd_backlight_on();
    pifacecad_lcd_write("Refuser\n");
    pifacecad_lcd_write("Fermer\n");
    sleep(5);
    pifacecad_lcd_backlight_off();
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");
    pifacecad_lcd_write("Fermer\n");
}
}

void EcranLCD::Utilitaire(char * PRecu)
{
    char temp[3];
    for (int i=0;i<3;i++)
    {
        temp[i]=*PRecu;
        PRecu++;
    }
}

```

```
if((temp[1]=='E') || (temp[1]=='S'))
{
    Mode[0]=temp[0];
    Mode[1]=temp[1];
}

if((temp[0]=='?' && temp[1]=='D') || (temp[1]=='F' && temp[2]=='U'))
{
    for (int i=0;i<3;i++)
    {
        Urgence[i]=temp[i];
    }
    if(Urgence[0] == '?' && Urgence[1]=='D' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("Urgence\n");
        pifacecad_lcd_write("Ouvert\n");
    }

    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("FinUrgence\n");
        sleep(2);
        pifacecad_lcd_backlight_off();
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");
        pifacecad_lcd_write("Fermer\n");
    }
}
if(Mode[0] == '?' && Mode[1]=='E')
{
    Etat=0;
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
```



```
{
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");
    pifacecad_lcd_write("Fermer\n");
}
}
if(Mode[0] == '?' && Mode[1]=='S')
{
    Etat=1;
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");
        pifacecad_lcd_write("Fermer\n");
    }
}
printf("Mode: %s\n", Mode);

int EcranLCD::getEtat()
{
    return(EcranLCD::Etat);
}

char* EcranLCD::getUrgence()
{
    return &(EcranLCD::Urgence[0]);
}
```

```
#ifndef SOCKET_H
#define SOCKET_H

#include "LecteurRFID.h"
#include "EcranLCD.h"

class Socket
{
public:
    Socket (LectureRFID Lecture, EcranLCD Ecran);
    Socket ();
    char* getReponse ();
    char* getRecu ();
    void RecuperationID(char*);
    void RecuperationEtat(int);
protected:
private:
    char Message[TAILLENUM];
    char Reponse[2];
    char Recu[3];
};

#endif // SOCKET_H
```

```
#include "Socket.h"
#include "LecteurRFID.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h> //strlen
#include <string> //string
#include <iostream> //cout
#include <cstdlib>

#define SOCKET_ERROR -1
#define ADDRIP "192.168.12.164"
#define PORT 8888
#define PORT1 8888

typedef int SOCKET;
typedef struct sockaddr_in SOCKADDR_IN;
typedef struct sockaddr SOCKADDR;

Socket::Socket (LectureRFID Lecture, EcranLCD Ecran)
{
    using namespace std;

    SOCKET sock;
    SOCKADDR_IN sin;

    // Cr ation de la socket
    sock = socket(AF_INET, SOCK_STREAM, 0);

    // Configuration de la connexion
    sin.sin_addr.s_addr = inet_addr(ADDRIP);
    sin.sin_family = AF_INET;
    sin.sin_port = htons(PORT);

    char* p = Ecran.getUrgence();
    char urq[3];
```

```
for (int i=0;i<3;i++)
{
    urg[i]=*p;
    p++;
}

if(urg[0]=='?' && urg[1] == 'F' && urg[2] =='U')
{
    printf("%'urgence est desactivée : %s \nConnexion autorisée \n",urg);

    // Si le client arrive à se connecter
    if(connect(sock, (SOCKADDR*)&sin, sizeof(sin)) != SOCKET_ERROR)
    {
        printf("Connexion à %s sur le port %d\n", inet_ntoa(sin.sin_addr), htons(sin.sin_port));

        char *myMessage = Lecture.getIDBadge();

        Socket::RecuperationID(myMessage);

        Socket::RecuperationEtat(Ecran.getEtat());

        //Envoyer Message
        if( send(sock , myMessage , 20 , 0) < 0)
        {
            perror("Send failed \n");
        }
        cout<<"\nData send\n";

        //Recevoir la reponse du serveur
        if( recv(sock , Socket::Reponse , sizeof(Socket::Reponse) , 0) < 0)
        {
            puts("recv failed\n");
        }
        printf("Serveur : %s\n",Socket::Reponse);
    }
    else
    {
        printf("Impossible de se connecter\n");
    }
}
```

```
        // On ferme la socket précédemment ouverte
        close(sock);
    }
}

else
    printf("L'urgence est activée : %s \nConnexion Refusée \n",urg);
}

void Socket::RecuperationID(char* myMessage)
{
    for (int i=0;i<TAILLENUM;i++)
        printf("%x ",myMessage[i]);
}

void Socket::RecuperationEtat(int E)
{
    if(E==0)
    {
        Message[0]='?';
        Message[1]='E';
    }

    if(E==1)
    {
        Message[0]='?';
        Message[1]='S';
    }
}

char* Socket::getReponse()
{
    return &(Socket::Reponse[0]);
}

char* Socket::getRecu()
{

```

```
        return &(Socket::Recu[0]);
    }

Socket::Socket()
{
    using namespace std;

    SOCKET sockU;
    SOCKADDR_IN sinU;

    // Création de la socket
    sockU = socket(AF_INET, SOCK_STREAM, 0);

    // Configuration de la connexion
    sinU.sin_addr.s_addr = inet_addr(ADDRIP);
    sinU.sin_family = AF_INET;
    sinU.sin_port = htons(PORT1);

    // Si le client arrive à se connecter
    if(connect(sockU, (SOCKADDR*)&sinU, sizeof(sinU)) != SOCKET_ERROR)
    {
        printf("Connexion du client utilitaire à %s sur le port %d\n", inet_ntoa(sinU.sin_addr), htons(sinU.sin_port));

        //Recevoir reponses du serveur utilitaire
        if( recv(sockU , Socket::Recu, sizeof(Socket::Recu) , 0) < 0)
        {
            puts("recv failed");
        }

        printf("Serveur utilitaire : %g\n",Socket::Recu);
    }
    else
        printf("Impossible de se connecter\n");
}
```

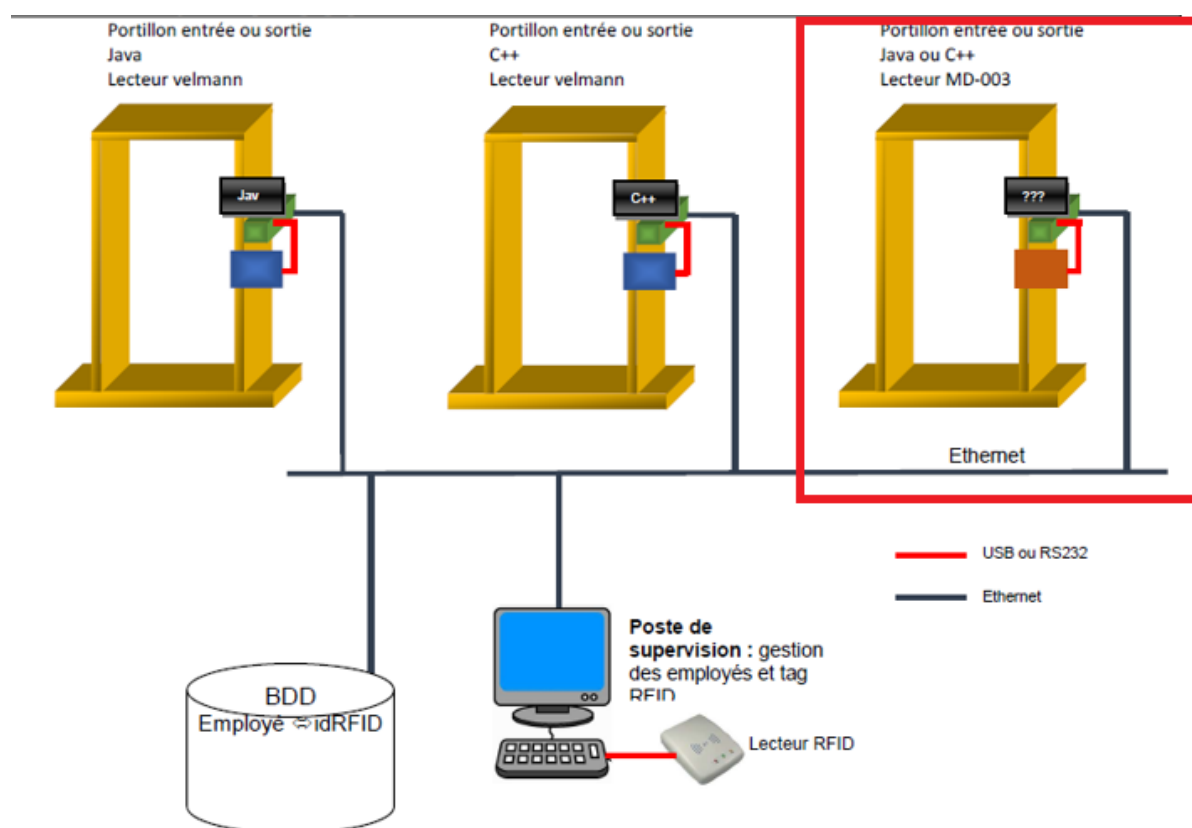
Tâche étudiant n°4

Baïche Othmane

Présentation de la partie personnelle

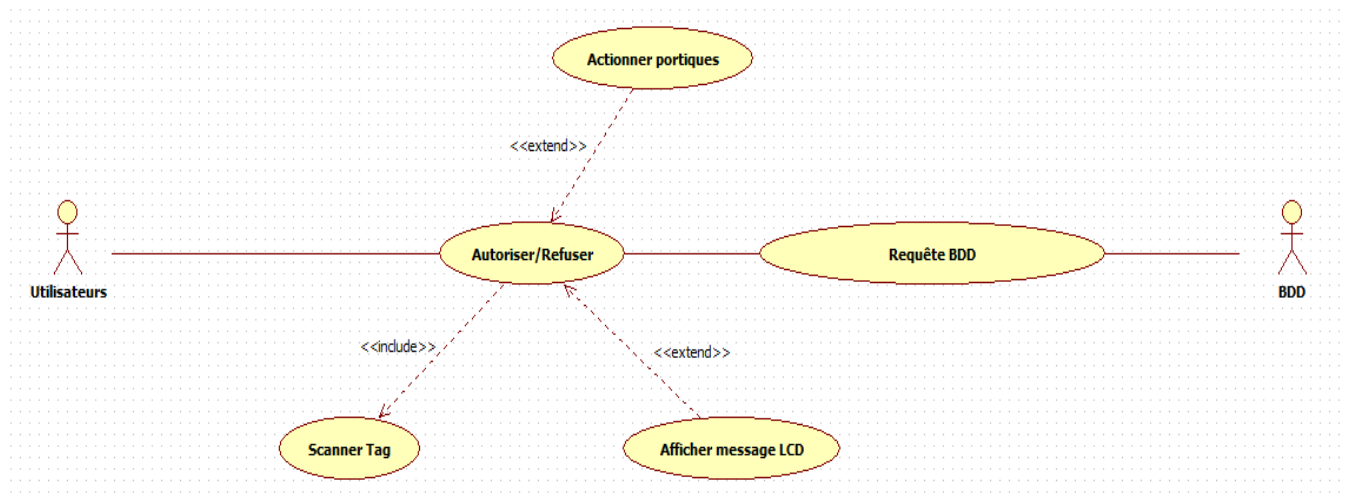
Pour que l'utilisateur puisse accéder au bâtiment, il va passer sa carte devant le lecteur, l'informatique embarquée va récupérer le code RFID et va envoyer ce code au serveur. Le serveur va ensuite interroger la base de données (Base de données MySQL). Ainsi on va pouvoir vérifier que l'utilisateur correspondant à ce code RFID existe réellement.

Un afficheur LCD va pouvoir exposer un message, autorisant ou non l'utilisateur à entrer.



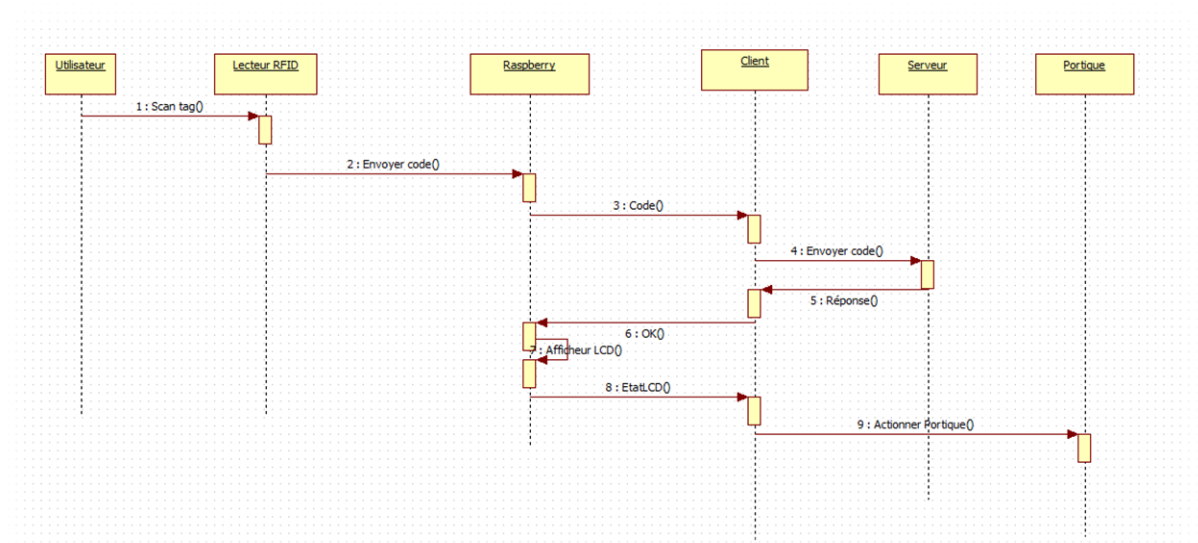
Analyse UML

1. Diagramme des cas d'utilisations personnel



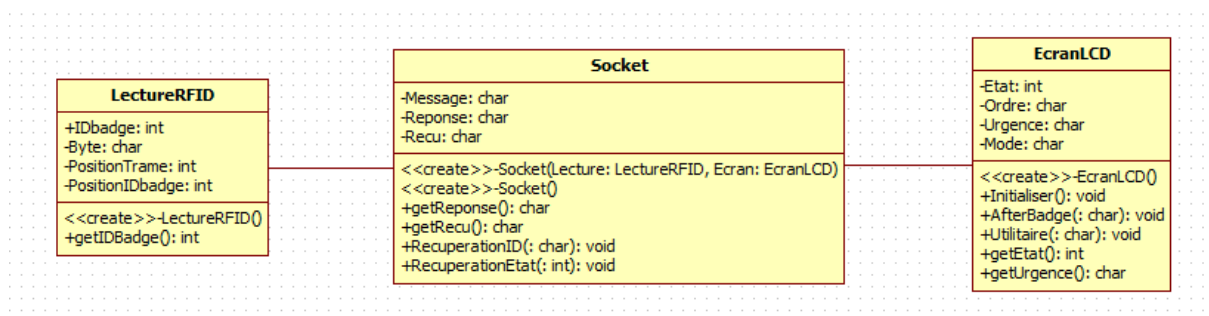
Ici, l'utilisateur passe son badge. Il est donc autorisé ou non selon la base de données (car elle permet l'actionner du portique et l'opérateur qui, à travers le poste de supervision va pouvoir ajouter, supprimer et modifier des données.)

2. Diagramme des séquences personnel



L'utilisateur passe son badge RFID sur le lecteur.
 Le code du badge passe par l'informatique embarquée (Raspberry) et est envoyé au serveur.
 Le serveur fait la comparaison avec la base de données, et retourne sa réponse au Raspberry.
 Selon la réponse, l'afficheur LCD autorisera ou non l'utilisateur.
 L'Etat de l'écran LCD est envoyé, et permet donc d'actionner le portique

3. Diagramme des classes



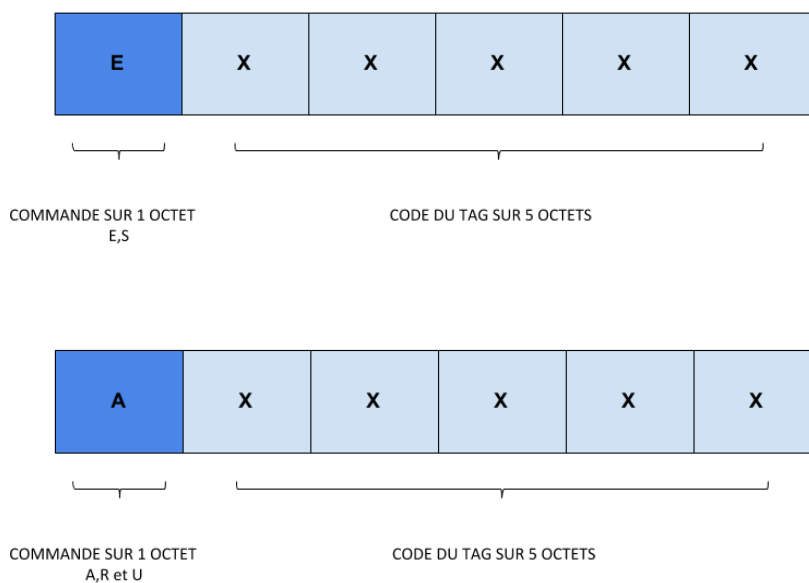
Choix OS, IDE et langage

- Système d'Exploitation : Linux
- IDE : CodeBlocks
- Langage de programmation : C++

Spécificités matérielles et logicielles

- PC de développement (IDE C++)
- Lecteur RFID MD-003
- Tag RFID
- Raspberry
- Afficheur LCD compatible avec le Raspberry

Protocole d'échange de données



Protocole TCP (unicast) : Permet la communication entre 2 machines et dispose d'un système d'accusés de réception.

Lecteur MD-003

Caractéristiques techniques

Alimentation	4,5 à 5,5 Vcc
Consommation	75 mA
Fréquence RFID	125 KHz
Distance de lecture max.	12 cm max.
Débit communication RFID	1953 bps
Durée de lecture d'un "bloc"	2 lectures / sec.
Sortie série	9600 bps / 8 bits / 1 stop / sans parité / niveaux TTL



Protocole

Une étiquette RFID est composée de 5 octets.

Le premier octet représente le numéro d'identification unique du fabricant , 01 pour le MD-003

Les 4 derniers octets sont utilisés pour valider la carte et représentent le numéro unique gravé sur la carte.

<01> = Always 01 hex

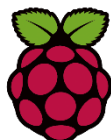
<??> = size (in bytes) of the entire packet

<??> = command byte

<ff> = always ff hex

<??>...<??> = optional extra data

Présentation du Raspberry



Le **Raspberry Pi** est un nano-ordinateur monocarte à processeur ARM conçu par le créateur de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi .

Cet ordinateur, qui a la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique

La fondation Raspberry Pi recommande d'utiliser [Raspbian](#), une distribution GNU/Linux optimisée pour le matériel du Raspberry Pi. Raspbian est basée sur Debian, embarquant l'environnement de bureau LXDE et le navigateur web Midori.

CARACTERISTIQUES :

- 1 lecteur micro SD / SDHC
- 1 audio Jack 3,5 mm
- 4 sorties USB 2.0
- GPIO 40 broches
- 1 port RJ45 (Ethernet 10/100)
- 1 port HDMI
- 1 audio Jack 3,5 mm
- GPIO 40 broches

Lecteur MD-003

Ouverture et Configuration :

```
//ouverture driver /dev/ttyUSB0 O_RDWR
int USB = open("/dev/ttyUSB0" , O_RDWR);
if(USB == -1)
{
    perror("\n Ouverture impossible !! ");
    close(USB);
}

termios config;

PositionIDbadge=0;

//acquisition de la configuration par défaut
tcgetattr(USB , &config);
//9600 bauds 8 bits de données
config.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
config.c_lflag = 0;
config.c_oflag = 0;
//pas de parité
config.c_iflag = IGNPAR;
//mode non canonique
config.c_iflag &= ~(ICANON);
//minuteur = 0
config.c_cc[VTIME] = 0;
//read retourne dès la réception d'un caractère
config.c_cc[VMIN] = 1;
//mise à jour de la configuration
tcsetattr(USB , TCSANOW , &config);
tcflush(USB . TCIOFLUSH);
```

On configure le lecteur en faisant l'acquisition de la configuration . Vitesse (9600 bauds)
8 bits de données, pas de parité, mode non canonique.
Minuteur = 0, read retourne dès la réception d'un caractère
Mise à jour de la configuration.

Lecture RFID

```
printf("Passez votre carte\n");

for(PositionTrame=0;PositionTrame<9;PositionTrame++)
{
    if (read(USB,&Byte,1)>0)
    {
        printf("%x ",Byte);
        if(PositionTrame>=4)
        {
            IDbadge[PositionIDbadge] = Byte&0xff;
            PositionIDbadge++;
        }
    } // if new data is available on the serial port, print

}

printf("\nTaille de L'ID badge: %d\n",sizeof(IDbadge));

printf("L'ID du badge est :");

for (PositionIDbadge=0; PositionIDbadge<5;PositionIDbadge++)
{
    printf(" %x ", IDbadge[PositionIDbadge]&0xff);
}
```

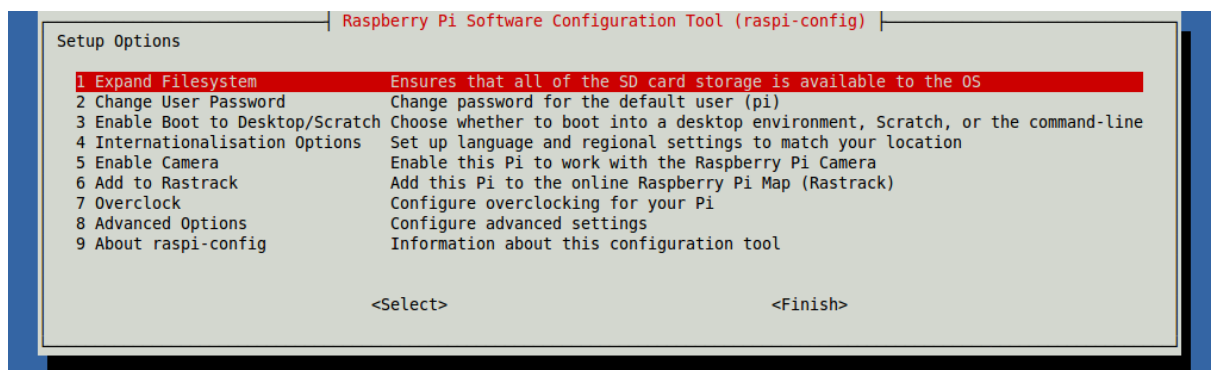
On fait la lecture RFID

On affiche la trame complète qui est de 9 octets

On récupère ensuite le code du badge RFID à partir du 4ème octet grâce à PositionIDbadge

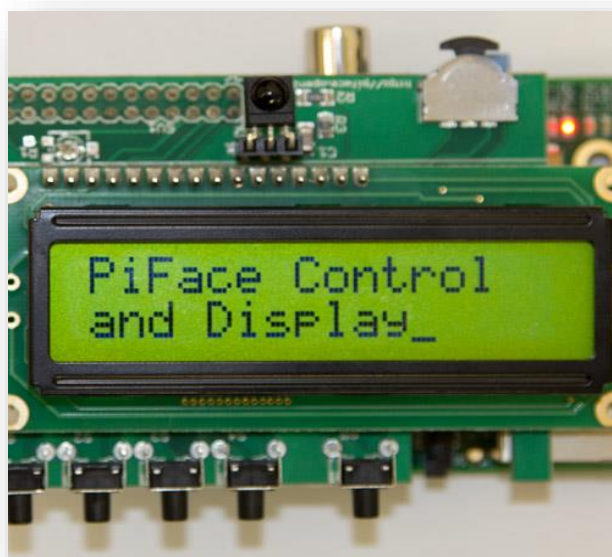
Librairies utilisées et Gestion de l'afficheur LCD (Configuration)

- Mise à jour du système : Sudo apt-get update / Sudo apt-get upgrade
- Configuration raspberry : Sudo raspi-config
- Adresse IP raspberry : 192.168.12.194
Adresse IP du PC : 192.168.12.164



PiFace Control and Display :

- Libpifacecad
- Libmcp23s17



Afficheur LCD

Initialisation

```
void EcranLCD::Initialiser()  
{  
    Etat=0;  
    Urgence[0]='?';  
    Urgence[1]='F';  
    Urgence[2]='U';  
  
    printf("Initialisation LCD...\n");  
    pifacecad_open();  
    pifacecad_lcd_blink_off();  
    pifacecad_lcd_cursor_off();  
    if(Etat==0)  
        pifacecad_lcd_write("Fermer\n");  
}
```

Initialisation de l'afficheur LCD : Etat = 0
Ecran éteint et pas de curseur.
Fermer.

Ouverture et fermeture

```

    pifacecad_lcd_clear();
    pifacecad_lcd_backlight_on();
    pifacecad_lcd_write("Autoriser\n");
    pifacecad_lcd_write("Ouvert\n");
    sleep(5);
    pifacecad_lcd_backlight_off();
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");

    pifacecad_lcd_write("Fermer\n");
}

if(Ordre[0] == '?' && Ordre[1]=='R')
{
    pifacecad_lcd_clear();
    pifacecad_lcd_backlight_on();
    pifacecad_lcd_write("Refuser\n");
    pifacecad_lcd_write("Fermer\n");
    sleep(5);
    pifacecad_lcd_backlight_off();
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");
    pifacecad_lcd_write("Fermer\n");
}
}

```

Si l'utilisateur est autorisé, on allume l'afficheur : Autoriser/Ouvert et on éteint l'afficheur.

On précise ensuite le Mode : Entré/Sortie.

Fermer

Si l'utilisateur est refusé, on allume l'afficheur : Refuser/Fermer et on éteint l'afficheur.

On précise ensuite le Mode : Entré/Sortie.

Ferme

Socket

Les sockets ont été mises au point en 1984, lors de la création des distributions BSD (*Berkeley Software Distribution*). Dans le contexte des logiciels, on peut le traduire par « connecteur réseau » ou « interface de connexion »¹.

Apparu dans les systèmes UNIX, un socket est un élément logiciel qui est aujourd'hui répandu dans la plupart des systèmes d'exploitation. Il s'agit d'une interface logicielle avec les services du système d'exploitation, grâce à laquelle un développeur exploitera facilement et de manière uniforme les services d'un protocole réseau.

Les ports sont des numéros allant de 0 à 216-1 inclus (soit 65535). Chacun de ces ports est associé à une application (à savoir que les 1024 premiers ports sont réservés à des utilisations bien précises).

Les sockets sont aussi associées à un protocole. Dans notre cas nous utiliserons le protocole TCP/IP. Les sockets servent à établir une transmission de flux de données (octets) entre deux machines ou applications.

Il lui sera ainsi par exemple aisé d'établir une session TCP, puis de recevoir et d'expédier des données grâce à elle. Cela simplifie sa tâche car cette couche logicielle, de laquelle il requiert des services en appelant des fonctions, masque le travail nécessaire de gestion du réseau, pris en charge par le système. Le terme socket désigne en pratique chaque variable employée dans un programme afin de gérer l'une des sessions.

Socket : Création et configuration

```
// Création de la socket
sock = socket(AF_INET, SOCK_STREAM, 0);

// Configuration de la connexion
sin.sin_addr.s_addr = inet_addr(ADDRIP);
sin.sin_family = AF_INET;
sin.sin_port = htons(PORT);

char* p = Ecran.getUrgence();
char urg[3];

for (int i=0;i<3;i++)
{
    urg[i]=*p;
    p++;
}

if(urg[0]=='?' && urg[1] == 'F' && urg[2] =='U')
{
    printf("L'urgence est désactivée : %s \nConnexion autorisée \n",urg);
}
```

Création de la socket : `sock = socket(AF_INET, SOCK_STREAM, 0);`

Configuration de la connexion : Adresse IP / Port

```
// Si le client arrive à se connecter
if(connect(sock, (SOCKADDR*)&sin, sizeof(sin)) != SOCKET_ERROR)
{
    printf("Connexion à %s sur le port %d\n", inet_ntoa(sin.sin_addr), htons(sin.sin_port));

    int *myMessage = Lecture.getIDBadge();

    //Socket::RecuperationID(myMessage);

    //Socket::RecuperationEtat(Ecran.getEtat());

    for (int i=0;i<TAILLENUM;i++)
        printf("%x ",myMessage[i]&0xff);
}
```

Récupération de l'état d'urgence et Autorisation de la connexion.

Socket : Envoyer et recevoir

On récupère l'ID du Badge

```
char *myMessage = Lecture.getIDBadge();

Socket::RecuperationEtat(Ecran.getEtat());

for (int i=0;i<TAILLENUM;i++)
    printf("%x ",myMessage[i]&0xff);

//Envoyer Message
if( send(sock , myMessage , 5 , 0) < 0)
{
    perror("Send failed \n");
}
cout<<"\nData send\n";

//Recevoir la reponse du serveur
if( recv(sock , Socket::Reponse , sizeof(Socket::Reponse) , 0) < 0)
{
    puts("recv failed\n");
}
printf("Serveur : %s\n",Socket::Reponse);
}
else
{
    printf("Impossible de se connecter\n");

    // On ferme la socket précédemment ouverte
    close(sock);
}
```

On envoie l'ID Badge au serveur et on récupère la réponse dans Socket::Reponse

Socket :Serveur

Création de la socket : `sock = socket(AF_INET, SOCK_STREAM, 0) ;`

Configuration de la connexion : Adresse IP / Port

```
SOCKET sockU;
SOCKADDR_IN sinU;

// Création de la socket
sockU = socket(AF_INET, SOCK_STREAM, 0);

// Configuration de la connexion
sinU.sin_addr.s_addr = inet_addr(ADDRIP);
sinU.sin_family = AF_INET;
sinU.sin_port = htons(PORT1);

// Si le client arrive à se connecter
if(connect(sockU, (SOCKADDR*)&sinU, sizeof(sinU)) != SOCKET_ERROR)
{
    printf("Connexion du client utilitaire à %s sur le port %d\n", inet_ntoa(sinU.sin_addr), htons(sinU.sin_port));
}
```

Récupération de l'état d'urgence et Autorisation de la connexion.

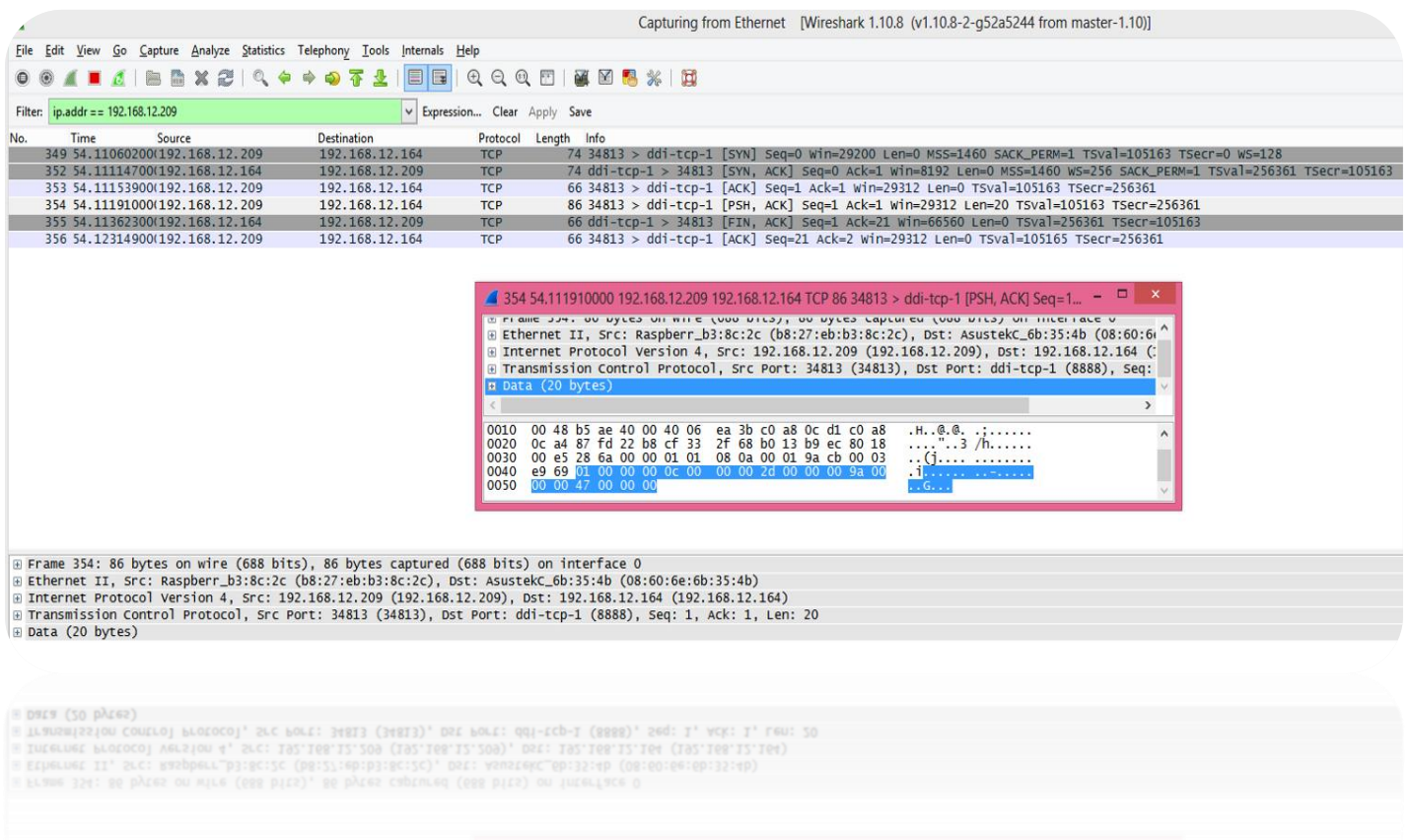
```
//Recevoir reponse du serveur utilitaire
if( recv(sockU , Socket::Recu, sizeof(Socket::Recu) , 0) < 0)
{
    puts("Recv failed");
}

printf("Serveur utilitaire : %s\n", Socket::Recu);
```

On attend la réponse du serveur (Socket::Recu)

WireShark

Je lance WireShark pour vérifier que l'ID Badge est bien envoyé



Capturing from Ethernet [Wireshark 1.10.8 (v1.10.8-2-g52a5244 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `ip.addr == 192.168.12.209` Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
349	54.110602000	192.168.12.209	192.168.12.164	TCP	74	34813 > ddi-tcp-1 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=105163 TSecr=0 WS=128
352	54.111147000	192.168.12.164	192.168.12.209	TCP	74	ddi-tcp-1 > 34813 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=256361 TSecr=105163
353	54.111539000	192.168.12.209	192.168.12.164	TCP	66	34813 > ddi-tcp-1 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=105163 TSecr=256361
354	54.111910000	192.168.12.209	192.168.12.164	TCP	86	34813 > ddi-tcp-1 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=20 TSval=105163 TSecr=256361
355	54.113623000	192.168.12.164	192.168.12.209	TCP	66	ddi-tcp-1 > 34813 [FIN, ACK] Seq=1 Ack=21 Win=66560 Len=0 TSval=256361 TSecr=105163
356	54.123149000	192.168.12.209	192.168.12.164	TCP	66	34813 > ddi-tcp-1 [ACK] Seq=21 Ack=2 Win=29312 Len=0 TSval=105165 TSecr=256361

354 54.111910000 192.168.12.209 192.168.12.164 TCP 86 34813 > ddi-tcp-1 [PSH, ACK] Seq=1...

Frame 354: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0

Ethernet II, Src: Raspberr_b3:8c:2c (b8:27:eb:b3:8c:2c), Dst: AsustekC_6b:35:4b (08:60:6e:6b:35:4b)

Internet Protocol Version 4, Src: 192.168.12.209 (192.168.12.209), Dst: 192.168.12.164 (192.168.12.164)

Transmission Control Protocol, Src Port: 34813 (34813), Dst Port: ddi-tcp-1 (8888), Seq: 1, Ack: 1, Len: 20

Data (20 bytes)

```

0010 00 48 b5 ae 40 00 40 06 ea 3b c0 a8 0c d1 c0 a8 ..H..@..:.....
0020 0c a4 87 fd 22 b8 cf 33 2f 68 b0 13 b9 ec 80 18 ....3/h.....
0030 00 e5 28 6a 00 00 01 01 08 0a 00 01 9a cb 00 03 ..(j).....
0040 e9 69 01 00 00 0c 00 00 00 2d 00 00 00 9a 00 .i.....
0050 00 00 47 00 00 00 ..G....

```

Frame 354: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0

Ethernet II, Src: Raspberr_b3:8c:2c (b8:27:eb:b3:8c:2c), Dst: AsustekC_6b:35:4b (08:60:6e:6b:35:4b)

Internet Protocol Version 4, Src: 192.168.12.209 (192.168.12.209), Dst: 192.168.12.164 (192.168.12.164)

Transmission Control Protocol, Src Port: 34813 (34813), Dst Port: ddi-tcp-1 (8888), Seq: 1, Ack: 1, Len: 20

Data (20 bytes)

0010 00 48 b5 ae 40 00 40 06 ea 3b c0 a8 0c d1 c0 a8 ..H..@..:.....

0020 0c a4 87 fd 22 b8 cf 33 2f 68 b0 13 b9 ec 80 183/h.....

0030 00 e5 28 6a 00 00 01 01 08 0a 00 01 9a cb 00 03 ..(j).....

0040 e9 69 01 00 00 0c 00 00 00 2d 00 00 00 9a 00 .i.....

0050 00 00 47 00 00 00 ..G....

Annexe

```
#ifndef LECTURERFID_H
#define LECTURERFID_H
#define TAILLENUM 5

class LectureRFID
{
public:
    char IDbadge[TAILLENUM];
    LectureRFID();
    char* getIDBadge();

protected:
private:
    char Byte;
    int PositionTrame, PositionIDbadge;
};

#endif // LECTURERFID_H
```

```
#include "LecteurRFID.h"
#include <iostream>
#include <unistd.h> // UNIX standard function definitions
#include <fcntl.h> // File control definitions
#include <errno.h> // Error number definitions
#include <termios.h> // POSIX terminal control definitions
#include <stdio.h>
#include <stdlib.h>

LectureRFID::LectureRFID()
{
    //ouverture driver /dev/ttyUSB0 O_RDWR
    int USB = open("/dev/ttyACM0", O_RDWR);
    if(USB == -1)
    {
        perror("\n Ouverture impossible !! ");
        close(USB);
    }

    termios config;

    PositionIDbadge=0;

    //acquisition de la configuration par défaut
    tcgetattr(USB, &config);
    //9600 bauds 8 bits de données
    config.c_cflag = B38400 | CS8 | CLOCAL | CREAD;
    config.c_lflag = 0;
    config.c_oflag = 0;
    //pas de parité
    config.c_iflag = IGNPAR;
    //mode non canonique
    config.c_iflag &= ~(ICANON);
    //minuteur = 0
    config.c_cc[VTIME] = 0;
    //read retourne dès la réception d'un caractère
    config.c_cc[VMIN] = 1;
    //mise à jour de la configuration
    tcsetattr(USB, TCSANOW, &config);
}
```

```

tcflush(USB , TCIOFLUSH);

printf("Passez votre carte\n");

for(PositionTrame=0;PositionTrame<9;PositionTrame++)
{
    if (read(USB,&Byte,1)>0)
    {
        printf("%x ",Byte);
        if(PositionTrame>=4)
        {
            IDbadge[PositionIDbadge] = Byte;
            PositionIDbadge++;
        }
        } // if new data is available on the serial port, print it out
    }

printf("\nTaille de L'ID badge: %d\n",sizeof(IDbadge));

printf("L'ID du badge est :");

for (PositionIDbadge=0; PositionIDbadge<5;PositionIDbadge++)
{
    printf(" %x ", IDbadge[PositionIDbadge]);
}

printf("\n");

close(USB);
}

char * LectureRFID::getIDBadge()
{
    return &(LectureRFID::IDbadge[0]);
}

```

```
#ifndef EcranLCD_H
#define EcranLCD_H

class EcranLCD
{
public:
    EcranLCD();
    void Initialiser();
    void AfterBadge(char *);
    void Utilitaire(char *);
    int getEtat();
    char* getUrgence();

protected:
private:
    int Etat;
    char Ordre[2];
    char Urgence[3];
    char Mode[2];
};

#endif // EcranLCD
```

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "pifacecad.h"
#include "mcp23s17.h"
#include "EcranLCD.h"

EcranLCD::EcranLCD()
{
    Initialiser();
}

void EcranLCD::Initialiser()
{
    Etat=0;
    Urgence[0]='?';
    Urgence[1]='F';
    Urgence[2]='U';

    pifacecad_open();
    pifacecad_lcd_blink_off();
    pifacecad_lcd_cursor_off();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    pifacecad_lcd_write("Fermer\n");
}

void EcranLCD::AfterBadge(char * PRep)
{
    for (int i=0;i<2;i++)
    {
        Ordre[i]=*PRep;
        PRep++;
    }
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        if(Ordre[0] == '?' && Ordre[1]=='A')
        {
```

```

        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("Autoriser\n");
        pifacecad_lcd_write("Ouvert\n");
        sleep(5);
        pifacecad_lcd_backlight_off();
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");

        pifacecad_lcd_write("Fermer\n");
    }

    if(Ordre[0] == '?' && Ordre[1]=='R')
    {
        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("Refuser\n");
        pifacecad_lcd_write("Fermer\n");
        sleep(5);
        pifacecad_lcd_backlight_off();
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");
        pifacecad_lcd_write("Fermer\n");
    }
}

void EcranLCD::Utilitaire(char * PRecu)
{
    char temp[3];
    for (int i=0;i<3;i++)
    {
        temp[i]=*PRecu;
        PRecu++;
    }
}

```

```
if((temp[1]=='E') || (temp[1]=='S'))
{
    Mode[0]=temp[0];
    Mode[1]=temp[1];
}

if((temp[0]=='?' && temp[1]=='D') || (temp[1]=='F' && temp[2]=='U'))
{
    for (int i=0;i<3;i++)
    {
        Urgence[i]=temp[i];
    }
    if(Urgence[0] == '?' && Urgence[1]=='D' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("Urgence\n");
        pifacecad_lcd_write("Ouvert\n");
    }

    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        pifacecad_lcd_backlight_on();
        pifacecad_lcd_write("FinUrgence\n");
        sleep(2);
        pifacecad_lcd_backlight_off();
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");
        pifacecad_lcd_write("Fermer\n");
    }
}
if(Mode[0] == '?' && Mode[1]=='E')
{
    Etat=0;
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
```



```
{
    pifacecad_lcd_clear();
    if(Etat==0)
        pifacecad_lcd_write("Mode Entre\n");
    else
        pifacecad_lcd_write("Mode Sortie\n");
    pifacecad_lcd_write("Fermer\n");
}
}
if(Mode[0] == '?' && Mode[1]=='S')
{
    Etat=1;
    if(Urgence[0] == '?' && Urgence[1]=='F' && Urgence[2]=='U')
    {
        pifacecad_lcd_clear();
        if(Etat==0)
            pifacecad_lcd_write("Mode Entre\n");
        else
            pifacecad_lcd_write("Mode Sortie\n");
        pifacecad_lcd_write("Fermer\n");
    }
}
printf("Mode: %s\n", Mode);

int EcranLCD::getEtat()
{
    return(EcranLCD::Etat);
}

char* EcranLCD::getUrgence()
{
    return &(EcranLCD::Urgence[0]);
}
```

```
#ifndef SOCKET_H
#define SOCKET_H

#include "LecteurRFID.h"
#include "EcranLCD.h"

class Socket
{
public:
    Socket (LectureRFID Lecture, EcranLCD Ecran);
    Socket ();
    char* getReponse ();
    char* getRecu ();
    void RecuperationID(char*);
    void RecuperationEtat(int);
protected:
private:
    char Message[TAILLENUM];
    char Reponse[2];
    char Recu[3];
};

#endif // SOCKET_H
```

```
#include "Socket.h"
#include "LecteurRFID.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h> //strlen
#include <string> //string
#include <iostream> //cout
#include <cstdlib>

#define SOCKET_ERROR -1
#define ADDRIP "192.168.12.164"
#define PORT 8888
#define PORT1 8888

typedef int SOCKET;
typedef struct sockaddr_in SOCKADDR_IN;
typedef struct sockaddr SOCKADDR;

Socket::Socket(LectureRFID Lecture, EcranLCD Ecran)
{
    using namespace std;

    SOCKET sock;
    SOCKADDR_IN sin;

    // Cr ation de la socket
    sock = socket(AF_INET, SOCK_STREAM, 0);

    // Configuration de la connexion
    sin.sin_addr.s_addr = inet_addr(ADDRIP);
    sin.sin_family = AF_INET;
    sin.sin_port = htons(PORT);

    char* p = Ecran.getUrgence();
    char urq[3];
```

```
for (int i=0;i<3;i++)
{
    urg[i]=*p;
    p++;
}

if(urg[0]=='?' && urg[1] == 'F' && urg[2] =='U')
{
    printf("%'urgence est desactivée : %s \nConnexion autorisée \n",urg);

    // Si le client arrive à se connecter
    if(connect(sock, (SOCKADDR*)&sin, sizeof(sin)) != SOCKET_ERROR)
    {
        printf("Connexion à %s sur le port %d\n", inet_ntoa(sin.sin_addr), htons(sin.sin_port));

        char *myMessage = Lecture.getIDBadge();

        Socket::RecuperationID(myMessage);

        Socket::RecuperationEtat(Ecran.getEtat());

        //Envoyer Message
        if( send(sock , myMessage , 20 , 0) < 0)
        {
            perror("Send failed \n");
        }
        cout<<"\nData send\n";

        //Recevoir la reponse du serveur
        if( recv(sock , Socket::Reponse , sizeof(Socket::Reponse) , 0) < 0)
        {
            puts("recv failed\n");
        }
        printf("Serveur : %s\n",Socket::Reponse);
    }
    else
    {
        printf("Impossible de se connecter\n");
    }
}
```

```
        // On ferme la socket précédemment ouverte
        close(sock);
    }
}

else
    printf("L'urgence est activée : %s \nConnexion Refusée \n",urg);
}

void Socket::RecuperationID(char* myMessage)
{
    for (int i=0;i<TAILLENUM;i++)
        printf("%x ",myMessage[i]);
}

void Socket::RecuperationEtat(int E)
{
    if(E==0)
    {
        Message[0]='?';
        Message[1]='E';
    }

    if(E==1)
    {
        Message[0]='?';
        Message[1]='S';
    }
}

char* Socket::getReponse()
{
    return &(Socket::Reponse[0]);
}

char* Socket::getRecu()
{

```

```
        return &(Socket::Recu[0]);
    }

Socket::Socket()
{
    using namespace std;

    SOCKET sockU;
    SOCKADDR_IN sinU;

    // Création de la socket
    sockU = socket(AF_INET, SOCK_STREAM, 0);

    // Configuration de la connexion
    sinU.sin_addr.s_addr = inet_addr(ADDRIP);
    sinU.sin_family = AF_INET;
    sinU.sin_port = htons(PORT1);

    // Si le client arrive à se connecter
    if(connect(sockU, (SOCKADDR*)&sinU, sizeof(sinU)) != SOCKET_ERROR)
    {
        printf("Connexion du client utilitaire à %s sur le port %d\n", inet_ntoa(sinU.sin_addr), htons(sinU.sin_port));

        //Recevoir reponses du serveur utilitaire
        if( recv(sockU , Socket::Recu, sizeof(Socket::Recu) , 0) < 0)
        {
            puts("recv failed");
        }

        printf("Serveur utilitaire : %g\n",Socket::Recu);
    }
    else
        printf("Impossible de se connecter\n");
}
```