



RAPPORT DE PROJET DOSIMETRE

Riehl Alexandre



Lycée des Métiers
Paul Emile Victor

Lycée
COUFFIGNAL

26 MAI 2016

TABLE DES MATIERES

1. Contexte	2
2. Matériels et logiciels utilisés	2
3. Interface web.....	3
3.1. Introduction.....	3
3.2. Bibliothèques et langages utilisés	3
3.3. Hébergement et base de données	5
3.4. Arborescence du site	7
3.5. Architecture client-serveur.....	7
3.6. Fonctionnement du site	8
3.7. Conclusion	15
4. Dosimètres et télécommande	16
4.1. Langages utilisés	16
4.2. Spécifications détaillées	16
4.3. Développement commun.....	19
4.4. Développement approfondi du dosimètre.....	26
4.5. Développement approfondi de la télécommande	32
4.7. Conclusion finale	35

1. CONTEXTE

Étant l'étudiant numéro 8, j'ai travaillé en collaboration avec deux autres étudiants EC, l'étudiant 6 chargé de la réalisation matérielle des dosimètres et l'étudiant 7 chargé de la réalisation matérielle de la télécommande.

Conformément au cahier des charges, je me suis chargé de la conception d'une interface web permettant au professeur de gérer la création et l'affectation d'activité à un étudiant qui au préalable a été ajouté dans la base de données.

D'autre part, nous avons modifié la répartition des tâches afin de me confier les tâches de conception logicielle du dosimètre et de la télécommande car elles étaient plus orientées pour l'option Informatique et Réseaux.

2. MATERIELS ET LOGICIELS UTILISÉS

MATERIELS UTILISÉS



PICkit est une famille de programmeurs pour microcontrôleur PIC de Microchip Technology. Ils permettent de programmer les microcontrôleurs et de déboguer les programmes.



Oscilloscope, ceci m'a permis d'observer les échanges effectués au niveau des liaisons séries présentes dans les dosimètres et la télécommande.

LOGICIELS UTILISÉS



MPLAB est un environnement libre de développement pour le développement d'applications embarquées sur PIC et microcontrôleurs dsPIC.

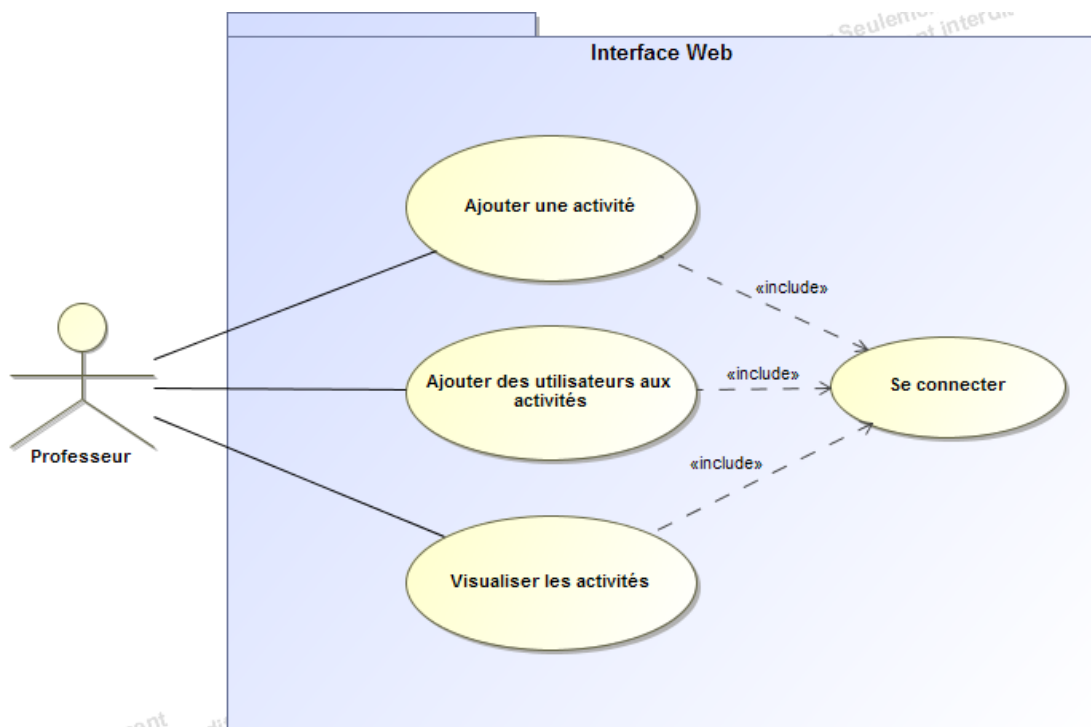


Notepad++ est un éditeur de texte qui intègre la coloration syntaxique de code source, cet éditeur a été utilisé pour le développement de l'interface web en HTML, PHP, CSS et Javascript.

3. INTERFACE WEB

3.1. Introduction

Voici un diagramme de cas d'utilisation, illustrant les besoins attendus par la partie « activité » de l'interface



Pour la gestion des activités des élèves, j'ai dû réaliser une interface web qui permet de créer et d'affecter des utilisateurs à une ou plusieurs activités. J'ai d'autre part dû générer une fiche individuelle composée d'un code-barres qui est utilisé lors de l'entrée en zone contrôlée.

3.2. Bibliothèques et langages utilisés

LES BIBLIOTHÈQUES UTILISÉES

Pour l'aspect graphique du site, nous avons décidé d'utiliser un environnement de travail du nom de « Materialize », cet environnement de travail rempli des fonctions élémentaires nous permettant de gagner du temps et de produire une interface ergonomique et agréable à voir.



Fonctionnalités intéressantes :

- Composants tels que des **barres de navigation**, des **boutons**, les **formulaires**
- Un grand choix de couleurs classé par nom, des couleurs dits « flat »
- Des composants **javascript** tels que des **boîtes de dialogues**
- L'environnement permet de produire facilement un contenu dit « responsive », c'est-à-dire adaptatif aux différentes tailles d'écrans/mobiles.



La librairie FPDF a été utilisée pour la génération de PDF lors de l'affichage de la fiche individuelle, cette librairie ainsi que son utilisation sera expliqué davantage dans la partie conception.

LES LANGAGES UTILISÉS



HyperText Markup Language 5 est un langage de balisage permettant de représenter la structure hypertexte d'une page web.



Cascading style sheets ou Feuille de style en cascade est un langage qui décrit la présentation des documents HTML. (Les couleurs, le placement, la taille ect...).



Le **jQuery** est une bibliothèque codé en javascript, le javascript est un langage qui permet de dynamiser une page web cote client. Le jquery a été créé dans le but de faciliter l'écriture du script qui pouvait s'avérer être assez fastidieux en javascript.



Hypertext Preprocessor est un langage de programmation orienté objet permettant de produire des pages web dynamique communiquant avec un serveur HTTP.



MySQL est un système de gestion de bases de données permettant comme son nom l'indique de gérer une base de données à l'aide de requêtes.



Ajax n'est pas un langage mais une architecture qui permet entre autre de communiquer de façon asynchrone avec le serveur. Cette architecture sera expliquée plus en détail dans la partie conception.

3.3. Hébergement et base de données

HEBERGEMENT

Le site est accessible par internet, ainsi le professeur peut gérer les utilisateurs, les activités et les batteries sur une seule interface accessible depuis n'importe quels périphériques connectés à internet.

Pour le développement nous avons choisis un hébergement gratuit du nom de « Chez », il permet d'héberger un site web compatible avec PHP5 et MySQL. Le gros avantage de ce dernier c'est qu'il est gratuit.

Le site est accessible à l'adresse suivante : <http://www.dosibtssncouf.chez.com>

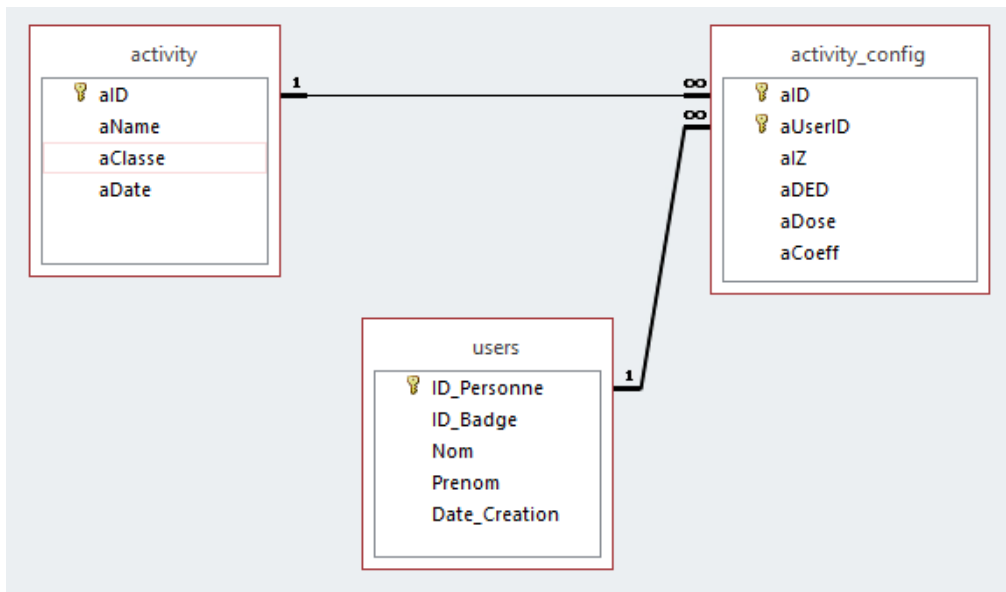
BASE DE DONNÉES

La base de données se nomme « dosibtssncouf », les tables que j'ai créées sont les suivantes :

- activity
- activity_config

En plus de ces tables j'utilise la table « users » créée par l'étudiant 2 car les activités sont liés aux élèves qui sont enregistrés dans la table « users ».

Ci-dessous est représenté le contenu des trois tables et les relations qu'elles ont entre-elles, j'ai réalisé ce schéma relationnelle avec « Access ».



Connexion à la base de données en PHP

Nous avons un fichier mysql.php qui comprend deux fonctions permettant la connexion et la déconnexion à la base de données.

```

function enableConnection($host="localhost", $dbname="dosibtssncouf", $username="dosibtssncouf", $password="PkFqfefJWK")
{
    try {
        $dbh = new PDO('mysql:host='.$host.';dbname='.$dbname.'', $username, $password);
        return $dbh;
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

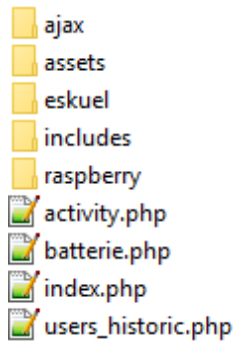
function disableConnection($dbh)
{
    $dbh = null;
}
  
```

CONVENTION DE NOMMAGE DANS LA BASE DE DONNÉES

Concernant le nommage des champs dans la table **activity** et la table **activity_config**, j'ai pris le choix d'utiliser le préfixe « a » pour « activité », ceci dans le but de garder une certaine cohérence et pour retenir plus facilement les noms des champs.

3.4. Arborescence du site

Le site a été organisé de façon à pouvoir travailler chacun sur sa partie sans interférer sur le travail de l'autre.

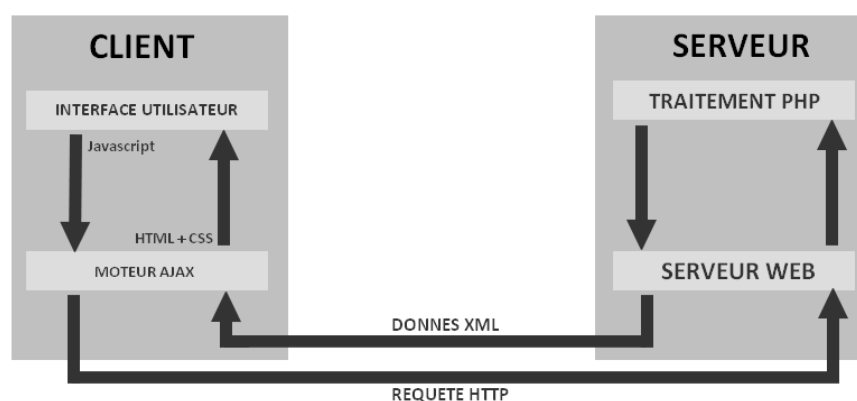


- Le dossier « ajax » contient les pages php pour le traitement (Ajout d'utilisateurs, ajout d'activité...) des requêtes Ajax.
- Le dossier « assets » contient les images, les fichiers javascript, les feuilles de styles et les polices d'écritures.
- Le dossier « eskuel » est externe au site, c'est une application qui nous est utile pour consulter et modifier la base de données, au même titre que PHPMyAdmin
- Le dossier « includes » contient les fichiers qui sont amenés à être inclus dans nos pages, comme l'entête du site, le pied de pages, la fonction de connexion MySQL et diverses autres librairies qui seront explicités par la suite.
- Le dossier « raspberry » contient des pages qui sont uniquement utilisé par le raspberry, ces pages permettent de récupérer des informations de la base de données avec des requêtes HTTP de type GET.

3.5. Architecture client-serveur

J'ai décidé d'utiliser l'architecture Ajax pour l'exécution de nos requêtes HTTP. Avec l'architecture Ajax, les requêtes sont envoyées de façon asynchrone et la réponse du serveur n'implique pas de rechargement de page. Contrairement à la méthode classique qui est synchrone, à chaque envoi de données le client attend la réponse et une fois qu'elle arrive cela implique un rechargement de page avec les nouvelles données HTML.

SCHEMA EXPLICATIF AVEC AJAX



Pour résumé l'utilisateur entre en interaction avec le site via des événements détectés en Javascript, suite à ces événements une requête HTTP est envoyée au serveur avec Ajax et la réponse est affichée sur la partie de la page concernée.

3.6. Fonctionnement du site

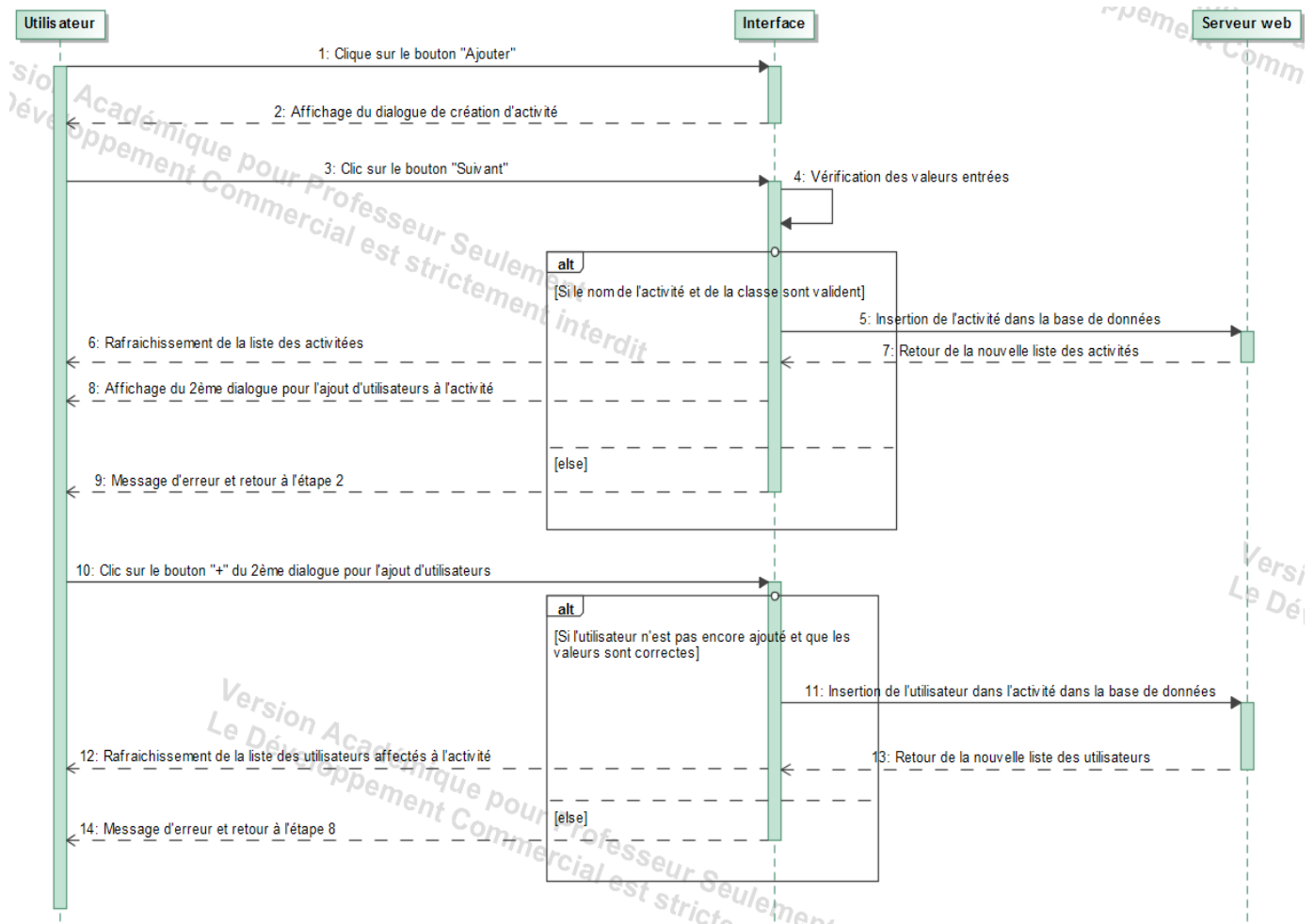
CHARGEMENT DE LA PAGE

Lors du chargement de la page deux requêtes Ajax sont envoyées. Ces requêtes ont pour but d'exécuter du code PHP afin de récupérer la liste des activités dans la base de données, le code ci-dessous est la requête Ajax pour l'affichage de la liste des activités.

```
$( document ).ready(function() // Lorsque la page est chargée
{
    $.ajax({ // Création de la requête HTTP
        method: "POST", // Requête de type POST
        url: "../ajax/ajax_activity.php", // URL cible vers lequel la requête va être envoyée
        data: {type: "activity"} // Données transmises lors de cette requête
    })
    .done (function(msg){ // Lorsque la requête est exécutée et que la réponse est arrivée
        $("#table_activity").html(msg) ;
    });
})
```

AJOUT D'UNE ACTIVITÉ

Afin d'expliquer au mieux le fonctionnement de la **partie activité** de l'interface, vous trouvez ci-dessous un **diagramme de séquence** pour un **scénario d'ajout d'activité** dans le cas où l'utilisateur est déjà authentifié.



Comme le décrit le diagramme de séquence ci-dessus, l'ajout d'une activité est divisé en deux parties.

La première partie est l'ouverture d'un dialogue lorsque l'utilisateur clique sur le bouton « Ajouter ».

AJOUTER

Nom de l'activité	Classe
TP3	TS2SN

SUIVANT

L'utilisateur doit choisir le nom de l'activité et la classe auquel cette activité sera affectée, conformément aux demandes du client.

Une requête Ajax est encore une fois envoyée vers le fichier « ajax_activity.php » et voici le traitement de cette requête dans le fichier de traitement PHP :

```
if(isset($_POST["type"]) && $_POST["type"] == "add_act")
{
    $query = $db->prepare("INSERT INTO activity (aName, aClasse, aDate) VALUES (:name, :classe, NOW())");
    $query->execute(array(
        "name" => $_POST["name"],
        "classe" => $_POST["classe"]));
    echo $db->lastInsertId();
}
```

Toutes les requêtes concernant les activités sont envoyées sur cette page, les différentes requêtes sont différenciées avec une variable transmise dans la requête Ajax qui définit le type de requête.

Dans le cas présent le type est « add_act », ce qui signifie que c'est une requête d'ajout d'activité.

Nous exécutons donc une requête de type « prepare » afin d'ajouter une nouvelle ligne dans la table activity.

Il existe plusieurs types de requêtes dans la classe PDO fournit avec le langage PHP pour gérer les bases de données. Parmi-ces requêtes nous en utilisons deux :

- **query** : Exécution classique d'une requête, utile pour les requêtes où l'utilisateur n'a pas la main sur les valeurs transmises.
- **prepare** : La méthode « prepare » permet comme son nom l'indique de préparer une requête, c'est-à-dire de l'analyser, la compiler et l'optimiser. Ce type de requête bloque les injections SQL.

La seconde étape est « optionnelle » car une fois l'activité créée, le dialogue présenté ci-dessous peut être à nouveau ouvert en cliquant sur « Voir » dans le tableau listant les activités.

The screenshot shows a web interface for managing users. At the top, there is a form with a dropdown menu for 'Utilisateur' (currently showing 'Riehl Alexandre'), and three input fields for 'DED' (0.254), 'Dose théorique' (0.12), and 'Coefficient d'exposition' (0.3). A blue circular button with a white plus sign is to the right of these fields. Below the form, the text 'Utilisateurs ajoutés' is displayed. Underneath is a table with the following data:

Utilisateur	DED	Dose théorique	Coefficient d'exposition	RTR	Retirer
Riehl Alexandre	0.254	0.12	0.3	Voir	Retirer

At the bottom right of the interface is a blue button labeled 'FERMER'.

Voici l'étape de modification de l'activité, cette étape consiste à ajouter un utilisateur à l'activité en le sélectionnant dans la liste et en lui affectant des valeurs de débit, de dose et un coefficient d'exposition. Ces valeurs sont utiles pour calculer la dose de la personne et seront expliqués dans la partie concernant les dosimètres.

Pour récupérer la liste des utilisateurs et les afficher dans la liste sélectionnable, j'ai créé la fonction suivante :

```
function GetUsersList()  
{  
    $db = enableConnection();  
    $query = $db->query("SELECT Prenom, Nom, ID_Personne FROM users");  
    $result = $query->fetchAll();  
    $db = NULL;  
    return $result;  
}
```

Cette fonction permet de récupérer tous les utilisateurs et retourne la liste des utilisateurs ainsi que leurs informations dans un tableau **\$result**.

Voici ci-dessous le code d'ajout d'un utilisateur dans une activité, lors du clic sur le bouton « + »

```
if(isset($_POST['type']) && $_POST['type'] == "add_user")  
{  
    srand();  
    $iz = rand (1000, 5000);  
    $iz *= 9999;  
  
    $query = $db->prepare("INSERT INTO activity_config (aID,aUserID,aIZ,aDED,aDose,aCoeff)  
VALUES (:aid, :auserid, :aiz,:aded, :adose, :acoeff)");  
    $query->execute(array(  
        "aid" => $_POST['activityid'],  
        "auserid" => $_POST['userID'],  
        "aiz" => $iz,  
        "aded" => $_POST['ded'],  
        "adose" => $_POST['dose'],  
        "acoeff" => $_POST['coeff']  
    ));  
}
```

Toujours avec une requête préparée pour éviter toute injection dans la base de données car l'utilisateur peut modifier les valeurs utilisés dans la requête.

La variable **\$iz** est un entier générer pseudo-aléatoirement et contenant 8 chiffres, il sera expliqué plus bas.

SUPPRESSION D'ACTIVITEES

Cette partie n'a pas nécessité de diagramme de séquence de par sa simplicité.

Lors du clic sur le bouton « Supprimer tout », une requête Ajax est exécutée vers le fichier `ajax_activity.php` et une requête se charge de tout supprimer.

Suppression d'une ligne du tableau

Nom	Date de création	Classe	Modification	Suppression
TP2	2016-05-22 13:53:09	TS2SN	Voir	Supprimer

Lors du clic sur « Supprimer », tout comme « Voir », il a fallu récupérer une valeur permettant d'identifier l'activité liée à la ligne.

Dans notre base de données « activity » nous avons une clé primaire du nom de « aID ». Lors du chargement/rechargement de la table un attribut du nom de « activityid » était ajouté dans chaque ligne du tableau.

Voici un extrait du code qui affiche le tableau de la liste des activités :

```
$var .= "<td><a href='#' class='modif_act' activityid='". $result['aID'] . "'>Voir</a></td>";
```

Comme vous pouvez le constater, la valeur de l'id de l'activité récupéré en MySQL est affecté à l'attribut « activityid », ainsi lors du clic il est possible de récupérer cette valeur en Javascript et de la transmettre dans les requêtes Ajax pour divers traitement PHP comme la suppression d'une ligne dans la base de données.

GÉNÉRATION DE LA FICHE INDIVIDUELLE

L'élément final de cette partie activité du site a été la génération et l'affichage de la **fiche individuelle**.

Dans cette première version du site et à l'heure où ce rapport est rédigé, la fiche individuelle est uniquement constitué du code barre, sans les informations annexes qui seront ajoutés par la suite.

Premièrement faisons un rappel sur la fiche individuelle

En dernière page du R.T.R (expliqué dans l'introduction de la partie commune) se trouve la fiche individuelle, cette fiche comporte les conditions radiologiques de l'activité ainsi que le code-barres d'accès en zone-contrôlée. L'image ci-dessus illustre un exemple de fiche individuelle :

Fiche individuelle à présenter en entrant en zone

1 Validité du RTR : 23/09/2006 au 23/09/2006

2 Activité : DEP/REP. CALO APG 011 RF pour

Interv. : DEP/REP. CALO APG 011 RF pour visit

Projet : 2P16 VP Tranche 2

Tranche : 2 Local : 2 NB0503

3 Net d'interv. (R) : 2 APG 011 RF

4 N°IZ : 99999999 1 (Code de travail : 316)

5 -- Valeurs prévues --

DED poste de travail : 0.050 mSv/h

Dont neutrons : 0.000 mSv/h

Dose ind. moy. activité : 0.035 mSv

Dose ind. moy. par jour : 0.035 mSv/j

6

1 Dates de validité du RTR

2 libellé de l'intervention (IZ) à laquelle est rattachée l'activité

3 Objet de l'intervention : Repère Fonctionnel, Système Élémentaire, ...

4 N° de l'IZ à taper si le code à barres est illisible

⚠ Ne pas oublier de taper l'indice (dernier chiffre)

5 Les doses et débit d'équivalent de dose prévus pour l'activité

6 Code à barres à flasher pour entrer en zone

Elle est utilisée lorsque le technicien se rend en zone contrôlée. Munit d'un dosimètre et de son badge, il devra ensuite scanner ce code barre afin de charger les informations d'activité, créées au préalable, dans le dosimètre afin que l'appareil réagisse en conséquence (déclenchement d'alarme).



La fiche individuelle se présente pour l'instant sous la forme suivante.

Comment visualiser cette fiche ?

Pour visualiser cette fiche l'utilisateur du site se rendra dans la page activité et ainsi il pourra l'avoir visualisée en format PDF en cliquant sur « Voir » dans la colonne « RTR » dans le dialogue de modification d'activité.

Code de redirection vers la page de création du PDF

```

/*
  Visualisation du RTR
*/
$(".view_rtr").live("click", function(){

  var activityid = $(this).attr("activityid");
  $.ajax({
    method: "POST",
    url: "../ajax/ajax_activity.php",
    data: {type: "view_rtr", activityid : $('#modal_add_users_act').attr("activityid"), userid : $(this).attr("userid")}
  })
  .done (function(msg){

    window.open("../includes/rtr.php?iz="+msg, "blank");

  });

});

if(isset($_POST['type']) && $_POST['type'] == "view_rtr")
{
  $query = $db->prepare("SELECT aIZ FROM activity_config WHERE aID=:aid AND aUserID=:userid");
  $query->execute(array("aid" => $_POST['activityid'], "userid" => $_POST['userid']));
  $result = $query->fetch();
  echo $result['aIZ'];
}

```

La requête HTTP est envoyée vers la page PHP, avec l'id de l'activité et l'id de l'utilisateur une requête SQL récupère le numéro IZ dans la table « activity_config » et ce résultat est renvoyé en réponse à la requête Ajax. Suite à cette réponse l'utilisateur est redirigé vers la page rtr.php avec une requête HTTP de type GET pour transmettre le numéro IZ.

Méthode de génération du PDF et du code barre

Pour générer un code-barres dans un PDF j'ai utilisé la librairie FPDF accompagné d'une classe créée en plus de cette librairie du nom de « barcoder ».

Cette classe est une généralisation de la classe FPDF, qui permet de générer des PDF, en lui apportant les méthodes nécessaires pour générer un code barre sous différent format.

Lors de notre étude préliminaire nous avons choisis un scanner code à barres fonctionnant avec différent format dont le « Code128 » qui est utilisé dans la librairie.

Plus haut nous avons générer un nombre aléatoire composé de 8 chiffres, ce nombre est en fait le numéro IZ présent dans la fiche individuelle de chaque technicien affecté à une activité.

Grâce à ce nombre, qui est unique à chaque utilisateur et à chaque activité, lors de l'entrée en zone contrôlé l'étudiant 2 chargé d'initialisé le dosimètre a pu récupérer les informations relatives à l'activité avec une requête MySQL en ayant au préalable l'id unique de l'utilisateur et le numéro IZ qui est récupéré avec le scanner.

```
require("barcoder.php");

// Instanciation de l'objet pdf
$pdf = new PDF_Code128();

// Initialisation du PDF pour l'affichage de l'activité
$pageSize = array(100,100);
$pdf->AddPage('P', $pageSize);
$pdf->SetFont('Arial','',10);
$pdf->SetTitle('R.T.R '.$_GET['iz']);

// Placement et affichage du code barre
$code=$_GET['iz'];
/*
 *   Argument: x,y,code,width,height
 */
$pdf->Code128(10, 10, $code, 80, 20);
$pdf->Text(10, 35, $code);

// Affichage du PDF
$pdf->Output();
```

La page rtr.php se charge de générer le code barre dans un PDF avec le numéro IZ qui est passé via une requête de type GET dans l'URL.

La méthode « Output » affiche le PDF dans la page PHP.

3.7. Conclusion

La difficulté principale que j'ai rencontrée pour la réalisation de l'interface est la suivante :

La création de l'activité se décompose en deux parties et il fallait pouvoir transmettre l'ID de l'activité qui venait d'être créée d'un dialogue à un autre sans utiliser les variables super-globales `$_SESSION` car les requêtes Ajax semblent être envoyées dans un autre thread, donc dans les fichiers du dossier « ajax » les variables de SESSION de l'utilisateur n'étaient pas accessibles. Pour remédier à cette solution j'ai passé l'id de l'activité dans un attribut HTML.

Cette version peut encore être améliorée notamment au point de vue de la fiche individuelle. Toujours dans l'optique de fournir un système de simulation le plus réaliste possible, à l'avenir la fiche individuelle pourrait être améliorée du point de vue visuel pour que celle-ci ressemble aux fiches individuelles utilisées en milieu professionnel.

4. DOSIMETRES ET TELECOMMANDE

4.1. Langages utilisés



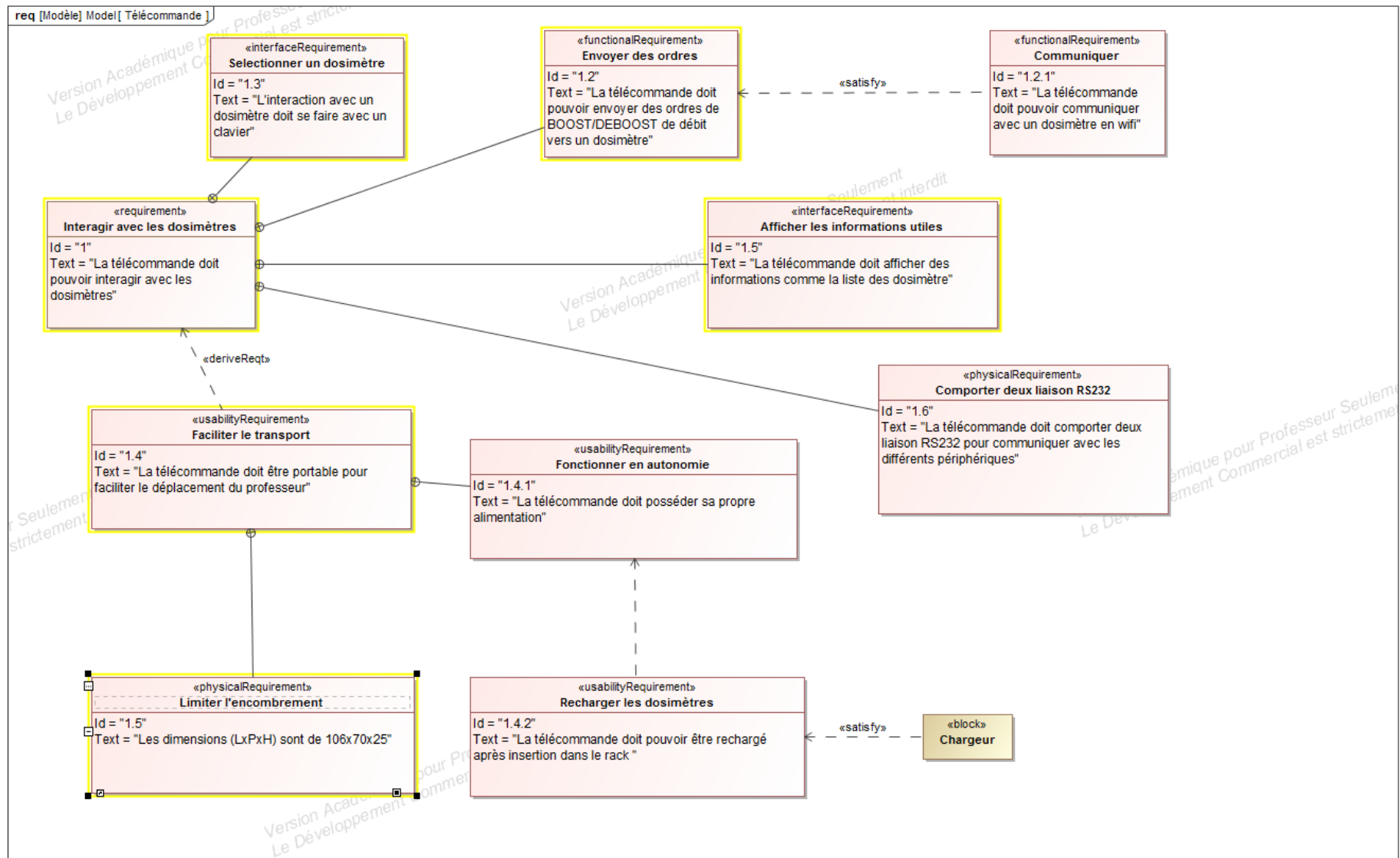
Le langage C est un des langages de programmation les plus connus et utilisés. Dans notre cas nous l'utilisons pour la programmation du PIC33.

4.2. Spécifications détaillées

Ci-dessous vous retrouvez les diagrammes d'exigences détaillés de la télécommande et du dosimètre, ces diagrammes ont été élaborés par le trinôme constitué de l'étudiant 6, l'étudiant 7 et moi-même, l'étudiant 8.

Ils ont permis de définir avec précision les différentes spécifications décrites dans le cahier des charges.

TÉLÉCOMMANDE



4.3. Développement commun

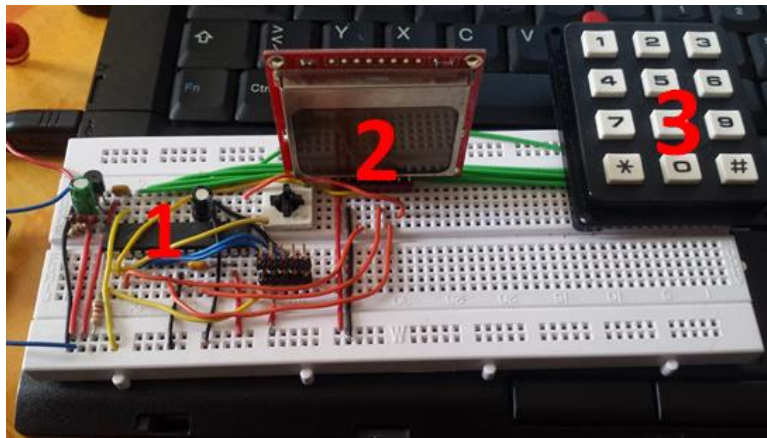
Cette sous partie sera consacré à la partie de développement commun au dosimètre et à la télécommande.

Comme l'illustre les diagrammes d'exigences ci-dessus, la **télécommande** et le **dosimètre partagent des exigences** telles que la communication en série, l'affichage sur un LCD Nokia3310, la communication en Wifi.

J'ai donc procédé à la mise en place d'un **seul programme** comprenant des **fonctions élémentaires** qui seront par la suite intégrées dans le dosimètre et dans la télécommande.

MISE EN PLACE D'UNE MAQUETTE

Avec notre trinôme nous avons dans un premier temps réalisé un petit montage sur une platine d'expérimentation afin que je puisse commencer à mettre en œuvre un début de programme sur le microcontrôleur.



1. Le microcontrôleur utilisé lors des tests : **DSPic33FJ128GP802**.
2. L'afficheur LCD Nokia3310, utilisé dans un premier temps pour le débogage (affichage des données envoyés/reçues).
3. Le clavier matriciel, il est utilisé pour la télécommande.

CONFIGURATION DU PIC

Le microcontrôleur nécessite des réglages avant de pouvoir fonctionner correctement, les réglages ci-dessous ont été faits à l'aide de la documentation du PIC33 et de notre professeur.

Les fonctions ci-dessous se trouvent dans le fichier configure.c présent dans l'annexe.

Configuration de l'horloge

```
void InitCPU_Clock(void)
{
    CLKDIVbits.FRCDIV=0;    // FRC non divisée : 7,37MHz
    CLKDIVbits.PLLPRE=0;    // Prédivision par 2 avant PLL : 3,685MHz
    PLLFBDbits.PLLDIV=40-2; // PLL : multiplication par 40 : 147,4MHz
    CLKDIVbits.PLLPOST=0;   // Division par 2 après PLL : Fosc = 73,7MHz
}
```

Cette fonction est utilisée afin d'initialiser l'horloge du microcontrôleur en fixant une valeur de fréquence et en utilisant le PLL (Boucle à verrouillage de phase), le PLL permet de stabiliser la fréquence.

Configuration du timer

```
void InitTimer1 (void)
{
    PR1=144; //base de temps
    T1CON=0x8030; // TCKPS=10 FcY div 256 ,Timer on, TGATE=0 TCS=0
    T1CONbits.TON=1; //timer on
    IEC0bits.T1IE=1; // validation des interruptions du Timer 1
    TMR1=0;
}
```

Cette fonction est utilisée afin d'initialiser le timer qui dans notre cas s'exécute toutes les 1ms. PR1 correspond à la valeur à laquelle le compteur interne se réinitialise et donc c'est cette valeur qui fait varier le temps de répétition du timer.

La fonction ci-dessous correspond à la fonction d'interruption du timer, c'est cette fonction qui sera appelée à chaque répétition du timer.

Prototype : `void _ISR _T1Interrupt (void)`

Avec ce timer il est possible de faire une fonction de temporisation par exemple, comme c'est le cas avec la fonction `void Delay(int delais)`

MISE EN PLACE DE L’AFFICHEUR LCD

Pour la mise en place de l’afficheur LCD nous avons utilisé une librairie développée par un de nos professeurs et qui a été utilisée lors d’un TP. Cette librairie permet de communiquer avec le contrôleur du LCD pour remplir des fonctions comme l’affichage d’une chaîne de caractère avec retour à la ligne automatique, déplacement du curseur d’écriture, effacement de l’écran...

Test de l’afficheur avec le code suivant

```
void main(void)
{
    InitCPU_Clock();
    InitTimer1();
    AD1PCFGL=0x01FF; // Pas d'entrée analogique

    Fill_Lcd(0); // Effacement de l'écran
    LcdGotoXY(0,0); // Placement du curseur aux coordonnées 0,0
    LcdString("TEST Ecran");
    // Affichage de la chaîne de caractère : "TEST Ecran"

    while(1){}
}
```

Résultat du test



MISE EN PLACE DE LA COMMUNICATION PAR UART

La télécommande et les dosimètres utilisent la communication série pour communiquer avec le module Wifi et pour communiquer avec des périphériques externes tels que le chargeur ou le portique d’identification.

Pour cela nous utilisons les deux UART disponibles sur le microcontrôleur.

Avant toute chose nous avons réalisé deux fonctions permettant d’initialiser l’UART :

```
void InitUART1(long Bauds) et void InitUART2(long Bauds)
```

Extrait de la fonction pour l'UART 1

```
void InitUART1(long Bauds)
{
    U1_Tx; // câblage TxD
    U1_Rx; // câblage RxD

    U1MODE=0x8000; // Module UART validé, 8 bits, pas de parité, 1 bit stop
                  // Pas d'inversion des signaux RX1 et TX1
                  // RTS et CTS non utilisés
    U1STA =0x0400; // UART TX validé
                  // Interruption RX à chaque caractère
    U1BRG = FCY/((long)16*Bauds)-1; // Vitesse de transmission

    _U1RXIP=7; // Priorité maximum car buffer hard de faible capacité
    _U1RXIF=0; // Raz indicateur interruption RX
    _U1RXIE=1; // Validation interruption RX

    ClearUartEspBuffer();
}
```

U1_Tx et U1_Rx sont des macros permettant d'affecter les bonnes valeurs au périphérique d'entrée associé à la patte (RX) et au périphérique de sortie associé à la patte (TX).

Tout comme les timers, nous avons choisis de faire fonctionner l'UART avec des fonctions d'interruption.

Envoi de données

L'envoi de données en liaison série se fait (Pour l'UART1) avec le registre « U1TXREG », à chaque coup d'horloge le caractère présent dans ce registre sera envoyé.

Nous avons réalisé une fonction du nom de `void SendData(char* message)` qui permet d'envoyer une chaîne de caractère avec une boucle « for », chaque caractère de la chaîne est affecté à U1TXREG, envoyé puis le suivant est affecté et envoyé...

Réception de données

La réception de données (Pour l'UART1) se fait avec une fonction d'interruption du nom de :

```
void _ISR _U1RXInterrupt(void)
```

Dans le même principe que l'envoi, la réception se fait à l'aide du registre « U1RXREG », à chaque fois qu'une valeur est présente dans ce registre la fonction d'interruption est appelée et nous récupérons cette valeur dans un tableau global de char afin de pouvoir l'utiliser partout dans le code.

Résultat du test

Pour les tests nous avons relié le PIC au PC via un module de développement usb-série UM232R, ainsi il nous a été possible de vérifier l'envoi et la réception des données transmises du PC vers le PIC via ce petit bout de programme :

```
void main(void)
{
    InitCPU_Clock();
    InitTimer1();
    AD1PCFGL=0x01FF;

    // Initialisation de l'UART1 avec une vitesse de 57600 bauds
    InitUART1(57600);

    Delay(100); // Temporisation pour l'initialisation de l'UART

    SendData("Ceci vient du PIC");

    // Temporisation de 5 secondes en attendant la réponse venant du PC
    Delay(5000);

    // Affichage de la réponse dans le buffer dédié à l'UART1
    Fill_Lcd(0);
    LcdGotoXY(0,0);
    LcdString(espResponse);

    while(1) {}
}
```

MISE EN PLACE DU MODULE WIFI

Comme nous l'avons défini dans l'étude préliminaire, le choix du module pour la communication Wifi se porte vers l'ESP2866.

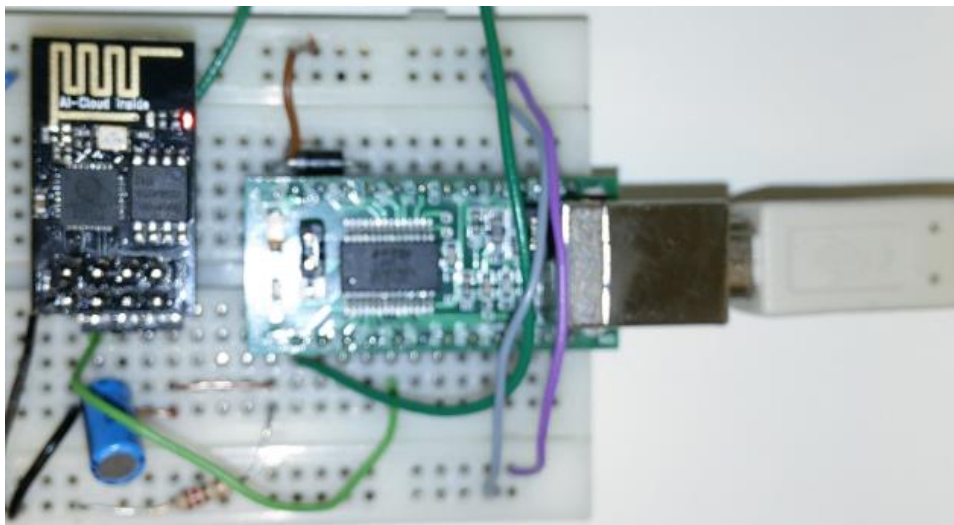
Fonctionnement du module

Afin d'utiliser le module Wifi, nous transmettons des commandes AT du microcontrôleur vers le module, en liaison série. Le module réceptionne la commande, se charge d'effectuer l'action voulue et renvoi un message d'acquiescement si il n'y a pas d'erreur : « OK » pour la plupart des commandes.

Prérequis avant utilisation

De base le module n'est pas implémenté d'un firmware, c'est pour ça qu'il faut en premier lieu installer le firmware dessus, cette procédure est détaillé dans l'annexe.

Nous avons réutilisé le module USB-Série pour cette fois-ci permettre la communication du module Wifi avec le PC.



Avec un logiciel fournit par le lycée, nous avons installé le firmware sur le module afin que les commandes puissent être interprétés. La **procédure d'installation** est **décrite** avec précision dans **l'annexe**.

La liste des commandes AT avec leurs utilisations est fournie dans l'annexe « Commandes AT du module Wifi ».

Résultat des tests

Suite à cette installation nous avons pu mettre en œuvre ces commandes sur le module afin de vérifier le bon fonctionnement.

La commande « **AT** » permet de faire une simple vérification, si le message « OK » est renvoyé c'est que la commande a fonctionné.

```
SDK version:1.1.1
Ai-Thinker Technology Co. Ltd.
Jun 23 2015 23:23:50
```

```
OK
AT+UART_DEF=57600,8,1,0,0
```

```
OK
xÿøþæžþ++ <0x`ðÀ`ø
PORT CLOSED
```

```
PORT OPEN 57600
```

```
Communication with MCU...
Got answer! AutoDetect firmware...

AT-based firmware detected.
AT+GMR
AT version:0.25.0.0(Jun 5 2015 16:27:16)
SDK version:1.1.1
Ai-Thinker Technology Co. Ltd.
Jun 23 2015 23:23:50
```

```
OK
ERROR
AT
OK
```

Ces logs capturés avec le logiciel ESPlorer montre la connexion au module Wifi depuis le PC, en tout dernier la commande « AT » est effectué et nous recevons « OK », ce qui signifie que le module Wifi est opérationnel.

Pour comprendre davantage le fonctionnement du module, nous avons mis en œuvre une seconde platine d'expérimentation afin de tester la communication client-serveur, dans l'exemple ci-dessous nous nous connectons depuis un module Wifi configuré en client à un module Wifi configuré en serveur.

Les aperçu des commandes ont été envoyé avec le logiciel « ESPlorer »

```
AT+CWJAP="DOSI2!", ""
WIFI CONNECTED
WIFI GOT IP

OK
```

Cette commande AT assure la connexion au SSID nommé « DOSI2 ! », la réponse nous informe que la connexion a bien été effectuée.

```
AT+CIPSTART="TCP", "192.168.1.2", 1500
CONNECT

OK
```

Cette commande permet de se connecter au serveur présent dans le SSID ci-dessus, la connexion se fait via le protocole TCP.

RÉPARTITIONS DES IP

Périphériques	IPV4
Télécommande	192.168.1.254
Dosimètre 1	192.168.1.1
Dosimètre 2	192.168.1.2
...	...
Dosimètre 50	192.168.1.50

Nous avons décidé de mettre en œuvre un programme pouvant comporter au maximum 50 dosimètres, ce qui convient largement car une classe de TP est en moyenne composée d'une vingtaine d'élève.

4.4. Développement approfondi du dosimètre

A ce stade du projet les programmes préliminaires sont testés et fonctionnels, il suffit maintenant de les adapter aux exigences du dosimètre.








Le schéma de câblage du dosimètre se trouve en annexe.

Pour résumé le dosimètre doit pouvoir :

1. S'initialiser en entrée de zone
2. Compter de façon classique avec la formule fournie et afficher la dose
3. Produire un BIP en fonction de la dose et du débit
4. Recevoir les ordres de la télécommande
5. Envoyer ses informations d'activité en sortie de zone

ORGANISATIONS DU CODE SOURCE

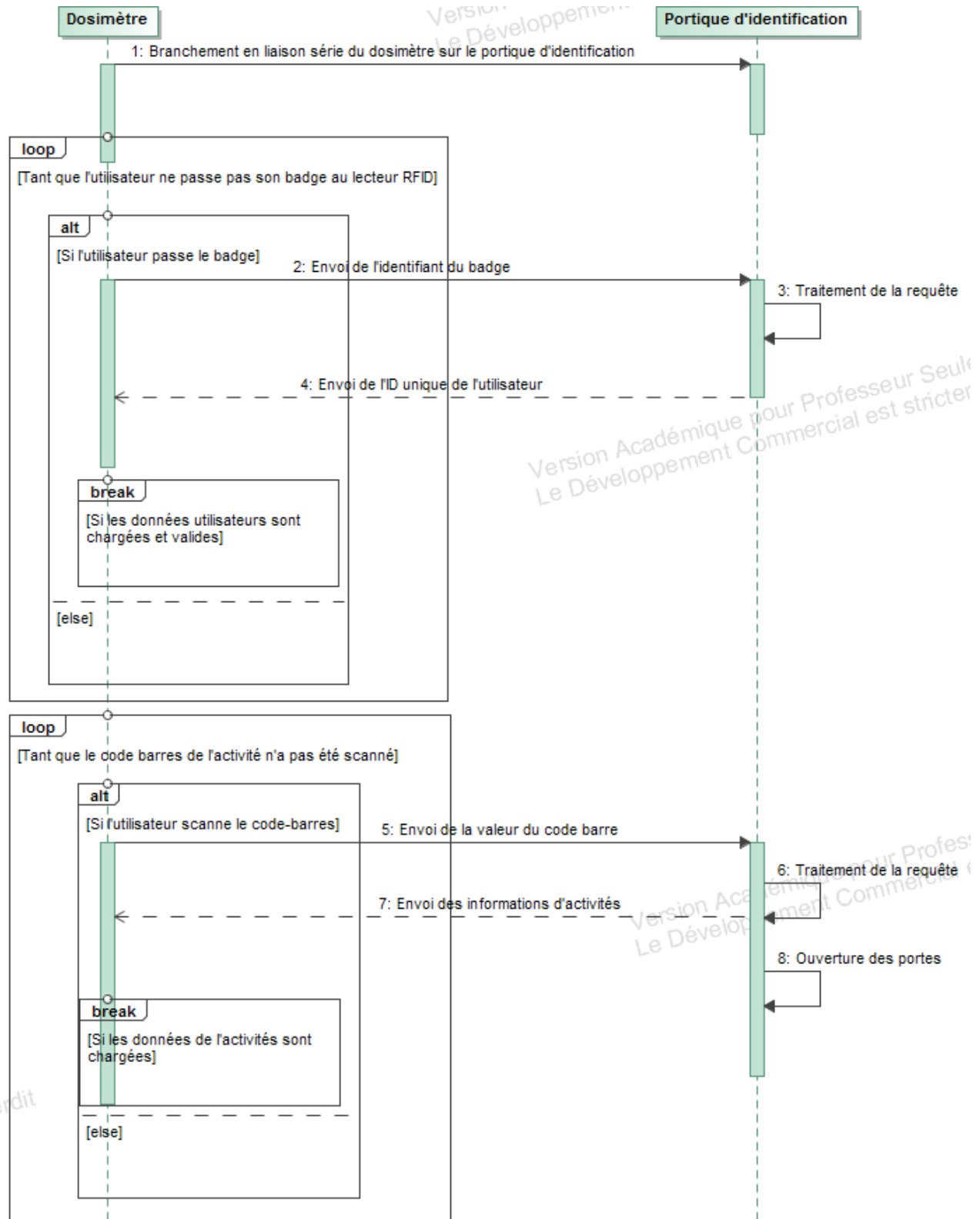
Afin d'avoir un code le plus concis possible, j'ai réparti les différents fichiers par « thème ».

 nokia.c	• nokia.c : Fichier comportant les fonctions de gestion de l'afficheur
 main.c	• main.c : Ce fichier est le point d'entrée du programme, il y a la fonction main
 uart.c	• uart.c : Ce fichier comportant toutes les fonctions concernant l'UART
 configure.c	• configure.c : Ce fichier comprend des fonctions utiles à la configuration du PIC
 wifi.c	• wifi.c : Ce fichier comprend les fonctions concernant le Wifi
 wifi.h	• wifi.h : Ce fichier est le fichier entête du fichier wifi.c
 defines.h	• defines.h : Ce fichier contient des constantes utiles pour le fonctionnement des périphériques du PIC (UART, Afficheur LCD)

GESTION DE L'ENTREE ET DE LA SORTIE DE ZC

Vous trouverez ci-dessous un diagramme de séquence simplifié permettant d'illustrer les étapes mise en œuvre par le dosimètre pour l'entrée et pour la sortie de ZC :

Entrée



Extrait de code pour l'initialisation

```
// On vide le buffer contenant les caractères reçu sur le RX de l'UART2
ClearUartSeqBuffer();

// Tant que la variable contenant l'ID unique n'est pas différente de 0
while(!userID)
{
    /*
    Format d'authentification : x AUTH
    Si dans la chaine de caractère, contenant les caractères reçus sur RX de
    l'UART2, il y a le mot "AUTH"
    */
    if(strstr(seqResponse, "AUTH"))
    {
        // On extrait l'entier de la chaine. Exemple de chaine : "12 AUTH" puis
        // on passe en attente de CODE BARRE
        char *end;
        userID = strtol(seqResponse, &end, 10);
        break;
    }
}
```

En entrée en zone contrôlé, en plus de recevoir l'identifiant unique le dosimètre reçoit l'activité ou le tableau d'initialisation. L'activité, créée au préalable avec l'interface web, est constituée de :

Activité	Explication
Dose	Ceci correspond au seuil maximal de dose que le professeur a fixé pour le technicien
D.E.D	Ceci correspond au débit équivalent dose, c'est la limite de débit que le professeur a fixé pour le technicien
Coefficient d'exposition	C'est un coefficient fixé par le professeur pour le technicien

Sortie

La sortie n'a pas nécessité de diagramme d'exigence pour la partie dosimètre, le code du dosimètre regarde à chaque itérations de la boucle si la valeur de _RA0 est à l'état bas cela signifie que le dosimètre est bien branché, il envoi donc toutes les informations nécessaire.

Ci-dessous vous trouverez les données à transmettre au portique de sortie :

Données en sortie	Explication
ID utilisateur	L'identifiant unique de l'utilisateur
ID activité	L'identifiant de l'activité
Dose	La dose accumulée par le technicien durant l'activité

D.E.D max	Le débit maximal enregistré lors de l'activité
Nombres alarme	Le nombre d'alarme qui se sont déclenchés
Types d'alarme	Le type d'alarmes qui se sont déclenchés

COMPTAGE DE LA DOSE ET AFFICHAGE

Pour le comptage de la dose, conformément à la demande du client j'ai dû respecter cette formule :

$$\text{Dose} = \text{DED} * \text{temps} * \text{coefficient d'exposition} * (1 + \text{Impuls} * 0.05)$$

temps : Temps en heure depuis l'entrée en zone

impuls : Valeur modifiable par la télécommande pour le boost/deboost du dosimètre

Extrait de programme et explications

```
while(1)
{
    // Incrémentation du temps en secondes (Delay de 500ms = 0.5 secondes)

    time+=0.5;
    /*
    Calcul de la dose

    limitDed : D.E.D maximal fixé par le professeur sur l'interface web
    impuls : Valeur de boost gérer par la télécommande du professeur
    coeffExpo : Coefficient d'exposition fixé par le professeur sur l'interface
    time : Temps en heure depuis l'entrée en zone
    */

    userDose += limitDed * (1 + impuls * 0.05) * coeffExpo * (time/3600);

    LcdGotoXY(0,2);
    char string[32] = {0};
    snprintf(string, sizeof(string), "DOSE= %.3fSv", userDose);
    LcdString(string);
    Delay(500);

    Delay(500); // Delay de 500 ms pour le calcul du temps
}
```

Une fois que le dosimètre est initialisé et que le technicien est en ZC, une boucle infinie s'exécute par pas de 500 ms permettant d'effectuer les opérations nécessaires dont le calcul et l'affichage de la dose.

Pour l'affichage de la dose nous utilisons la fonction **snprintf**, cette fonction permet de formater une chaîne de caractère. Dans notre cas nous affichons la dose avec l'identificateur « %.3f », c'est l'identificateur pour les variables de type flottantes et le « 3 » indique au programme de couper la variable afin d'obtenir seulement 3 chiffres après la virgule, conformément à la demande du client.

DÉCLENCHEMENT D'ALARME

Le dosimètre est constitué d'un buzzer piézo-électrique, il est utilisé afin d'alerter le technicien en cas de soucis. Il existe 3 types d'alarmes dont deux concernent la dose et une concerne le D.E.D.

(A l'heure ou ce rapport est rédigé, l'avertissement visuel n'est pas encore intégré)

Type d'alarme	Avertissement sonore	Avertissement visuel	Condition de déclenchement
Pré dose	Son discontinu pendant 20 secondes	Clignotement LED 20s	Dose > 0.05*dose max
Dose	Son continu jusqu'à sortie de ZC	Clignotement LED	Dose > 1.20*dose max
D.E.D	Son discontinu jusqu'à rétablissement du DED	Clignotement LED	DED > 1.20*DED max

Caractéristiques du buzzer

Pour le bon fonctionnement du buzzer nous devons générer un signal carré et l'injecter sur la patte reliée au buzzer.

La fonction InitBuzzer() permet de faire cette opération et d'affecter la valeur de la patte au registre RPOR6.

Extrait de programme et explications

```

if(userDose > (limitDose * 1.20) && !alarmDose)
{
    PlaySound(-1);
}
else if((userDose > (limitDose * 1.05)) && !alarmPreDose)
{
    alarmPreDose = 1;
    PlaySound(20000);
}
else if(limitDed * (1 + impuls * 0.05) > (1.20 * limitDed))
{
    if(!IsAlarmRing())
    {
        if(!alarmPreDed)
        {
            alarmPreDed = 1;
        }
        if(!timeDed) timeDed = time;

        if( (timeDed + 500) > time)
            PlaySound(200),
            timeDed=0;
    }
}

```

Les 3 conditions présentées dans le code ci-dessus représentent les conditions de déclenchement d'alarme présenté dans le tableau ci-dessus.

J'ai réalisé 3 fonctions permettant de faciliter l'utilisation du buzzer :

- PlaySound : Permet d'activer le buzzer pendant n en millisecondes ou indéfiniment (n = -1)
- StopSound : Permet de stopper le son du buzzer
- IsAlarmRing : Permet de vérifier si le buzzer est en route ou non

La fonction PlaySound peut prendre soit une valeur supérieure à 0 pour activer le buzzer pendant un temps déterminé ou la valeur -1 pour un temps indéfini.

La fonction PlaySound fonctionne avec le timer 1 et une variable globale « alarmTimeout » en guise de compteur.

Pour l'alarme sur DED (dernière condition), l'alarme se déclenche pendant 200 ms toutes les 500 ms afin de produire un son discontinu.

RECEVOIR LES ORDRES DE LA TELECOMANDE

Pour la communication en Wifi nous avons décidé de configurer les dosimètres en serveur et ainsi la télécommande qui est configuré en client peut communiquer avec chaque dosimètre.

Le fichier Wifi.c comporte des fonctions permettant de manipuler le module avec les commandes AT fournit en annexe :

- SetWifiMode : Permet de configurer le mode du module (1 client et 2 serveur)
- SetWifiIP : Permet de configurer l'adresse IP du serveur et donc du module
- SetMultiConnection : Permet d'activer les connexions multiplies au serveur
- InitSSID : Permet d'initialiser le SSID en choisissant le nom, le mot de passe, le cryptage ou non et le canal
- StartServer : Permet de lancer le serveur sur le port 1500
- StopServer : Permet de stopper le serveur
- InitWifi : Permet d'initialiser le Wifi lors du lancement du dosimètre.

Identification par SSID

Le module Wifi du dosimètre est identifié et différencié des autres par son SSID, lors de l'appel de la fonction « InitWifi », le SSID est automatiquement affecté au dosimètre de la façon suivante :

```
void InitWifi(int dosiNum)
{
    ClearUartEspBuffer();
```



```

char SSID[8] = {0};
snprintf(SSID, sizeof(SSID), "DOSI%d!", dosiNum);
InitSSID(SSID, "", 1);
Delay(4000);
}

```

L'extrait de code ci-dessus, montre comment est affecté le SSID du dosimètre. Le SSID du 1^{er} dosimètre est par exemple : « DOSI1 ! ».

Pour les ordres la télécommande peut envoyer trois types de valeurs :

- !BOOST : Incrémentement de 1 de la valeur « Impuls »
- !DEBOOST : Décrémentement de 1 de la valeur « Impuls »
- !RAZ : Remise à 0 de la valeur « Impuls »

```

if(strstr(espResponse, "!BOOST"))
{
    ClearUartEspBuffer();
    impuls++;
}
else if(strstr(espResponse, "!DEBOOST"))
{
    ClearUartEspBuffer();
    if((impuls-1) >= 0) impuls--;
}
else if(strstr(espResponse, "!RAZ"))
{
    ClearUartEspBuffer();
    impuls=0;
}

```

Dans la boucle infinie de traitement de la dose, je vérifie à chaque itération ce que contient le buffer RX lié au module Wifi, s'il contient un des 3 mots je réalise l'opération sur la valeur « impuls » qui fait varier le débit d'incrémentement de la dose.

4.5. Développement approfondi de la télécommande

Le schéma de câblage de la télécommande se trouve en annexe.











La procédure d'utilisation de la télécommande est fournie en annexe.

Pour résumé la télécommande doit pouvoir :

1. Doit pouvoir sélectionner un dosimètre à l'aide du clavier
2. Doit pouvoir envoyer des ordres à un dosimètre sélectionné

ORGANISATIONS DU CODE SOURCE

Afin d'avoir un code le plus concis possible, j'ai réparti les différents fichiers par « thème ».

 nokia.c	• nokia.c : Fichier comportant les fonctions de gestion de l'afficheur
 menu.c	• menu.c : Contient le code permettant de faire fonctionner le menu
 main.c	• main.c : Ce fichier est le point d'entrée du programme, il y a la fonction main
 uart.c	• uart.c : Ce fichier comportant toutes les fonctions concernant l'UART
 wifi.c	• configure.c : Ce fichier comprend des fonctions utiles à la configuration du PIC
 configure.c	• clavier.c : Ce fichier comporte les fonctions associées à l'utilisation du clavier
 clavier.c	• wifi.c : Ce fichier comprend les fonctions concernant le Wifi
 defines.h	• wifi.h : Ce fichier est le fichier entête du fichier wifi.c
 wifi.h	• defines.h : Ce fichier contient des constantes utiles pour le fonctionnement des périphériques du PIC (UART, Afficheur LCD)
 menu.h	• menu.h : Ce fichier contient les prototypes des fonctions et les variables utiles pour menu.c

CONNEXION EN WIFI

Tout comme le dosimètre, la télécommande est composé de plusieurs fonctions permettant de gérer le module Wifi. La télécommande étant configurée en client les commandes AT diffèrent et sont présentes en annexe.

Les fonctions sont très proches à l'exception de quelques fonctions qui ont été ajoutées dont :

- ConnectToSSID : Permet la connexion à un SSID à portée
- ConnectToServer : Permet la connexion à un serveur en TCP/IP à condition d'être sur son SSID
- DisconnectFromServer : Permet de se déconnecter d'un serveur
- WaitForConnect : Fonction permettant de vérifier N fois si la connexion a été établie
-

SELECTION DES DOSIMETRES

La sélection des dosimètres se fait à l'aide d'un clavier matricielle 12 touches.

Le fichier clavier.c comporte deux fonctions :

- InitClav : Cette fonction permet de configurer les bits pour l'utilisation du clavier
- Lecture_Clav : Cette fonction retourne la touche enfoncée lors de l'appel, la variable retournée est de type char.

Partie de code de Lecture_Clav

```
char Lecture_Clav(void)
{
    _TRISB12 = 0;
    Delay(20);

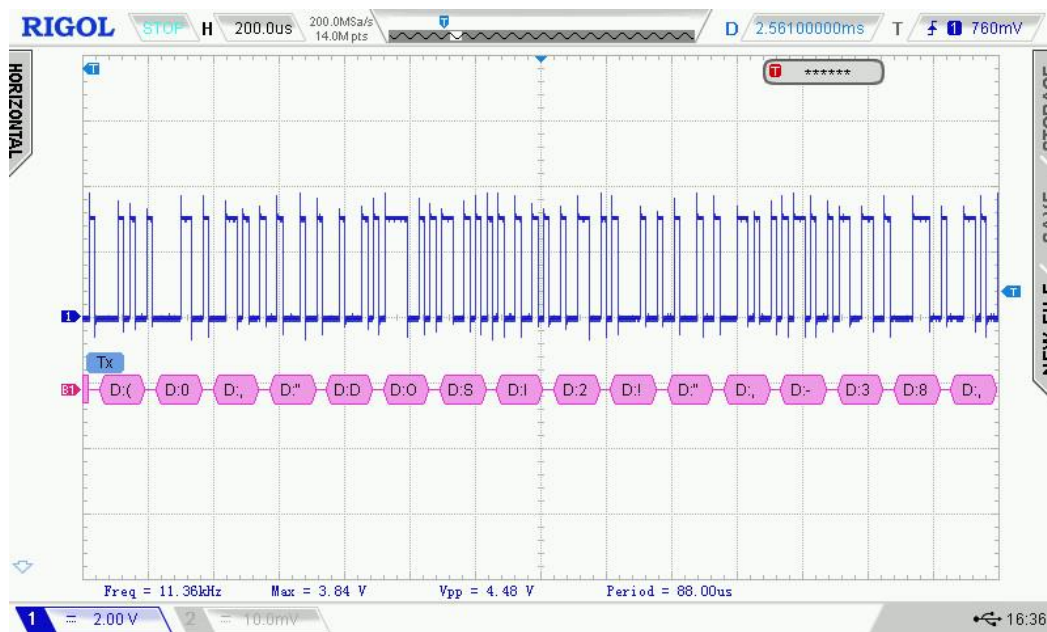
    if(C1==0) { _TRISB12 = 1; return('1'); }
}
```

Dans le cas présenté dessus, la patte RB12 est mise en sortie pour la lecture puis nous regardons les valeurs de C1 (Bit qui correspond à la touche 1), si cette valeur est à l'état bas cela signifie que la touche est enfoncée et donc nous remettons la patte RB12 en entrée et nous retournons le caractère '1'.

Le fichier menu.c comporte trois principales fonctions :

- ShowMenu: Permet d'afficher le menu de base « 1 : Dosimetre »
- ShowDosiListMenu : Cette fonction permet d'afficher la liste des dosimètres détectés par le module Wifi via une commande AT.
- ShowDosiMenu : Cette fonction permet d'afficher le menu de BOOST/DEBOOST/RAZ du dosimètre sélectionné.

Pour afficher la liste des dosimètres nous utilisons la commande « AT+CWLAP », elle permet de récupérer la liste des points d'accès à proximité du module Wifi, voici une capture de la trame :



Parmi la liste des points d'accès nous pouvons voir qu'il détecte un SSID du nom de « DOSI2 ! ».

4.7. Conclusion finale

Contraintes

Les contraintes que j'ai rencontrées pour le développement de la télécommande et du dosimètre ont principalement été présentes au début du projet. Étant habitué à programmer sur des systèmes d'exploitation (OS) tel que Windows ou encore Linux, j'ai eu quelques difficultés à m'habituer à la programmation sur un microcontrôleur notamment au niveau de la liaison série car contrairement aux OS, la programmation sur microcontrôleur est de bas niveaux et cela nécessite des opérations qui sont à l'heure d'aujourd'hui faites sans qu'on le sache. C'est pour cela que j'ai fait en sorte de réaliser un maximum d'automatismes et de surcouches au niveau du programme comme les fonctions concernant le Wifi, cela dans le but de rendre le code le plus compréhensible et modifiable possible.

Ce qui reste à faire et à améliorer

De mon côté le programme est fonctionnel mais peut être amélioré. Concernant la télécommande il reste à faire en sorte de pouvoir sélectionner plusieurs dosimètres afin de leur envoyer des ordres, pour réaliser cette partie il faut aussi prendre en compte la partie ergonomique de l'utilisation de la télécommande qui peut aussi être améliorée. Concernant le dosimètre à l'avenir il serait envisageable d'utiliser un boîtier moins volumineux pour qu'un technicien puisse le mettre dans sa poche et utiliser un autre type d'écran pour le placer au-dessus du boîtier comme sur un dosimètre réel.

Ce que le projet m'a apporté

Ce projet m'a permis de bien comprendre comment mener à bien un gros projet surtout au niveau de l'étude nécessaire avant le commencement du projet. Contrairement à un travail individuel, l'étude préliminaire est essentielle dans un projet de groupe afin de bien définir les tâches, les répartir au mieux pour optimiser au maximum le temps. D'autre part, concernant le développement, je me suis bien amélioré sur le développement sur microcontrôleur.

Remerciement

Je tiens premièrement à remercier nos professeurs monsieur Hentz, monsieur Cremmel et monsieur Lahache pour l'aide et les conseils qu'ils nous ont apportés tout au long de la réalisation de notre projet.

D'autre part je remercie le professeur Briere Thomas, demandeur du projet, pour nous avoir accueillis dans ses locaux. Cette visite nous a permis de bien comprendre la demande et les exigences du cahier des charges.