Dédicaces

A l'âme de mon père

En signe de reconnaissance,
à ma chère mère,
à CyberBox,
à ma femme,
à mon frère & à ma sœur,
et aux personnes que j'ai oublié

Remerciements

Je tiens à remercier :

- **Mr Charfeddine Beskri**, mon encadreur, pour son soutien, ses précieux conseils, son encouragement et tout le temps qu'il m'a consacré.
- **Mr Jalel Kantaoui**, Gérant de Founoun El Kantaoui, qui a accepté de m'accueillir et de m'assigner un projet qui pourrait élargir la gamme de produits offerts par sa société tout en s'adaptant à ma formation académique au sein de l'ISEFC.
- **Dr Kamel Mbarek**, mon parent, qui a volontairement révisé et corrigé ce rapport.

Liste des Abréviations

ASCII: American Standard Code for Information Interchange

CMOS: Complimentary Metal Oxide Semi-conductor

EEPROM: Electrically Erasable Read Only Memory

ESC: Escape character

HEX: Hexadecimal

I_2C : Inter-Integrated Circuit

LCD: Liquid Crystal Display

MCU : Micro Controller Unit (Microcontroller)

MP3 : MPEG 1, Layer III

MPEG : Motion Picture Expert Group

RAM: Random Access Memory

RC5 : Rivest Cipher 5

RTC: Real Time Clock

SCL: Serial Clock Line

SDA: Serial Data Line

SGBD : Système de Gestion de Bases de Données

TTL: Transistor-Transistor Logic

UML : Unified Modelling Language

Sommaire

Liste de Figures

Liste des tableaux

Etude de besoins

1. Introduction - Importance des horaires de prière

Un hadith sharîf rapporté dans les livres intitulés "Mukaddimet Assalât", "Attafsîri Al Madharî" et "Al Halabî Al Kebîr" par Ibn Abbas (Radhia Allah Anhouma) par le prophète (Salla Allah Alyhi wa Sallem) : "**Jabraîl aleihissalâm fut mon imâm pendant deux jours tout au près de la porte de Kaaba. Nous les deux, nous accomplîmes la prière de l'aube du jour avec le commencement du fajr et nous accomplîmes celle du milieu du jour quand le soleil commençait son déclin à partir du zénith; et nous accomplîmes la salât de l'après-midi quand l'ombre des objets est devenue deux fois plus grande qu'eux-mêmes et nous accomplîmes celle du soir quand le soleil se couchait [son bord supérieur disparaissait] et celle de nuit quand le crépuscule a pris fin et l'obscurcissement entier est arrivé. Et le deuxième jour, nous accomplîmes la salât du matin pendant l'aube, celle de midi quand l'ombre des objets est devenue deux fois plus grande qu'eux-mêmes, celle de l'après-midi à la suite de celle-ci, celle du soir quand on cessait de jeûner et celle de la nuit quand le tiers de la nuit est tombée. Puis, il dit: "Ô Muhammed! Les temps des prières pour toi et pour les Prophètes précédents sont ceux-ci. Que ton umma (communauté) accomplisse chacune des prières entre ces deux temps comme nous les fîmes**". Cela s'est passé le lendemain du mi'râj (ascension), deux années avant l'Hégire. A partir de ce jour-là la pratique des prières rituelles cinq fois par jour est devenue une obligation.

Il est fard (obligation) pour tous les musulmans, hommes ou femmes pubères, de pratiquer à l'heure prescrite les prières rituelles cinq fois par jour. Si on effectue une prière de salât avant l'arrivée de son temps, elle ne serait pas valable. Il faut s'acquitter d'une prière de salât à l'heure pour qu'elle soit valable, de même qu'il est fard (obligatoire) d'être au courant du temps correspondant à chaque prière.[1]

[1] Source, http://www.namazvakti.com/fr.1.pdf

Le hadith déjà évoqué indique aux musulmans les horaires de prières avant l'invention de la montre et des outils de navigation modernes. De nos jours les astronomes ont inventés des méthodes de calculs basés sur des formules sophistiquées pour trouver ces horaires d'une façon très précise.

Des centaines d'applications, et des montres spéciales exploitant ces formules sont disponibles pour le public.

2. Etude de cas : Les mosquées en Tunisie

L'appel à la prière (Azan) est diffusé dans les mosquées cinq fois par jour et ce aux horaires spécifiques diffusés par le Ministère des affaires religieuses.

Dans la majorité des mosquées en Tunisie, les horaires de prières sont affichés sur un tableau statique présentant ces horaires (figure 1).

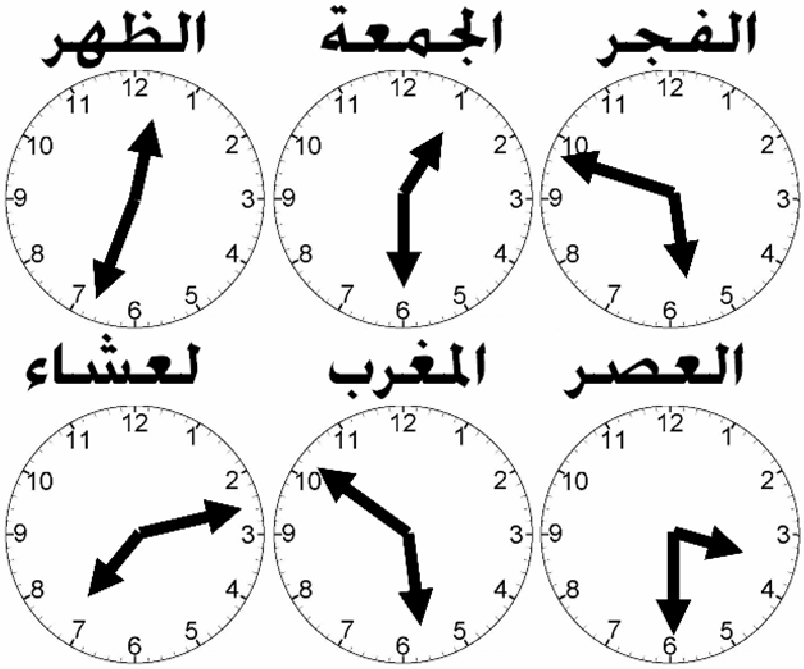


Figure 1, tableau d'affichage des prières

Ce tableau des prières est régulièrement mis à jour, d'une façon manuelle, par le responsable de la mosquée. Toutefois l'absence du responsable (pour une cause quelconque) peut induire en erreur les pratiquants qui peuvent arriver tard parce que ce tableau n'a pas été mis à jour.

Certaines mosquées utilisent des montres électroniques (Made in china) qui permettent de résoudre ces problèmes mais en créant, toutefois, de nouveaux problèmes :

- Lors de la mise hors tension de ces montres il faudra passer par une longue procédure de configuration pour régler l'heure, la date, et la région.
- Aussi, ces nouvelles montres ne sont pas très visibles de loin.

Suite à la révolution tunisienne et à l'inhibition du rôle de la police politique en Tunisie les mosquées sont de plus en plus visitées par les pratiquants, de nouvelles mosquées sont en cours de construction : le besoin d'afficher les horaires de prières dans ces lieux sacrés est donc en hausse.

Nous essayerons dans ce projet de concevoir un nouveau produit qui affiche la date et l'heure, les horaires de prières et diffusant l'Azan aux horaires affichés.

Ce produit sera destiné à l'ensemble des mosquées en Tunisie.

3. Objectif

Concevoir, réaliser et commercialiser un produit sur le marché tunisien destiné spécialement aux mosquées et vendu à prix compétitif.

Ce produit sera réalisé pour le compte de la société **Founoun El Kantaoui** qui se chargera de sa réalisation et de sa commercialisation à grande échelle.

Dans ce rapport il s'agira de réaliser un prototype pour pouvoir évaluer les coûts et le temps nécessaires.

Ce projet a été entamé le 30 Juin 2010 et a été achevé le 15 Septembre 2010.

Introduction

1. Présentation de la société Founoun El kantaoui

Founoun El Kantaoui, sise à Hammam Sousse, est une jeune société, SARL, opérant dans le domaine de l'affichage publicitaire et signalétique. La gamme de produits qu'elle offre à ses clients s'étend des plaques d'immatriculation des voitures aux enseignes lumineuses des magasins, en passant par les plaques signalétiques sur plexiglas et les bandeaux publicitaires sur bâches.

Pour satisfaire les exigences de ses clients, la société compte se lancer dans la conception/fabrication d'enseignes lumineuses à base de diodes LED. Ce nouveau produit requiert des compétences plus grandes et des frais supplémentaires pour embaucher la main d'œuvre qualifiée et acheter la matière première nécessaire.

Afin de valider cette nouvelle idée, le chef de la société a réalisé une étude qui comporte tous les aspects du projet : technique, financier et commercial. L'aspect technique permet d'énumérer les compétences techniques requises pour concevoir et fabriquer le produit. L'aspect financier permet d'estimer le coût du produit final et de chercher les ressources de financement. L'aspect commercial garantira le succès du projet à travers la détermination d'un prix de vente compétitif et le choix adéquat de la clientèle ciblée.

2. Travail Demandé

Réaliser un prototype pour un nouveau produit, destiné aux mosquées, dénommé « Prayer Caller » qui permettra d'afficher l'horaire journalier d'appel aux prières « Azan ».

La tâche m'a été confiée dans le cadre de mon Projet de Fin d'Etude au sein de l'ISEFC.

Cahier de Charges

Le cahier de charges permet de cerner, entre autres, les différentes exigences fonctionnelles et non fonctionnelles du système. Il est plus efficace de rédiger ces exigences selon un modèle tel que « FURPS[2] » qui est retenu dans le cadre de ce projet.

Le modèle FURPS classe les exigences comme suit :

- **Exigences fonctionnelles** résument les principales fonctions offertes par le système.
- **Exigences non fonctionnelles** sont dans l'ordre:
    - d'utilisabilité : efficacité, efficience et satisfaction lors de l'utilisation du produit
    - de fiabilité : exactitude et précision du système, disponibilité, durée de fonctionnement avant défaillance.
    - de performance : temps de réponse, débit, utilisation des ressources
    - de maintenabilité : relatives à la maintenance du système

Il est à signaler que d'autres modèles supportent d'avantages d'exigences non fonctionnelles.

1. Exigences fonctionnelles

Le système, Prayer Caller, doit être capable de :

- Afficher la date, l'horaires des différentes prières
- Afficher le jour de la semaine
- Diffuser l'Azan aux horaires indiqués
- Régler la date et l'heure du système
- Personnaliser l'emplacement géographique du système

[2] Functionnality Usability Reliability Performance Supportability

2. Exigences non fonctionnelles

Le système est destiné aux mosquées qui souhaitent afficher les horaires de prières et diffuser l'Azan. Le produit conçu doit satisfaire les exigences fonctionnelles déjà explicitées. Pour se démarquer de la concurrence, le système à concevoir, doit obéir aux exigences non fonctionnelles décrites ci-dessous.

a. Utilisabilité

Le système doit :

- Etre d'un usage simple, aucun apprentissage n'est nécessaire pour sa mise en route.
- Etre accessible à distance, utilisation de télécommande infrarouge.
- Offrir un affichage clair des informations, utilisation d'afficheur LCD.
- Offrir un affichage visible, depuis plus de cinq mètres et même à faible éclairage, afficheurs 7 segments de grande taille.

b. Fiabilité

Les horaires indiqués par le système doivent être suffisamment précis et aussi proches que possible de la réalité.

Lorsqu'il est mis sous tension, le système doit fonctionner sans surchauffe, ni défaillances.

Le système peut fonctionner sous batterie 7.5V -- 24V.

c. Performances

Le système doit sauvegarder la date, l'horaire de prière et la configuration du système même hors tension.

d. Maintenabilité

Le système doit :

- Etre modulaire, afin de pouvoir cerner l'emplacement de la défaillance.
- Utiliser des composants non onéreux et disponibles sur le marché local.

Conception Modulaire du système

1. Présentation

Lors de la conception et durant les étapes de réalisation de ce Projet, j'ai essayé d'adopter une démarche modulaire progressive. Ce projet sera mené en quatre étapes :

1. Créer une vue d'ensemble du système pour pouvoir cerner les composants du système
2. Passer à la conception et à la réalisation d'un composant du système. Cette étape est divisée en quatre étapes :
    a. **L'étape de conception :** consiste à :
        i. Choisir les composants électroniques
        ii. Consulter les datasheets des différents composants pour connaître leurs propriétés physiques
        iii. Dessiner le schéma de principe
    b. **L'étape de simulation :** permet de valider le schéma de principe
    c. **L'étape d'implémentation & de réalisation :** dans lesquelles il faudra dans l'ordre :
        i. Effectuer le routage du circuit imprimé
        ii. Réaliser du circuit imprimé
        iii. Souder les composants
    d. L'étape de test unitaire :
        i. Implémenter le programme à charger dans les µC
        ii. Tester le bon fonctionnement du composant
3. **Procéder à l'assemblage des modules :** Relier les composants physiquement et logiquement
4. **Programmer le module principal :**
    a. Calcul des horaires de prière
    b. Implémentation du menu de configuration du système

Dans le paragraphe suivant nous donnerons une vue d'ensemble du système. Nous consacrerons par la suite un chapitre pour détailler chaque composant :

1. **Chapitre 1** – Module lecteur MP3
2. **Chapitre 2** – Module afficheur sept segments
3. **Chapitre 3** – Module afficheur LCD série
4. **Chapitre 4** – Mémoire I_2C et Horloge RTC
5. **Chapitre 5** – Commande infrarouge
6. **Chapitre 6** – Module de contrôle, de commande et de configuration

Une application utilitaire sera développée pour permettre de créer les fichiers HEX nécessaires pour initialiser les mémoires du montage.

7. **Chapitre 7** – Application Utilitaire

2. Vue d'ensemble

Chaque rectangle de la vue d'ensemble du système (Figure 2) représente un composant du système qui sera conçu et réalisé par la suite.

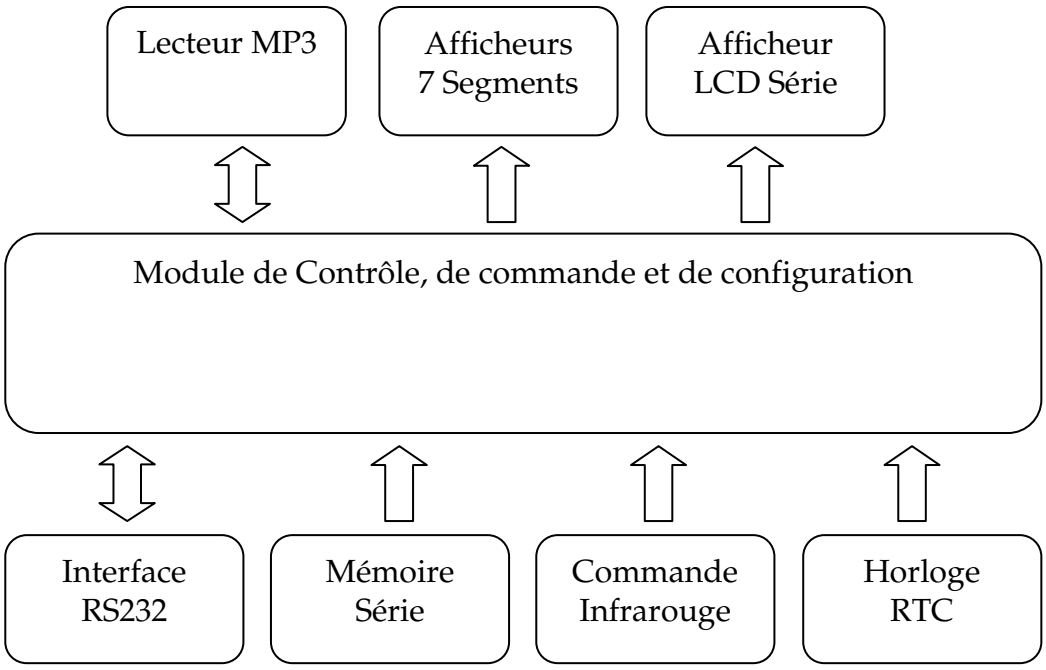


Figure 2, Vue d'ensemble du système

Chapitre 1 - Lecteur MP3

1. Présentation

Le lecteur MP3 (figure 3) utilisé dans cette réalisation est vendu dans le commerce.

Figure 3, Lecteur MP3

Il présente toutes les caractéristiques adéquates pour satisfaire le besoin envisagé :

- **Lecteur MP3 :** Besoin fonctionnel à assurer par Prayer Caller. La lecture d'un fichier est effectuée à partir d'un flash-disc ou d'une mémoire SD.
- **Interface de commande :** Possibilité de commande par Infrarouge (figure 4).
- **Interface de contrôle :** Afficheur sept segments simple (figure 5).

Le démontage du Lecteur MP3 nous a permis de récupérer les modules : d'alimentation, de lecture et les haut-parleurs (figure 4).

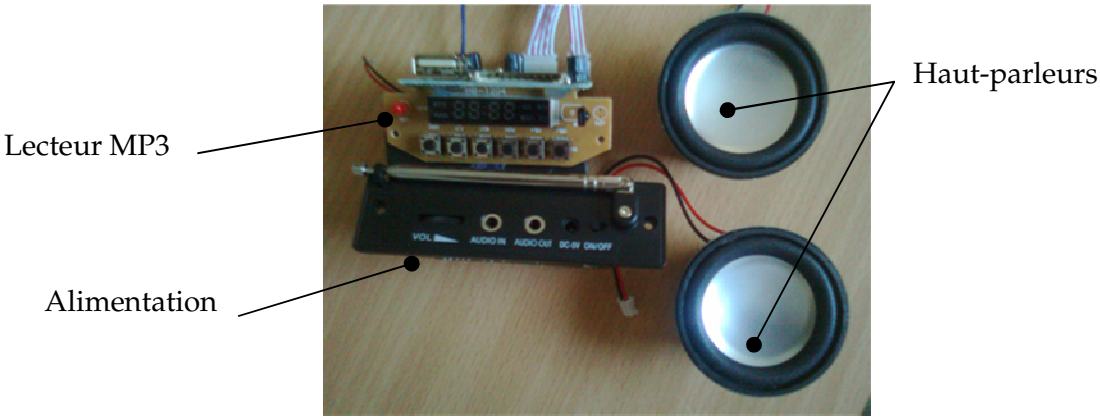


Figure 4, Composants récupérés du lecteur MP3

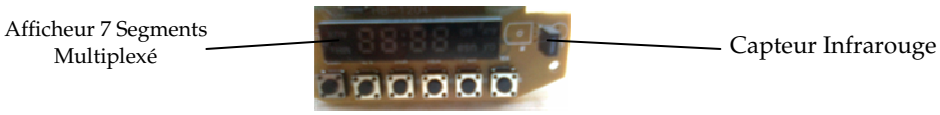


Figure 5, Interfaces de contrôle et de commande du lecteur MP3

a. Interface de commande

Le lecteur MP3 est commandé à l'aide d'une télécommande. Pour pouvoir l'exploiter, il faudrait reproduire les commandes (figure 6) envoyées à partir de la télécommande.

Les chronogrammes des commandes (Figure 5) montrent le signal reçu par le capteur infrarouge du lecteur.

Figure 6, Chronogramme des commandes reçues de la commande du Lecteur MP3

b. Interface de contrôle

L'afficheur 4 chiffres 7 segments du lecteur est un afficheur multiplexé à anode commune (figure 7). Ce dernier est souvent utilisé afin de réduire le nombre de broches du composant et surtout pour minimiser le courant utilisé dans la commande de l'afficheur.

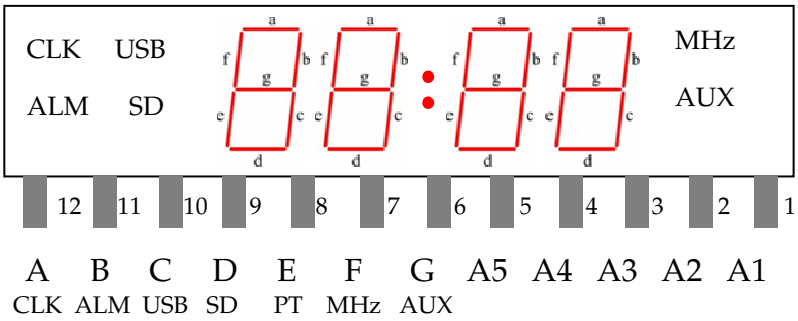


Figure 7, Brochage de l'afficheur du lecteur MP3

L'interface de contrôle peut être réalisée par la lecture continue de l'état actuel des broches de l'afficheur. Pour se faire il faudrait comprendre le fonctionnement multiplexé de cet afficheur (tableau 1, figure 8). Par exemple, pour que l'afficheur montre la valeur « USB 01:38 » le lecteur MP3 change périodiquement l'état des broches de l'afficheur à une fréquence fixe, donc à des intervalles de temps Δt fixes.

Table 1, fonctionnement multiplexé de l'afficheur du lecteur MP3

Instant	Affichage	A	B	C	D	E	F	G	A5	A4	A3	A2	A1
t	USB, :	H	H	L	H	L	H	H	H	L	L	L	L
t+ Δt	0	L	L	L	L	L	L	H	L	H	L	L	L
t+2.Δt	1	L	L	H	H	H	H	H	L	L	H	L	L
t+3.Δt	3	L	L	L	L	H	H	L	L	L	L	H	L
t+4.Δt	8	L	L	L	L	L	L	L	L	L	L	L	H

Le tableau 1 monte l'état des broches de l'afficheur à chaque Δt. Le choix d'un Δt approprié permet d'obtenir un affichage stable.

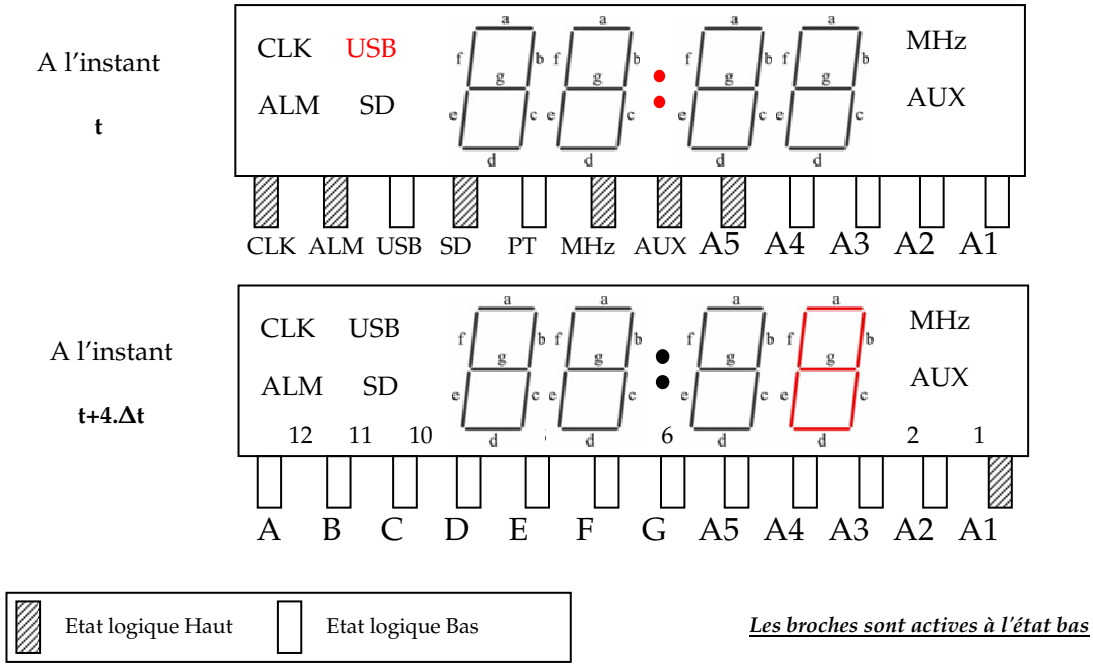


Figure 8, Etat de l'afficheur 7 segments du lecteur MP3

Afin d'assurer le pilotage du lecteur MP3, il faudrait concevoir une carte d'interface pour contrôler l'état du lecteur et commander l'opération de lecture.

2. Vue d'ensemble

Le lecteur MP3 possède deux interfaces une pour l'afficheur 7 segments et l'autre pour la commande Infrarouge.

Le premier sera utilisé pour la lecture de l'état du lecteur tandis que le second sera exploité lors de l'envoi de commandes au lecteur (figure 9).

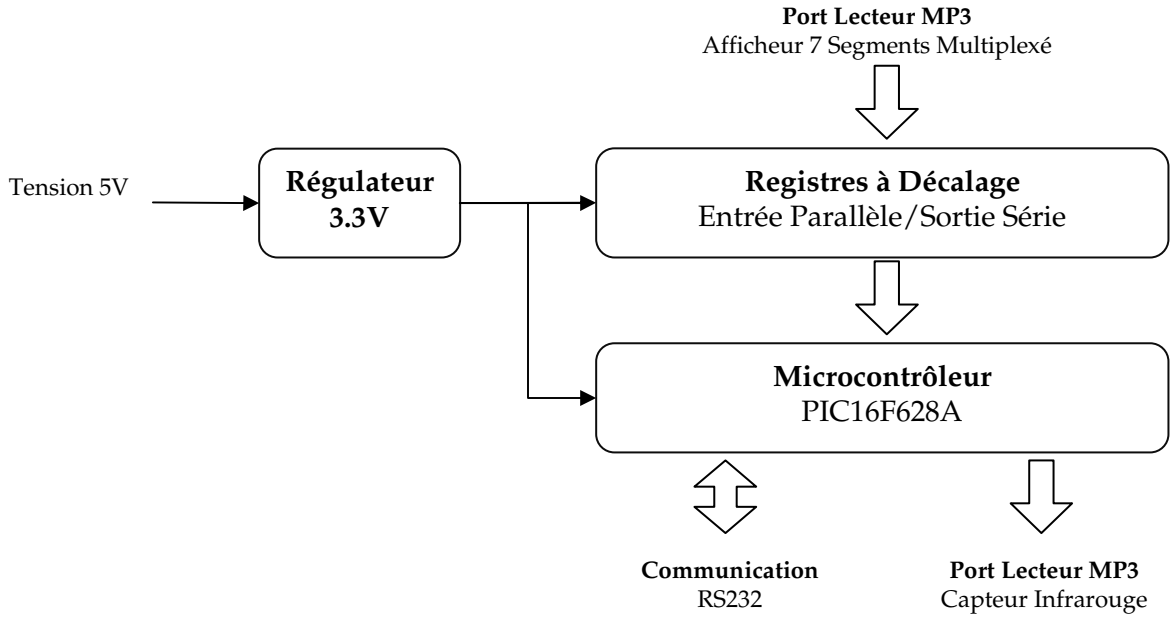


Figure 9, Vue d'ensemble de l'interface de commande du lecteur MP3

3. Conception

Le lecteur MP3 utilisé fonctionne avec une tension de 3.3V. L'interface de commande doit fonctionner avec cette tension pour ne pas endommager le lecteur, l'ajout d'un **régulateur** est inévitable.

Le microcontrôleur PIC16F628A fonctionne dans une plage de tension variant de 3V à 5,5V pour cela il a été retenu bien qu'il ne dispose pas de suffisamment de broches pour connecter toutes les sorties de l'afficheur 7 segments du lecteur. Pour palier à ce problème, il faudrait faire recours aux **registres à décalage**. Cette solution a l'inconvénient de monopoliser le microcontrôleur pendant 0.2ms pour la lecture d'un état du lecteur. Il est, aussi, possible d'utiliser un microcontrôleur qui comprend un nombre plus important de broches et qui fonctionne sous 3.3V.

Le microcontrôleur est utilisé pour analyser l'état du port de l'afficheur 7 segments, et donc pour en déduire l'état de fonctionnement du lecteur : Le numéro du morceau joué, la position de lecture actuelle, etc. Il servira, en outre, pour envoyer les commandes de lecture, d'arrêt, de changement de morceau à travers l'entrée Infrarouge.

La carte d'interface (annexe 1.4) sera pilotée en utilisant le protocole RS232 : débit 19200bps.

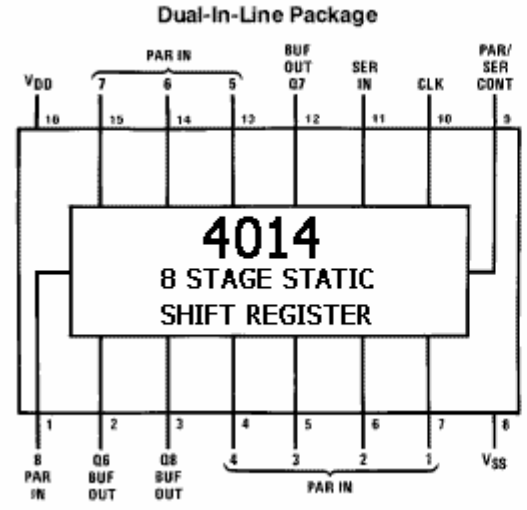
Son schéma de principe montre sept fonctions numérotées de F1 à F7 :

- F1 : Récupération de la sortie MP3, décalage des données (entrée parallèle/sortie série)
- F2 : Lecture données en série, Analyse de l'état du lecteur MP3, communication
- F3 : Interfaçage de la carte
- F4 : Régulation de tension 5V → 3.3V
- F5 : Adaptation des tensions d'entrées et des tensions de sortie : 3.3V ←→ 5V
- F6 : Commande de la mise sous tension du lecteur MP3, Activation/Désactivation des hauts parleurs
- F7 : Lecture de l'état du lecteur MP3, Commande du lecteur à travers l'entrée du capteur infrarouge, Communication avec les modules externes.

i. Fonction 1 - Décalage

La fonction décalage des données est basée sur un circuit intégré de la famille CMOS CD4014. Ce dernier est capable de fonctionner avec une tension de 3 volts. Il s'agit d'un registre à décalage 8bits à double fonction : **entrée série/sortie série** ou à **entrée parallèle/sortie série**.

Comme le montre le tableau de vérité (figure 10) l'activation du PAR/SER CONT pin (mise à 1) permet de charger l'état des entrée PAR IN dans des bascules au front montant de l'horloge CLK.



CL*	Serial Input	Parallel/ Serial Control	PI 1	PI n	Q1 (Internal)	Q_n
⌐	X	1	0	0	0	0
⌐	X	1	1	0	1	0
⌐	X	1	0	1	0	1
⌐	X	1	1	1	1	1
⌐	0	0	X	X	0	Q_{n-1}
⌐	1	0	X	X	1	Q_{n-1}
⌐	X	X	X	X	Q1	Q_n

Figure 10, Brochage et table de vérité du CD4014

Le circuit offre trois sorties Q8, Q7, Q6 pour les trois bits les plus significatifs. Pour accéder au contenu des autres bascules, il faudrait, commencer par désactiver PAR/SER CONT (mise à 0).

Un premier front montant de l'horloge CLK, par la suite, permet de décaler le contenu du registre (Q8=Q7, Q7=Q6, Q6=Q5) et donc de récupérer Q5. Un deuxième front montant permet de récupérer Q4, etc. Après chaque décalage Q1 = SER IN.

ii. Fonction 2- Etat du lecteur MP3

*** Solution Matérielle**

Le schéma fourni en annexe 1.4 ne montre aucune horloge externe pour la fonction [F2]. Le circuit PIC16F628A a été configuré pour fonctionner sur son horloge interne cadencée à 4MHz avec une précision de ±1%.

Une horloge externe (type quartz) peut permettre au MCU de fonctionner plus rapidement jusqu'à 10MHz sous une tension de 3 Volts avec une précision de ±0.003%.

L'usage d'une telle fréquence de fonctionnement avec une telle précision n'est pas justifié sachant que la fréquence maximale de l'horloge tolérée par le registre à décalage utilisé (CD4014) est de l'ordre de 4 MHz sous 5V et qu'elle diminue sous une tension de 3V.

Par contre, l'absence d'horloge externe, a des avantages tels que :

- Economiser deux pins du MCU. Ces pins seront utilisés pour interfacer le registre à décalage.
- Réduire l'encombrement de la carte d'interface et donc des dimensions du circuit imprimé.
- Minimiser le coût des composants de la carte d'interface.
- S'assurer du fonctionnement immédiat du MCU.

*** Solution Logicielle**

Le programme stocké dans le MCU est basé sur 5 procédures :

- **mp3_inst_state()** permet la lecture de l'état des sorties du lecteur MP3 à un instant donnée t.
- **mp3_state()** permet la lecture de l'état des sorties du lecteur MP3 durant les instants t, t+Δt, t+2.Δt, t+3.Δt, t+4.Δt (**tableau 1**).

- **ext_isr()** : routine d'interruption qui permet de communiquer l'état du lecteur sur activation (mise à 1) de RB0.

La fonction **mp3_inst_state()** s'exécute en 120µs à 4MHz. L'algorithme ci-dessous résume le fonctionnement de cette procédure.

```
1. Configurer les circuits U1 et U2 en mode entrée parallèle/sortie série
2. Créer un front montant, pour sauvegarder les données de J1 dans les
    circuits Latch de U1 et U2
3. Configurer les circuits U1 et U2 en mode entrée série/sortie série
4. Lire les trois bits G, F, E à partir de U1
5. Lire les trois bits C5, C4, C3 à partir de U2
6. Créer trois fronts montants, pour effectuer 3 décalages
7. Lire les trois bits D, C, B à partir de U1
8. Lire les deux bits C2, C1 à partir de U2
9. Créer un font montant, pour effectuer un décalage
10. Lire un bit A à partir de U1
```

Comme le montre le tableau 1, pour avoir une idée sur l'affichage actuel du lecteur MP3, il faudrait scruter la sortie du lecteur pendant les instants t, t+Δt, t+2.Δt, t+3.Δt, t+4.Δt. Ces instants correspondent au changement de l'état des cinq anodes du lecteur, une seule anode est active à un instant donné. La procédure **mp3_state()** réalise cette opération.

```
1. Lire (state.state) lorsque (A5 = 1)
2. Lire (state.dig1)  lorsque (A4 = 1)
3. Lire (state.dig2)  lorsque (A3 = 1)
4. Lire (state.dig3)  lorsque (A2 = 1)
5. Lire (state.dig4)  lorsque (A1 = 1)
6. Si state != oldstate alors
          Oldstate = state
          Répéter les étapes de 1 à 6
     Fin Si
7. Décoder(state.dig1)
8. Décoder(state.dig2)
9. Décoder(state.dig3)
10.   Décoder(state.dig4)
```

La fonction **Décoder()** permet de faire la correspondance entre le code hexadécimal de l'état des segments de l'afficheur et le caractère ASCII représenté par ces segments (figure 11).

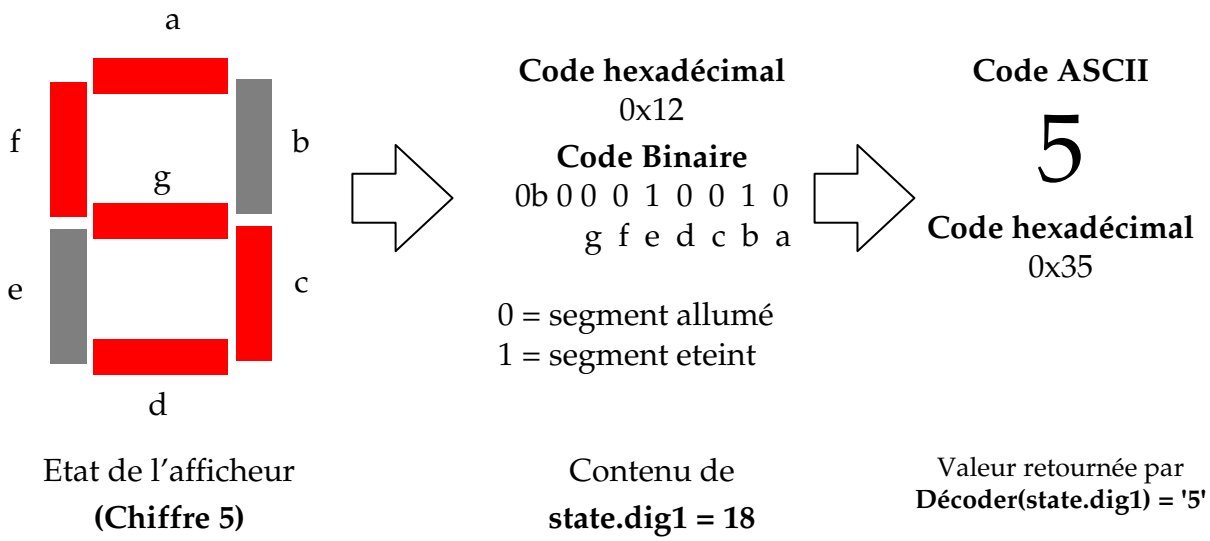


Figure 11, utilité de la fonction Décoder()

Le MCU PIC16F628A permet de gérer plusieurs types d'interruptions, dont l'interruption externe qui est déclenchée à l'activation du pin RB0/INT.

Dans notre montage (schéma en annexe 1.4) l'activation de cette broche nommée MP3 RQ SD permet de déclencher l'interruption qui fera appel à la routine d'interruption **ext_isr()**.

Cette routine permet de transférer l'état du lecteur MP3 au périphérique qui a déclenché l'interruption.

iii. Fonction 4 – Régulation

La tension d'alimentation du montage est de 5V, l'entrée/sortie du lecteur MP3 est de l'ordre de 3,3V. D'où il faudrait utiliser un régulateur pour assurer l'alimentation des composants fonctionnant avec 3,3V.

Bien qu'il existe dans le commerce des régulateurs fixes délivrant cette tension, j'ai utilisé le régulateur variable LM317T. Ce dernier permet d'obtenir une tension de sortie stable, en fonction des valeurs des résistances montées en pont diviseur de tension sur sa sortie (figure 12).

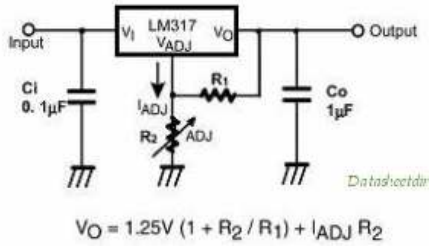


Figure 12, brochage du régulateur variable LM317T

La tension de sortie est calculée sur la base de la relation (**Vo**) tout en négligeant $I_{ADJ} \approx 100\mu A$:

$$\mathbf{Vo=1.25\ (1 + R2/R1)}$$

La valeur recommandée pour la résistance R1 est 240Ω. Sachant que: Vo = 3.3V, R1 = 240Ω on en déduit que ;

$$\mathbf{R2 = R1.(V0/1.25 – 1)} = 393.6\Omega \approx \mathbf{390\Omega}$$

iv. Fonction 5 – Adaptation

La carte d'interface communique avec l'extérieur sous une tension de 5 Volts.

- Les signaux entrants OE et RX devraient être ramenés à 3 Volts en utilisant un pont diviseur.
- Le signal sortant TX devrait être amplifié de 3,3V à 5V en utilisant des transistors bipolaires fonctionnant en régime de commutation (saturés, bloqués).

Si PIC TX est à l'état logique 0, Q1 est bloqué, V_{CE} = 5 Volts. S'il est à l'état logique 1, Q1 est saturé, VCE ≈ 0 Volts. On remarque que Q1 fonctionne comme une porte logique non. L'ajout de R5 et de Q2 permet de retrouver l'état logique correct à 5V (figure 13).

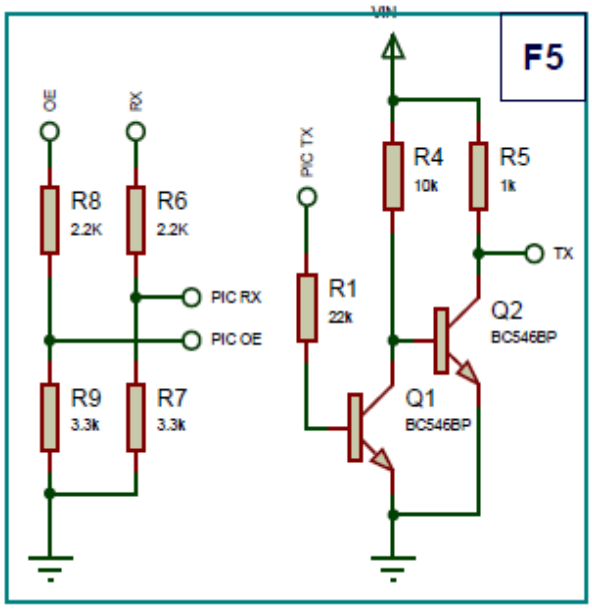


Figure 13, fonction adaptation

Les valeurs des résistances R1, R2 et R5 ont été calculées pour réduire le courant de base des transistors Q1 et Q2.

v. Fonction 6 – Commutation

Dès sa mise sous tension, le lecteur MP3 commence automatiquement la lecture. Cette propriété est considérée comme un défaut. La fonction de commutation permet de contrôler la procédure de lecture.

Cette fonction est basée sur deux relais 5 Volts commandés par le MCU à travers deux transistors de moyenne puissance 2N2222. Les diodes D1 et D2 sont deux diodes de signal, dites aussi diodes roues libre ou diodes de protection.

Ces diodes servent à limiter le courant inverse généré par la bobine du relais à l'instant de son déclenchement. Ce courant risque de détruire les transistors de commutation.

Chapitre 2 - Afficheurs Sept Segments

1. Présentation

L'affichage des heures de prières est considéré comme la fonction la plus importante du système. Le système comportera huit afficheurs : l'heure, la date, les horaires des cinq prières et l'heure de levée du soleil (utilisée pour déterminer l'heure du salat « A'dhuha »).

Afin de se conformer au cahier de charges, les valeurs affichées doivent être visibles depuis une distance supérieure à cinq mètres et dans des conditions de faible éclairage.

Les afficheurs sept segments répondent à ces exigences, ils seront retenus pour le projet.

2. Vue d'ensemble

Le module comporte plusieurs étages (figure 14). Le nombre réduit des pins du MCU incite à l'utilisation des registres à décalage. Un étage de puissance est requis pour fournir le courant nécessaire à l'afficheur sept segments.

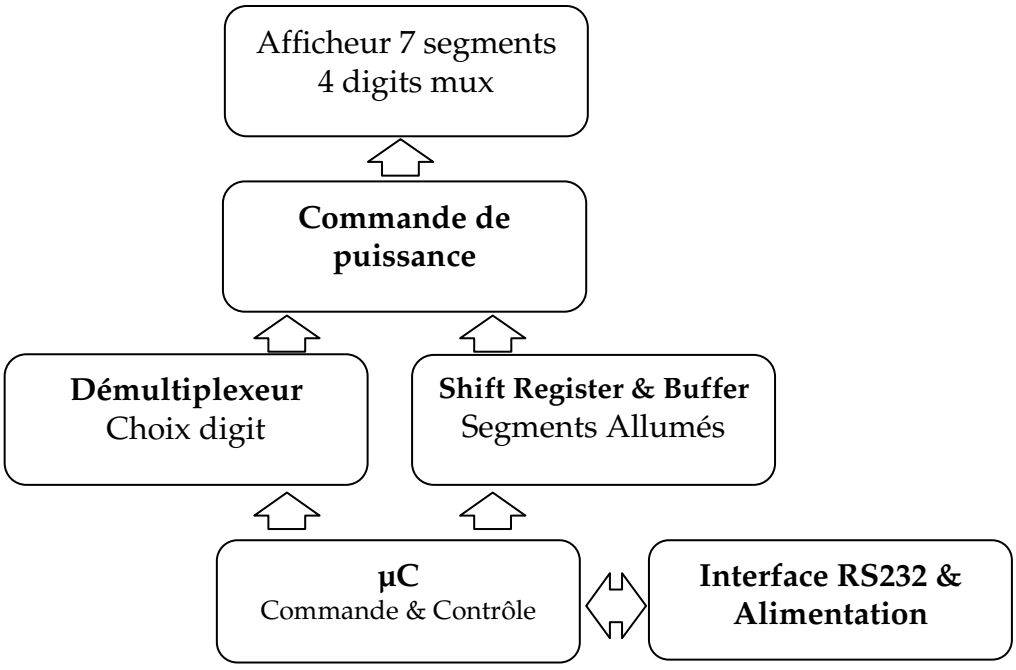


Figure 14, Vue d'ensemble de l'afficheur sept segments

3. Conception

Lors de l'utilisation d'un afficheur sept segments multiplexé à cathode commune, la logique de commande doit procéder comme suit :

1. Sélectionner un digit, la cathode commune devra être reliée à la masse
2. Allumer les segments requis pour afficher un chiffre, chaque segment doit être commandé sous une tension de 2,2 Volts et un courant de 25mA.
3. Maintenir pendant un laps de temps entre 2 ms et 10 ms
4. Procéder de même pour afficher les autres chiffres

Le composant, sujet de ce chapitre, est un composant intelligent capable de recevoir une chaîne de caractères (32 caractères maximum) et de l'afficher.

Ce composant est construit autour d'un microcontrôleur et composé de 5 étages/fonctions :

1. **Fonction interface/alimentation** permet d'assurer l'alimentation du composant et son interface avec l'extérieur via le protocole RS232.
2. **Fonction contrôle & commande** assurée par le microcontrôleur.
3. **Fonction mémoire à décalage** a pour rôle de commander les sept segments de l'afficheur avec un minimum de pins du microcontrôleur.
4. **Fonction démultiplexeur** assure la sélection d'un chiffre.
5. **Fonction commande de puissance** permet de fournir le courant et la tension adéquats pour attaquer l'afficheur à partir des signaux de commande.

a. Fonction interface/alimentation

La tension d'alimentation utilisée pour alimenter le composant est de 5 volts. Deux condensateurs de filtrage, notamment, un de moyenne capacité 470µF et l'autre de faible capacité 10µF sont utilisés pour maintenir la tension d'alimentation stable.

Comme une chute de tension peut affecter le fonctionnement du microcontrôleur le condensateur de 10µF est branché tout près du MCU. Le condensateur de 470µF est aussi utilisé pour cette fin mais tout près de l'étage de puissance.

b. Fonction contrôle et commande

Le microcontrôleur utilisé dans l'afficheur sept segments est le PIC16F628A. Ce circuit est retenu pour plusieurs raisons telles que :

- Sa capacité de mémoire RAM : 224 octets, dont 32 octets seront réservées pour accueillir la chaîne de caractères à afficher.
- Son support matériel du protocole RS232.
- Sa taille : le boîtier est de 18 pins.
- Son horloge interne est cadencée à 4 MHz. Aucun oscillateur externe n'est requis.

Le microcontrôleur est programmé pour fonctionner en mode interruption :

- Interruption Timer 0 est déclenchée périodiquement toutes les 2ms pour assurer l'affichage d'un caractère du tampon.
- Interruption Timer 1 est déclenchée périodiquement toutes les 262ms pour réaliser l'animation du point décimal.
- Interruption RDA (Received DAta) est déclenchée à chaque réception d'un caractère ASCII sur le pin RX du microcontrôleur. Le caractère reçu est stocké dans une mémoire tampon.

c. Fonction mémoire à décalage

Le circuit CD4094 (figure 15) est un registre à décalage/buffer 8 bits avec des sorties à 3 états. Il servira dans notre montage pour stocker l'état des sept segments d'un chiffre et l'état du point décimal.

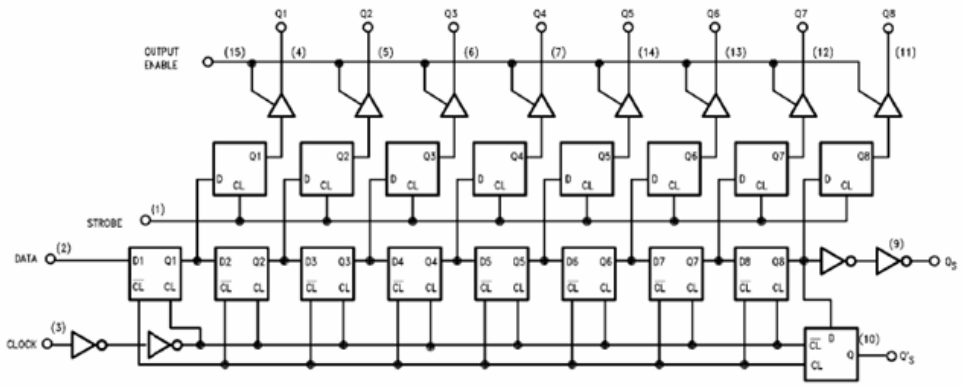


Figure 15, schéma de principe du CD4094

A chaque front montant de l'horloge (CLOCK), les données (DATA) sont décalées. Les données (Q1, Q2 … Q8) sont chargées dans un buffer au front montant du STROBE. Elles sont disponibles lorsque OUTPUT ENABLE est à l'état haut.

d. fonction démultiplexeur

J'ai opté pour un démultiplexeur 3bits/8sorties parce qu'aux débuts du PFE j'ai imaginé d'utiliser un afficheur 8 digits, le circuit démultiplexeur 3 bits (74HC238) a été choisi (figure 16).

Ce circuit peut être étendu pour commander 64 sorties, et ce, grâce à ses trois entrées ENABLE.

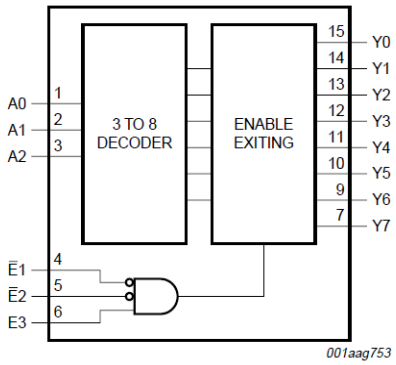


Figure 16, schéma de principe du 74HC238

Uniquement E3 est utilisé dans notre montage (Annexe 1.6) les autres entrées (E1 et E2) sont branchées à la masse.

e. Etage de puissance

Puisque les quelques milliampères en sortie du 74HC238 et du CD4094 ne sont pas suffisants pour piloter l'afficheur sept segments le recours à un étage de puissance est inévitable.

Cet étage utilisera des transistors bipolaires fonctionnant en mode commutation. Les transistors à effet de champs MOSFET peuvent être aussi utilisés. Ils sont plus utiles lorsque des courant plus forts sont requis.

L'afficheur sept segments à commander fonctionne sous une tension de 2,2 volts et un courant de 25mA par segments. Il s'agit d'un afficheur à cathode commune.

Dans le cas d'un affichage à l'aide d'un seul digit, pour chaque segment allumé, le courant I_D est de 25mA ; alors que le courant IK est la somme des courants des segments allumés (I_K = 150mA) (Figure 17).

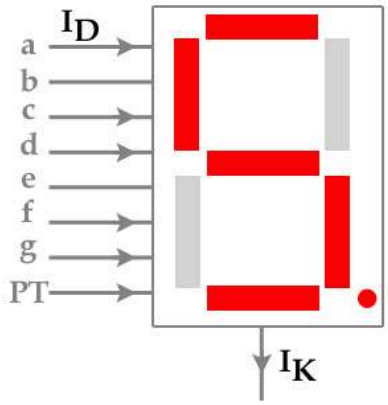


Figure 17, grandeurs électriques nécessaires pour commander un chiffre de l'afficheur 7 segments

Le montage suivant (Figure 18) donne le principe de commande d'un segment de l'afficheur à l'aide de deux transistors bipolaires commandés par des signaux logiques aux entrées IA et DIG1.

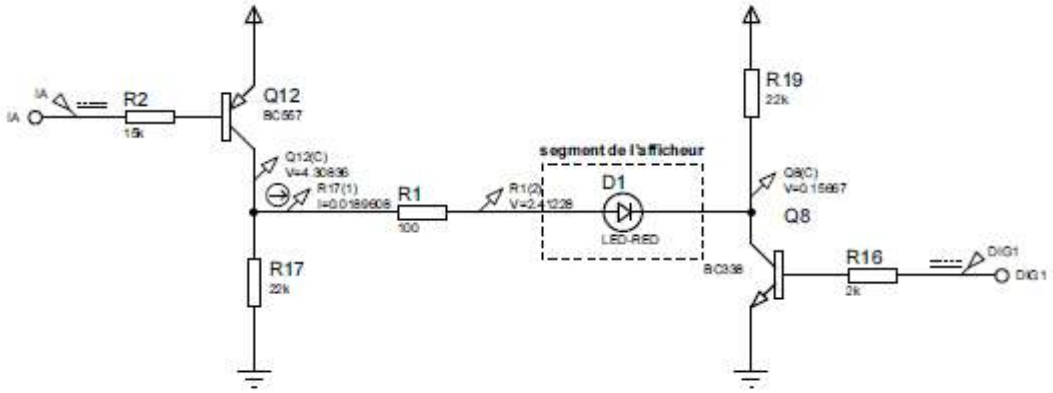


Figure 18, commande d'un segment de l'afficheur

Les valeurs des résistances R1, R2, R16, R17 et R19 ont été calculées. La simulation montre que ces valeurs sont convenables :

- Le segment de l'afficheur est soumis à une tension de 2.26V
  - R1(2) = 2,41V, Q8(C) = 0,16V
- Le courant qui parcoure ce segment est de 19mA.

i. Choix de R17 et R19

Les résistances R17 et R19 sont utilisées afin de polariser les transistors (en absence de charge) leurs valeurs doivent être grandes pour permettre un courant très faible.

Comme le courant requis par la charge est de 25mA, le courant traversant ces résistances devrait être négligeable. Pour un courant $I_{R17} = I_{D1}/100$, $U_{D1} = 4.8V$, $I_{D1} = 25mA$

$R17 \geq U_{R17} / I_{R17} = 100.U_{R17}/I_{D1} = 19200\Omega$, la valeur normalisée la plus proche est 22KΩ.

ii. Calcul de R2 et R1

Afin de simplifier les calculs, il serait utile de simplifier la figure 18 et la remplacer par la figure 19. Le transistor est utilisé en mode tout ou rien, il est soit saturé, soit bloqué.

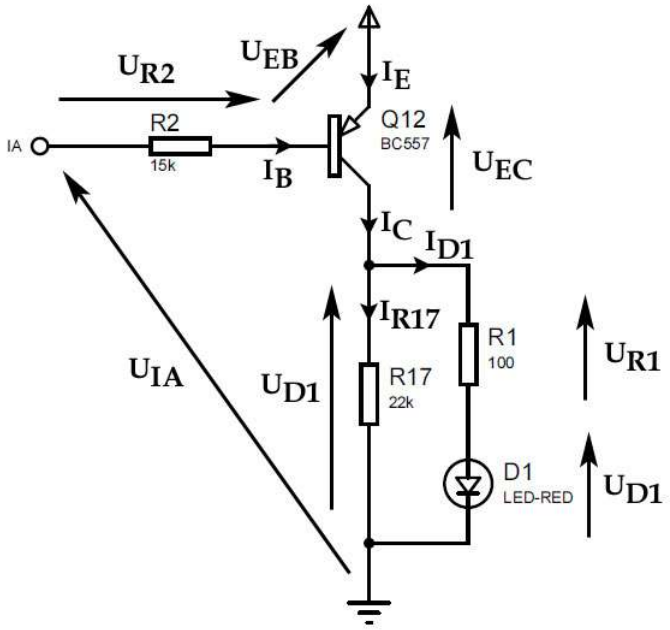


Figure 19, Polarisation du transistor BC557

Le transistor est bloqué lorsque : $U_{IA} = 5V$ ➔ $U_{EB} = 0V$ et $U_{EC} = 5V$

Le transistor est saturé lorsque : $U_{IA} = 0V$ ➔ $U_{EB} = 0,7V$ et $U_{EC} \approx 0,2V$

$U_{CC} = U_{EC} + U_{R1} + U_{D1}$, avec $U_{R1} = R1.I_{R1}$, $I_{D1} = I_{R1}$ ➔ $R1 = (U_{CC} – U_{EC} – U_{D1}) / I_{D1}$

$U_{CC} = U_{EB} + U_{R2}$, avec $U_{R2} = R2.I_{R2}$ ➔ $R2 = (UCC – UEB) / I_{R2}$

$I_C = I_{D1} + I_{R17} = \beta.I_B$ et $I_B = I_{R2}$

Comme I_{R17} est négligeable devant I_{D1}, on suppose que $I_{D1} = \beta.I_{R2}$,

Pour $U_{D1} = 2,2V$ et $I_{D1} = 25mA$ ➔ $R1 = 104\Omega$, la valeur la plus proche est 100Ω

$R2 = \beta.(U_{CC} – U_{EB}) / I_{D1} = 17200\Omega$, la valeur normalisée la plus proche est 15KΩ

iii. Calcul de R16

Les entrées DIG1 à DIG4 permettent de sélectionner le chiffre de l'afficheur sept segments à activer (Figure 20). Lorsqu'une entrée DIG1 est activée elle doit être capable de

commander un courant de 200mA sur le collecteur du transistor (IC=200mA en négligeant IR19 devant ICH).

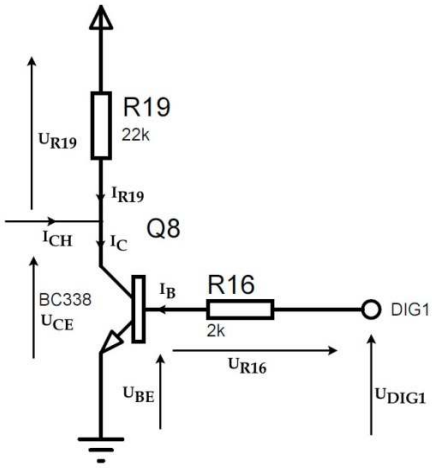


Figure 20, Polarisation du transistor BC338

$U_{DIG1} = U_{R16} + U_{BE} = R16.I_B + U_{BE}$, transistor saturé ➔ $U_{DIG1} = U_{CC}$

$U_{CC} = U_{R19} + U_{CE}$

$R16 = (U_{CC} – U_{BE}) / I_B = \beta. (U_{CC} – U_{BE}) / I_C = 2150\Omega$,

La valeur normalisée la plus proche est 2KΩ.

4. Réalisation & Tests

Le circuit électronique comporte deux parties : hardware et software. La première partie se termine par la réalisation physique du circuit ; alors que la deuxième est achevée lorsque le circuit est complètement fonctionnel.

a. Hardware

Le schéma de principe du circuit électronique complet est présenté en annexe 1.6. Ce schéma a été utile pour réaliser des simulations. Par la suite il a servi à réaliser le routage du circuit imprimé.

Le développement du circuit imprimé a été confié à une société spécialisée à partir du routage que j'avais réalisé. Il faudrait, par la suite, procéder à la soudure des composants sur le circuit imprimé.

Pour ne pas griller les composants fragiles lors de la soudure, un ordre logique doit être maintenu : diodes, circuits intégrés, transistors… J'ai commencé, premièrement, par les via de connexion. Ensuite, les résistances, les supports des circuits intégrés, les transistors et

enfin les capacités et les connecteurs. La partie hardware du circuit électronique est terminée par la soudure du dernier composant.

b. Software

Le circuit ne sera fonctionnel qu'à la suite de l'upload du programme dans le microcontrôleur. D'abord la tache parait simple. Mais pour assurer le minimum de disfonctionnements et de problèmes de mise en route, il faut procéder à des tests unitaires. Ces derniers seront réalisés à l'aide d'une carte électronique qui permet de fournir la source d'énergie et la connectivité nécessaire à un PC.

Pour le circuit en cours de développement :

1. **Test du fonctionnement général :** allumer chaque segment pour chaque digit de l'afficheur sept segments.
2. **Test de l'afficheur avec des séquences de quatre chiffres :** réaliser un compteur de 0 à 9999.
3. **Test de l'afficheur avec une chaîne de plus de quatre caractères :** afficher des chaînes de caractères avec plus de quatre caractères afin de trouver la temporisation idéale de défilement.
4. **Test de la communication :** envoyer des chaînes de caractères à partir du PC pour évaluer la réponse du circuit et faire le bon choix du débit.

Chapitre 3 - Afficheur LCD Série

1. Présentation

L'afficheur LCD est important dans Prayer Caller puisqu'il assure l'interface de commande avec l'utilisateur et sera utile au paramétrage du montage.

L'afficheur LCD diffère de l'afficheur sept segments abordé dans le chapitre précédent : les afficheurs vendus dans le commerce intègrent leur logique de commande, leur protocole de communication, mais présentent d'autres contraintes.

L'afficheur LCD qui sera utilisé requiert un minimum de 12 signaux dont 4 pour le transfert des données par nibbles (4 bits) et supporte un protocole de communication, presque, standard. Cet afficheur présente plusieurs contraintes dues à son encombrement et au nombre important de signaux requis pour son pilotage.

Pour détourner toutes les contraintes et garantir l'indépendance de ce composant des autres composants de Prayer Caller, l'idée la plus simple sera de transformer l'afficheur en afficheur série supportant le protocole RS232. Un µC sera chargé de l'interfaçage de l'afficheur.

2. Vue d'ensemble

Les données affichées par l'afficheur LCD sont transmises en série à travers un registre à décalage piloté par un MCU (figure 21).

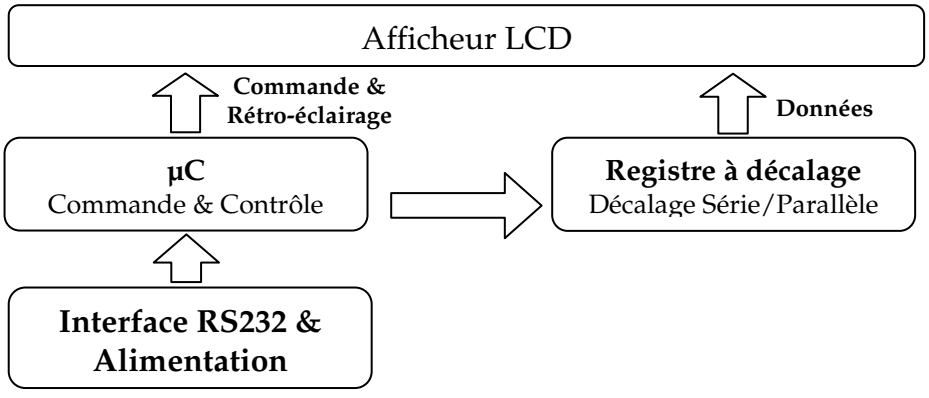


Figure 21, Vue d'ensemble de l'interface de l'afficheur LCD Série

3. Conception

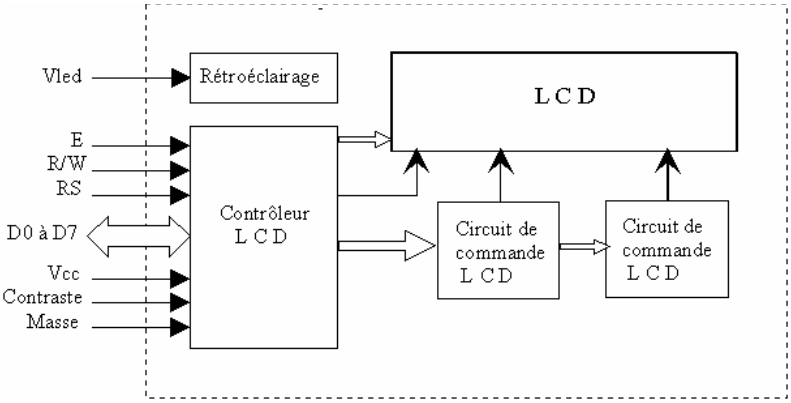


Figure 22, schéma de principe de l'afficheur LCD standard

L'afficheur LCD vendu dans le commerce (Figures 22) est un composant qui nécessite un interfaçage adéquat pour afficher un caractère :

- Positionner l'entrée Register Select :
  - RS = 0 : Caractère de commande
  - RS = 1 : Caractère à afficher
- Désactiver l'entrée Read/Write : R/W = 0
- Activer l'entrée Enable : E = 1
- Envoyer les données D0 ➔ D7
- Désactiver l'entrée Enable : E = 0

L'afficheur LCD série essayera d'intégrer les fonctions offertes par l'afficheur LCD standard tout en ajoutant de nouvelle fonctions : l'effacement de l'écran, le positionnement du curseur, la suppression de caractères, etc. Ces fonctions seront disponibles via des séquences d'échappement : ESC = Caractère ASCII 27

- [ESC] C : Effacer l'écran
- [ESC] L : Effacer toute la ligne en cours, positionner le curseur en début de ligne
- [ESC] E : Effacer de la position du curseur à la fin de ligne sans modifier l'emplacement du curseur
- [ESC] N : Allumer le rétro-éclairage
- [ESC] F : Eteindre de rétro-éclairage
- [ESC] G [X] [Y] : Positionner le curseur à la position X, Y
- [ESC] P [L | R | U | D] : Positionner le curseur

Le circuit comporte trois étages (figure 21) :

1. **Fonction contrôle et commande :** assurée par le microcontrôleur.
2. **Fonction décalage série/parallèle :** permettant de réduire les coûts de réalisation.
3. **Fonction rétro-éclairage**.

a. Fonction contrôle et commande

Le microcontrôleur retenu dans cette réalisation est le PIC16F628A. A l'inverse des autres réalisations un oscillateur externe (quartz) sera utilisé pour cadencer le composant. La fréquence interne de l'oscillateur (4MHz) étant insuffisante : le microcontrôleur exécute 1 MIPS.

Suite à la réception d'un caractère, l'interruption INT_RDA du microcontrôleur est appelée :

- Lecture du caractère
- Si séquence d'échappement ➔ traitement de la commande
- Si caractère normal ➔ stockage dans la mémoire tampon
- Rafraîchissement de l'afficheur

Les caractères d'une chaîne de caractères arrivent à des intervalles de 1041µs avec une connexion série 9600 bauds.

- Avec un oscillateur 4MHz ce temps permet d'exécuter seulement 1041 instructions.
- Avec un oscillateur 20MHz on pourrait exécuter 5205 instructions.

On remarque que l'oscillateur interne ne permet pas au microcontrôleur (avec 1041 instructions) d'assurer correctement le traitement des séquences d'échappements reçues.

b. Fonction décalage série/parallèle

Le circuit retenu pour assurer la fonction décalage série/parallèle est le 74HC595 ce composant est presque identique au CD4094 utilisé précédemment.

Au début, la simulation du montage a été réalisée avec le CD4094. Mais, comme le courant qu'il fourni est insuffisant pour commander l'afficheur LCD je l'ai remplacé par son équivalent TTL :74HC595 (Figure 23).

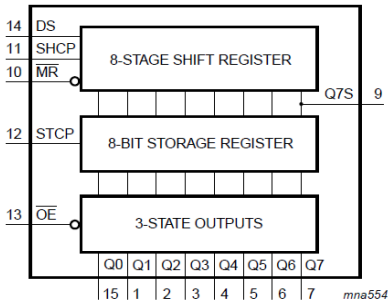


Figure 23, schéma fonctionnel du registre à décalage 74HC595

c. Fonction Rétro-éclairage

Avant la levée et après le coucher du soleil la lecture des valeurs sur l'afficheur LCD n'est pas possible sans la fonction rétro-éclairage. Selon la Datasheet, le courant nécessaire pour activer cette fonction est de l'ordre 300mA d'où la nécessité d'utilisation d'un commutateur à transistor. Le montage ci-après (Figure 24) permet de limiter le courant de commande du rétro éclairage.

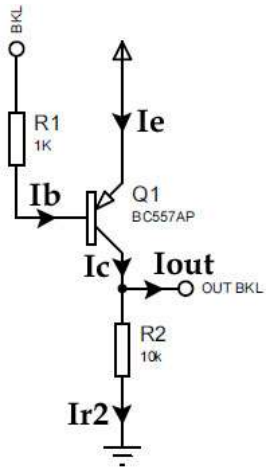


Figure 24, commande du rétro-éclairage

Les caractéristiques requises pour activer le rétro éclairage :

$$\begin{cases} I_{out} = 300mA, \\ Vout = 3.2V. \end{cases}$$

Lorsque le transistor est saturé :

$$\begin{cases} V_{CC} = V_{EB} + R1.I_B \\ V_{CC} = V_{EC} + V_{OUT} \end{cases}$$

Comme :

$$\begin{cases} I_{r2} << I_{OUT} \rightarrow I_C \approx I_{OUT} ; \\ I_B = I_C / \beta \text{ ; avec } \beta \approx 100 \end{cases}$$

Le courant de commande requis est uniquement de l'ordre de :

$$I_B = I_C / \beta = 3mA$$

La résistance R2 est une résistance de polarisation, sa valeur a été choisie pour dissiper un courant I_{r2} très faible devant I_{OUT} :

$$I_{r2} = V_{OUT} / R2 = 0.32mA$$

La résistance R1 a été calculée :

$$R1 = (V_{CC} – V_{EB}) / I_B = (V_{CC} – V_{EB}).\beta / I_C = 1433\Omega$$

La valeur normalisée la plus proche est de : 1KΩ

Chapitre 4 - Périphériques I2C - Mémoire Série/Horaires de Prière - Horloge RTC

1. Présentation

L'horaire des prières sera stocké dans une mémoire série I_2C. Cette mémoire d'une capacité 512Kbits (64Ko) permet de stocker les horaires de prière pour 13 villes différentes.

Ce chapitre sera consacré pour discuter les aspects suivants :

- Bus I_2C
- Méthodologie de stockage et d'accès à une mémoire série I_2C
- Méthode de stockage des horaires de prière
- Calcul des horaires de prière
- Horloge RTC

2. Vue d'ensemble

Le boîtier comporte 8 broches deux pour connecter la mémoire sur un bus I_2C (SDA, SCL), deux pour l'alimentation (V_{CC}, V_{SS}), trois pour sélectionner l'adressage du composant (A0, A1, A2), une pour le verrouillage en écriture de la mémoire (WP) (Figure 25).

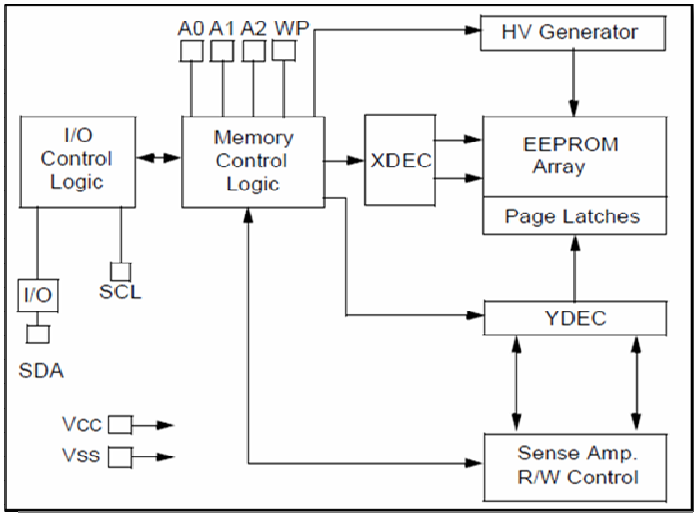


Figure 25, Vue d'ensemble de la mémoire I_2C, 24LC512

3. Bus I_2C

a. Présentation

Le bus I_2C (Figure 26) (Inter Integrated Circuit) a été inventé par Philips. C'est un bus simple basé sur uniquement deux fils. Il permet à plusieurs composants électroniques (Esclaves) connectés en cascade d'échanger des données avec un microcontrôleur (Maitre). Uniquement deux résistances pull-up sont nécessaires pour son fonctionnement.

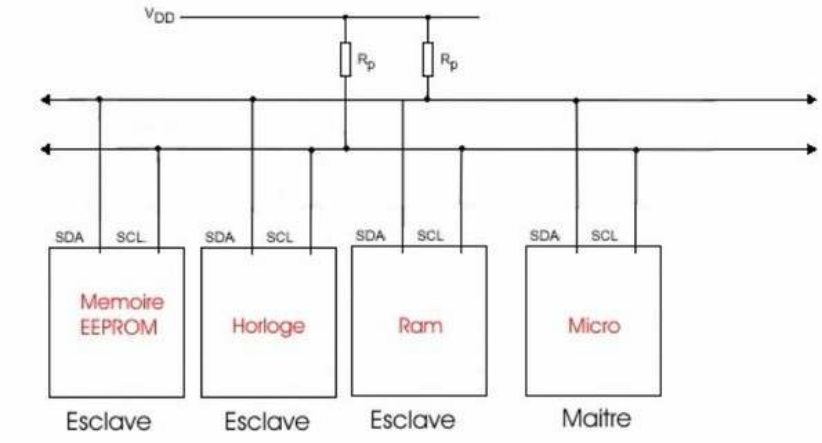


Figure 26, Bus I_2C

Le maître du bus génère le signal d'horloge SCL. Les données circulent dans les deux sens :

- Maître → Esclave : L'esclave génère les signaux d'acquittement (ACK)
- Maître ← Esclave : Le maître génère les signaux d'acquittement (ACK)

Les signaux d'acquittement (ACK) permettent d'indiquer que les données sont bien reçues par le destinataire.

Les données échangées sont codées sur 8 bits. La transmission commence un Start Bit et se termine nécessairement par un Stop Bit. Il est possible d'envoyer plusieurs Start Bit pour changer le mode de transmission (Lecture/Ecriture).

b. Protocole de lecture

La figure 27 présente la procédure de lecture de données depuis un périphérique I_2C. Le maître du bus peut lire un ou plusieurs octets. Il commence par sélectionner l'esclave en mode lecture @S/R[3]. L'esclave sélectionné doit acquitter ACK, il est alors prêt à envoyer ses données.

Le maître commence la lecture en acquittant chaque octet (ACK), le non acquittement (NOACK) indique la fin de l'opération.

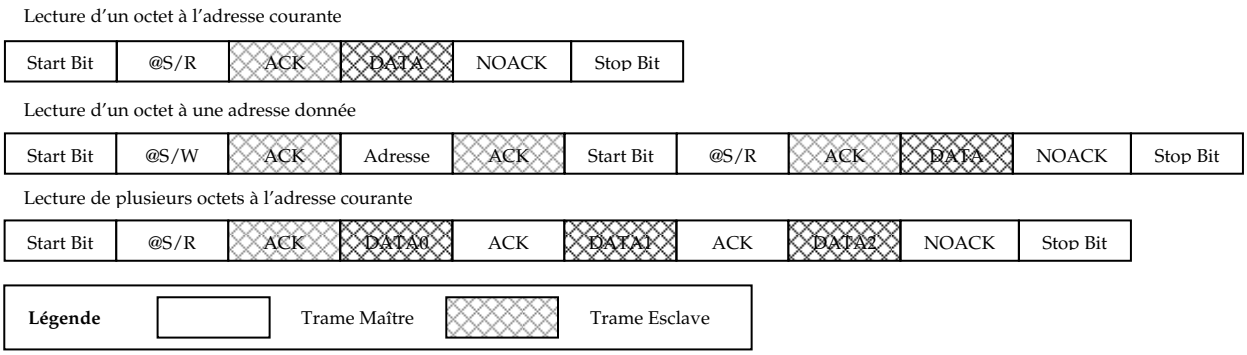


Figure 27, Protocole de lecture I_2C

c. Protocole d'écriture

La figure 28 présente la procédure d'écriture de données dans un périphérique I_2C.

Le maître du bus peut écrire un ou plusieurs octets dans une seule opération. Il commence par sélectionner l'esclave en mode écriture @S/W[4].

L'esclave doit acquitter (ACK) pour indiquer qu'il est prêt. Le maître lui envoie l'adresse initiale d'écriture puis les données. L'esclave doit acquitter (ACK) chaque octet reçu, il est informé par la fin d'opération lorsque le maître émet le bit de stop.

[3] @S/R = Slave Address/Reading Mode
[4] @S/W = Slave Address/Writing Mode

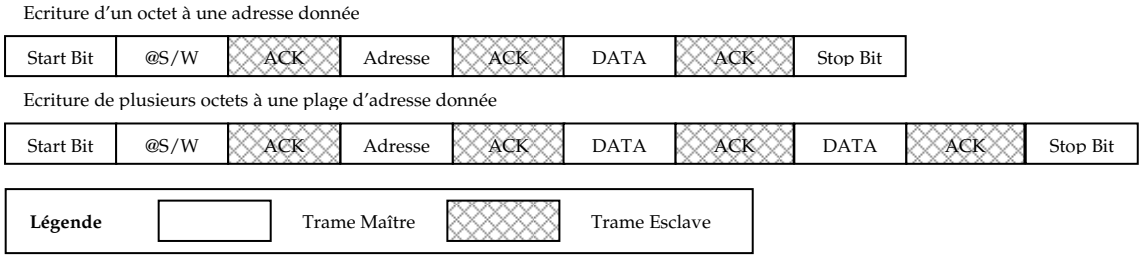


Figure 28, Protocole d'écriture I_2C

4. Accès à une mémoire série

La mémoire série 24LC512 est une mémoire 512Kbits (64 Ko) divisée en 512 pages de taille 128 octets. L'adresse d'une case mémoire est codée sur 16 bits (2 octets).

Lors de l'écriture de plusieurs octets à partir d'une adresse indiquée on doit tenir compte des limites de la page sinon on risque d'écraser des données au début de page.

La figure 29 montre la procédure d'écriture de 6 octets à partir de l'adresse 0x007C, on remarque que les octets 0x0000 et 0x0001 ont été écrasés lors de cette opération.

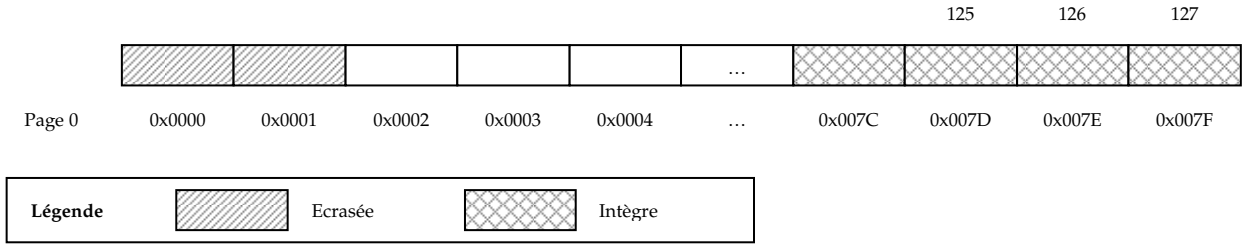


Figure 29, écriture audela des limites d'une page

5. Méthode de stockage des horaires de prières

Les horaires de prières d'une ville nécessitent 38 pages mémoire.

La première page sera réservée aux données de la ville : son nom et ses coordonnés, comme suit :

- Pays : 16 caractères
- Région : 16 caractères
- Ville : 16 caractères
- Longitude : 6 octets
- Latitude : 6 octets

Les pages suivantes seront consacrées aux horaires de prières. Une page pouvant contenir les horaires de 10 journées. Chaque journée nécessite 12 octets à raison de 2 octet (codes en BCD) pour chaque prière.

6. Calcul des horaires de prière

Le module de calcul des horaires de prière est basé sur la bibliothèque libITL (Islamic Times Library) initialement développée en langage C pour l'environnement Linux.

La bibliothèque contient des modules paramétrables pour le calcul des heures de prière, pour la conversion entre les dates grégoriennes et les dates héjires et des modules pour déterminer les dates des fêtes Islamiques (Aïd Kébir, Aïd Al Adha, Jour de l'an Arabe, etc.). J'ai porté, uniquement, le module de calcul des horaires de prière sous le langage Python. Les étapes de développement du logiciel seront explicitées par la suite.

7. Horloge RTC

L'horloge RTC (DS1037, figure 30) est un périphérique I_2C qui possède comme avantages :

- Horloge de précision avec une faible dérive
- Nombre minimal de broches nécessaires pour accéder aux informations avec possibilité de cascade avec d'autres périphériques (2 mémoires I_2C dans notre montage Annexe 1.7)
- Fonctionnement hors tension sur pile avec une consommation négligeable

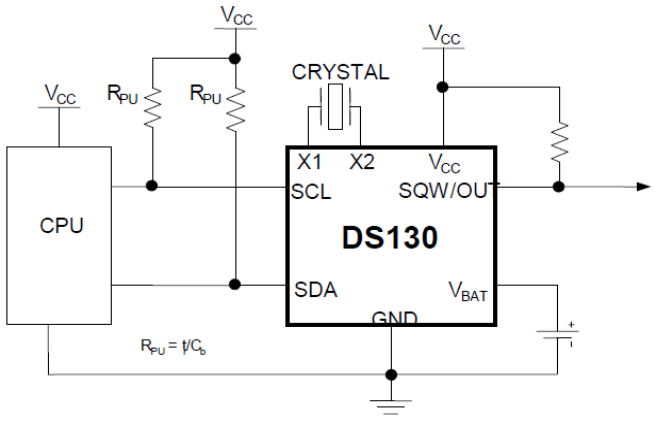


Figure 30, Montage typique du DS1037

Le circuit nécessite une horloge Quartz 32KHz et une source d'alimentation secondaire (une pile 3Volts [CR2032]) V_{BAT} (Figure 30).

Les données sont accessibles selon le modèle indiqué en 3.b (Tableau 2).

Table 2, Résumé des fonctions offertes par le circuit DS1307

Adresse	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Fonction	Intervalle
00h	CH	10 Secondes			Secondes				Secondes	00 – 59
01h	0	10 Minutes			Minutes				Minutes	00 – 59
02h	0	12	10 Heures		Heures				Heures	00 – 23
		24	AM/PM	10 Heures						1 – 12 AM/PM
03h	0	0	0	0	0	Jour			Jour	0 – 7
04h	0	0	10 Jour du mois		Jour du mois				Jour du mois	01 - 31
05h	0	0	0	10 Mois	Mois				Mois	01 – 12
06h	10 Année				Année				Année	00 – 99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Contrôle	-
08h – 3Fh									RAM 56 x 8	00 – 0xFF

Chapire 5 - Commande Infrarouge

1. Présentation

La commande du montage sera exclusivement en Infrarouge à partir d'une télécommande d'un récepteur Echo Star. Cette télécommande utilise le protocole standard de Philips RC5.

Un montage basé sur un microcontrôleur PIC a été réalisé afin de décoder ce protocole. Le décodage permettra d'affecter un code ASCII à chaque touche de la télécommande.

Le module de commande communique avec le module principal en utilisant le protocole RS232.

2. Vue d'ensemble

Les capteurs infrarouges disponibles dans le commerce nécessitent peu de composants, un seul MCU sera utilisé dans le module infrarouge (figure 31).

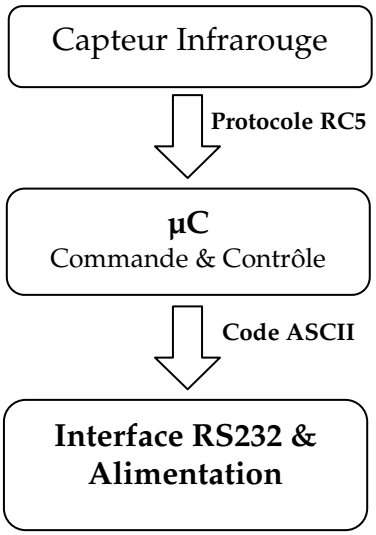


Figure 31, Vue d'ensemble du module de commande Infrarouge

3. Conception

Le circuit est réalisé autour du PIC16F628A, il implémente trois fonctions :

- **Fonction décodage du protocole RC5**
- **Fonction Notification :** qui permet de notifier un circuit externe de l'appui d'une touche
- **Fonction communication**

a. Protocole RC5

Le protocole RC5 a été initialement crée pour piloter les appareils audio/vidéo (télévision, chaîne HiFi, récepteur numérique, etc.). Le protocole RC5 utilise un codage Manchester inversé où chaque bit est transmis par deux états (Figure 32).

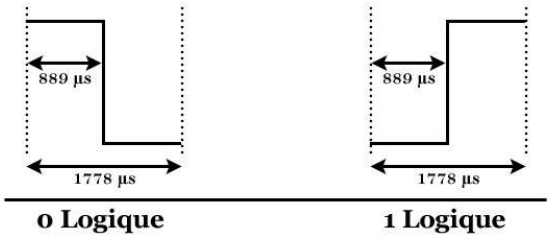


Figure 32, Codage Manchester Inversé

Une trame RC5 est codée sur 14 bits (figure 33) :

- Bits de début : 2 bits
- Bit de commutation : 1 bit
- Adresse de l'appareil : 5 bits
- Instruction : 6 bits

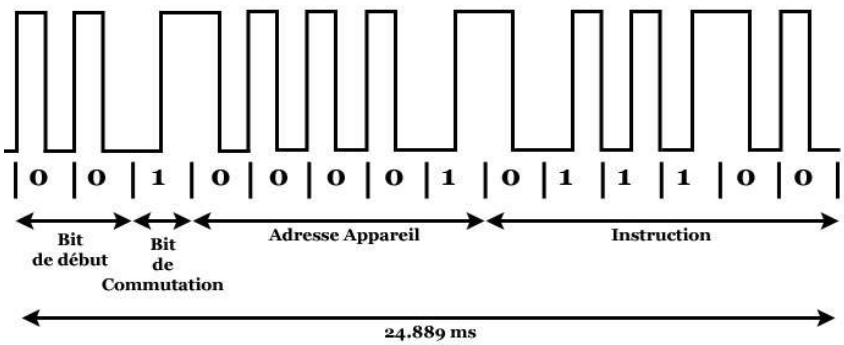


Figure 33, Protocole RC5

Pour le cas de la télécommande Echo Star utilisée l'appui sur la touche On/Off génère le code binaire 0x0E94 et 0x2E94 pour un appui répétitif sur la même touche (figure 34). Le premier octet se répète pour toutes les autres touches, uniquement le deuxième octet est distinct.

0x0E94 = b00	0	01110	100101	00
0x2E94 = b00	1	01110	100101	00
	Bits début	Bit de commutation	Adresse appareil	Instruction	Bits non utilisés

Figure 34, Code RC5 de la touche On/Off maintenue

b. Fonction décodage du protocole RC5

La méthode de décodage utilisée est la plus simple des méthodes disponibles. Elle consiste à effectuer le découpage temporel du signal provenant du capteur Infrarouge (Figure 35). Le résultat de ce dernier découpage est analysé pour trouver le code RC5 de la télécommande.

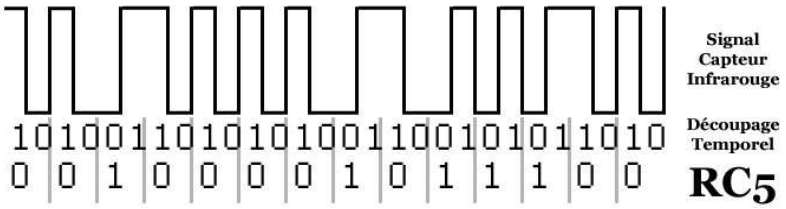


Figure 35, Découpage temporel et décodage RC5

c. Fonction Notification/Communication

Lorsque l'utilisateur appuie sur une touche de la télécommande le module de commande infrarouge reçoit une trame RC5. La trame est analysée pour connaître quelle touche a été appuyée.

Afin de simplifier les traitements dans le module principal un code ASCII est attribué à chaque touche selon le tableau 3.

Lorsqu'une donnée est disponible le module active la sortie IR DATA READY du MCU (Annexe 1.7)

Table 3,Codes RC5 des touches de latélécommande Echo Star

Bouton Télécommande	Code RC5	Caractère ASCII	Bouton Télécommande	Code RC5	Caractère ASCII
ON/OFF	0x0E94	A	Exit	0x0ED0	G
Mute	0x0E90	B	Up	0x0EFC	H
EPG	0x0E8C	C	Left	0x0EF4	I
…	0x0E88	D	Right	0x0EF0	J
I	0x0E84	E	Down	0x0EF8	K
1	0x0EB8	1	OK	0x0E80	L
2	0x0EB4	2	P+	0x0ECC	M
3	0x0EB0	3	P-	0x0EC8	N
4	0x0EAC	4	RCL	0x0EC4	O
5	0x0EA8	5	TV/RADIO	0x0EC0	P
6	0x0EA4	6	FAV	0x0EEC	Q
7	0x0EA0	7	Language	0x0EE8	R
8	0x0E9C	8	Bleu	0x0EE4	S
9	0x0E98	9	Blanc	0x0EE0	T
0	0x0EBC	0	MENU	0x0ED4	F

Le module principal exploitant cette donnée devrait activer la ligne IR RQ SEND pour déclencher l'interruption responsable du transfert des données depuis le module infrarouge. Après le transfert des données, IR DATA READY est désactivée.

Chapitre 6 - Module de Contrôle, de commande et de configuration

1. Présentation

Le module de contrôle, de commande et de configuration est le module central qui permet de commander l'ensemble des autres modules. Il doit:

- Etre suffisamment rapide pour adresser tous les modules périphériques,
- Avoir suffisamment de pins pour connecter tous les modules périphériques,
- Avoir assez de mémoire pour stocker les données intermédiaires de chacun de ses périphériques.

Dans son état actuel le système assure la lecture des horaires de prières de la mémoire I2C, la date et l'heure actuelle de l'horloge RTC, l'affichage de la date, de l'heure et des horaires de prière, l'affichage du jour de la semaine actuel, l'interfaçage avec l'afficheur LCD série et la commande Infrarouge.

2. Vue d'ensemble

Le module central reçoit des données de l'horloge RTC, du module infrarouge et des mémoires série. Il commande 8 afficheurs sept segments, un afficheur LCD série et un lecteur MP3 (figure 36).

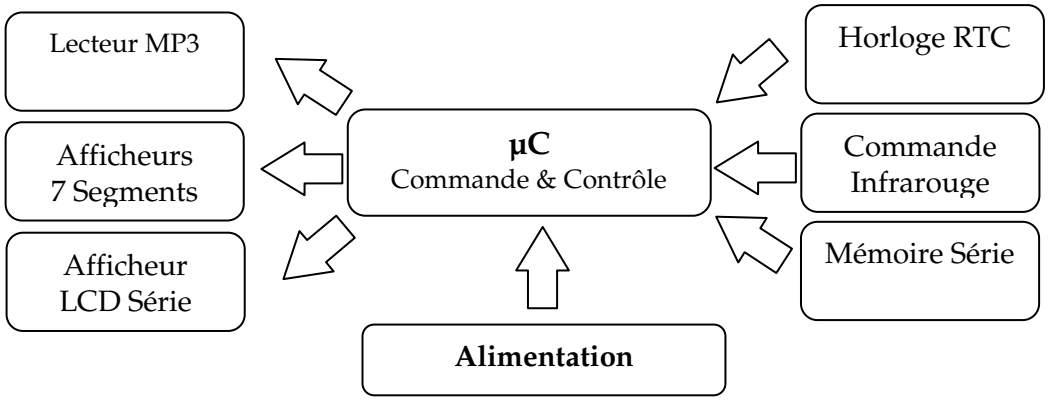


Figure 36, Vue d'ensemble du module central

3. Conception

Le module est construit autour du PIC18F4550 qui comporte 35 pins d'entrées/sorties, 32Ko de mémoire programme et 2Ko de mémoire vive. Ce circuit peut fonctionner sous une fréquence de 48MHz et exécuter 12MIPS. Il possède 4 timers programmables, chacune d'elle pouvant déclencher une interruption à des intervalles réguliers. L'interruption du TIMER0 a été utilisée pour lire la date/heure actuelle de l'horloge RTC et ensuite mettre à jour les afficheurs 7 segments toutes les 350ms. Il faudrait noter que le TIMER0 dans les PIC18F est un compteur 16 bits (8 bits dans la famille 16F).

A chaque minuit ou sous tension du montage, le PIC calcule le nombre de jours écoulés depuis le 01/01/2000 pour retrouver le jour de la semaine correspondant à la date actuelle indiquée par l'horloge RTC. Il retrouve aussi les horaires de prières pour la nouvelle journée et les affiches sur les afficheurs 7 segments.

Actuellement, le PIC scrute l'état du module de commande Infrarouge pour déterminer si l'utilisateur a appuyé sur une touche de la télécommande et affiche le caractère transmis par le module Infrarouge.

J'ai trouvé des difficultés dans les parties suivantes du projet :

- Notion d'évènements pour afficher des messages sur l'afficheur LCD
- Driver du module MP3
- Configuration du système pour sélectionner la date, l'heure actuelles, configuration du son de l'Azan, ainsi que le volume du son, etc.

L'architecture actuelle peut supporter toutes ces fonctions, mais je n'arrive pas à trouver une méthode d'implémentation. Ces parties restent non implémentées.

Le microcontrôleur PIC est incapable de calculer les horaires de prières. Il faudrait concevoir un petit utilitaire qui permet de les calculer et de générer le fichier Hex correspondant.

Chapitre 7 - Développement d'un utilitaire pour le calcul des horaires de prières

1. Introduction

L'utilitaire à développer est uniquement à l'usage du concepteur du système. Il permet de :

- Sélectionner treize villes d'une liste de plus de 25000 à travers le monde
- Créer de nouvelles villes
- Calculer les horaires de prières pour les villes sélectionnées et de les éditer
- Générer le fichier Hex correspondant aux horaires calculés

Le processus de conception : Iconix Process.

Le langage de programmation : Python.

L'interface GUI : wxPython.

Le SGBD : sqlite 3.

a. Iconix Process

Iconix Process[5] est un processus de développement logiciel basé sur le langage UML, il se distingue de ces concurrents et surtout du RUP par sa simplicité. D'abord, il permet au concepteur d'atteindre plus rapidement la phase de développement car avec ce processus, il suffit d'utiliser 20% de la notation UML pour faire 80% du travail.

Le processus Iconix Process (Figure 37) est centré sur les cas d'utilisation, tout comme RUP. Il comporte principalement 4 phases se terminant chacune par une étape de validation : Capture des besoins, Analyse, Conception et Implémentation.

[5] http://iconixprocess.com/

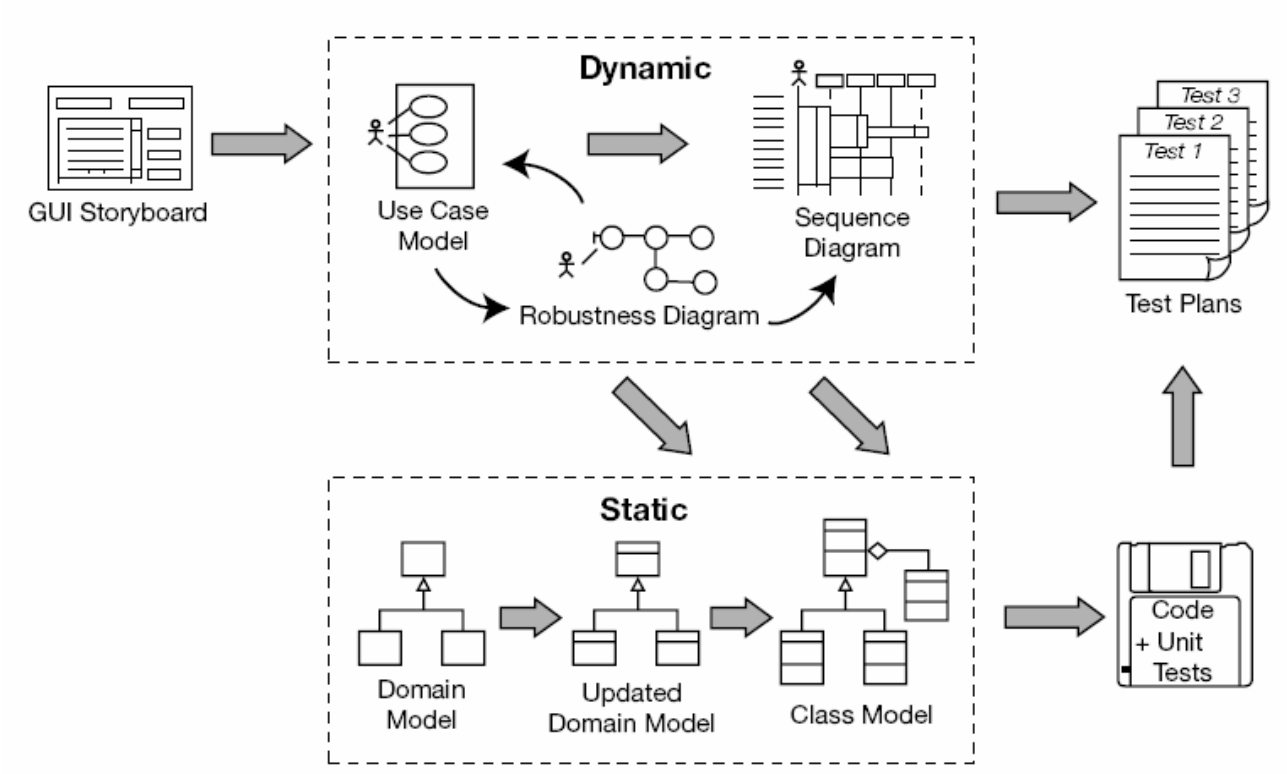


Figure 37, Iconix Process

La capture de besoins a pour objectif de rédiger des cas d'utilisation qui seront utiles lors des étapes d'analyse et de conception. Pour garantir le bon déroulement de cette phase, il est conseillé de :

- Dessiner le modèle de domaine qui permettra d'attribuer des noms pour les objets du système.
- Dessiner les croquis des interfaces graphiques de la future application.
- Rédiger les cas d'utilisation en utilisant la règle des deux paragraphes : chemin principal et chemins alternatifs. Il faut veiller à utiliser la voie active tout en utilisant les termes issus du modèle de domaine et en basant les interactions sur les croquis des interfaces.

La phase d'analyse est basée sur les cas d'utilisations rédigées durant la phase précédente. Durant la phase de l'analyse il s'agira d'extraire les objets utiles pour un cas d'utilisation et d'assigner des responsabilités pour chacun de ces objets. Deux types de diagrammes seront dessinés lors de cette phase :

- Le diagramme de robustesse qui permet de retrouver les objets assurant le fonctionnement d'un cas d'utilisation.

- Le diagramme de séquences qui permet de découvrir et d'attribuer des responsabilités pour chaque objet.

A la fin de la phase d'analyse, le modèle de domaine est mis à jour des attributs et des méthodes extraites des diagrammes précédents y seront insérés.

Dans la phase de conception les classes commencent à prendre forme. D'autres attributs et méthodes seront ajoutés au modèle de domaine qui se transforme en modèle de classe. La mutation est le résultat d'affinement des diagrammes de robustesses et des diagrammes de séquence. La phase d'implémentation est basée sur les tests unitaires pour s'assurer qu'un cas d'utilisation est bien implémenté selon le scénario rédigé et de valider les phases précédentes.

b. Langage Python

Le langage Python[6] est un langage orienté objet, interprété, de l'univers des logiciels open source. C'est un langage simple très utilisé sous le système d'exploitation Linux. Je l'ai utilisé parce qu'il nécessite peu de bibliothèques externes et parce qu'il est très facilement débuggable.

c. Bibliothèque wxPython

wxPython[7] est le portage du Framework graphique wxWindow pour le langage python.

wxWindow est un ensemble de classes qui encapsulent les différents composants d'une interface graphique. Le but ultime de ces classes est d'unifier l'implémentation de ces composants pour plusieurs systèmes d'exploitation. C'est ainsi que les programmes basés sur ce Framework sont alors portables.

d. SQLITE 3

Le SGBD sqlite[8] qui est à sa version 3 actuellement est un SGBD, multi-plateforme, très léger, conçu pour les applications centrées sur un volume de données restreint. Il supporte les requêtes SQL simples et il est adéquat pour notre application pour stocker les coordonnées des villes et les horaires de prières.

[6] http://*www.**python**.org*/
[7] http://www.wxpython.org/
[8] http://www.sqlite.org/

2. Capture de besoins

L'utilisateur (acteur) unique du programme est le **concepteur**. Il a besoin de :

- Sélectionner 13 villes parmi les villes disponibles
- Ajuster les coordonnées et les paramètres des villes sélectionnés
- Générer et éditer les horaires de prières pour ces villes
- Générer le fichier HEX comportant les données qui devront être transférés dans la mémoire I_2C de notre montage.

On identifie les cas d'utilisations suivants :

- Afficher villes sélectionnées
- Ajouter Ville
- Ordonner villes
- Retrouver coordonnés initiales ville
- Sélectionner ville (liste choix)
- Supprimer ville
- Afficher coordonnées villes
- Modifier coordonnées ville
- Afficher horaires de prières
- Générer horaires de prières
- Modifier heures de prières
- Générer fichier Hex

Ces cas d'utilisations seront organisés en 5 packages (Annexe 2.1).

a. Cas d'utilisations paquetage : Sélectionner Villes

Cas d'utilisation : Afficher Villes Sélectionnées

Chemin principal

Le concepteur clique sur le bouton **Sélection des villes** de l'interface **MainFrame**.

Le système crée le dialogue **TownsSelectDialog**.

Le système charge la collection des villes sélectionnées **SelectedVillesCollection**, les affiches dans **lstFinale**, charge la liste des Pays disponibles **country_name**, initialise la liste **lstCountries** avec ces valeurs.

Chemins alternatifs

Le concepteur a consulté la liste des villes sélecionnées

Le concepteur ferme la fenêtre en cliquant sur le bouton btnFermer.

Cas d'utilisation : Ajouter ville

Chemin Principal

Le concepteur sélectionne une ville dans les listes disponibles dans le dialogue TownsSelectDialog. Le système crée une structure contenant les données de la ville sélectionnée criteria, il active, ensuite, le bouton btnAjouter.

Le concepteur clique sur le bouton btnAjouter, le système ajoute la ville ***criteria*** dans la liste finale lstFinale.

Chemins alternatifs

La liste lstFinale contient déjà 13 villes

Le bouton **Ajouter** est désactivé. L'ajout est impossible, le concepteur devra supprimer une entrée de lstFinale.

Cas d'utilisation : Ordonner villes

Chemin Principal

Le concepteur clique sur une ville dans la lstFinale du dialogue TownsSelectDialog. Si l'entrée sélectionnée n'est pas la dernière entrée, le système active le bouton btnDescendre. Le concepteur clique sur ce bouton, le système permute l'élément de la liste sélectionné avec l'élément suivant. Si l'entrée sélectionnée n'est pas le premier élément de lstFinale, le système active le bouton btnMonter. Le concepteur clique sur ce bouton, le système permute l'élément de la liste sélectionné avec l'élément précédent.

Chemins alternatifs

L'élément sélectionné est le premier élément de la liste

Le bouton btnMonter est désactivé

L'élément sélectionné est le dernier élément de la liste

Le bouton btnDescendre est désactivé

Cas d'utilisation : Retrouver coordonnés initiales des villes

Chemin Principal

Le concepteur clique sur le bouton enregistrer du dialogue TownsSelectDialog. Le système crée un objet SelectedVillesCollection, le système détermine les coordonnés des villes figurant dans lstVilles en interrogeant VillesUtility et les ajoute à l'objet SelectedVillesCollection.

Le système enregistre les données.

Chemins alternatifs

Aucun chemin alternatif n’est envisagé.

Cas d'utilisation : Sélectionner ville (liste choix)

Chemin Principal

Le concepteur tape les premières lettres du pays auquel appartient la ville désirée dans edtCountry. Le système cherche via VillesUtility la liste des pays commençant par les lettres tapées dans la liste lstCountry il met à jour cette dernière liste.

Le concepteur clique sur un élément de la liste lstCountry, le système récupère la liste des régions dans ce pays regions_names via VillesUtility et il initialise lstRegions avec les valeurs trouvées.

Le concepteur tape les premières lettres de la région désirée dans **edtRegion**, le système met à jour **regions_names** et **lstRegions**. Le concepteur clique sur le nom d'une région. Le système la liste des villes de cette région **villes_names** à partir de **VillesUtility** et initialise **lstVilles**.

Le concepteur tape les premières lettres de la ville désirée dans **edtVille**, le système met à jour **villes_name** et **lstVilles**. Le concepteur clique sur le nom d'une ville. Le système met à jour **criteria**, récupère les coordonnés de la ville Ville coorespondant à **criteria** et les affiche dans **edtInfo**.

Chemins alternatifs

La ville désirée ne se trouve pas dans la liste de choix

Sélectionner une ville quelconque et éditer ses paramètres par la suite.

Cas d'utilisation : Supprimer ville

Chemin Principal

Le concepteur sélectionne une ville dans la liste lstFinale du dialogue TownsSelectDialog. Le système active le bouton btnSupprimer.

Le concepteur clique sur le bouton btnSupprimer, le système supprime l'élément sélectionné de la liste finale lstFinale.

Chemins alternatifs

Aucune ville sélectionnée, liste vide

Le bouton btnSupprimer est désactivé il n'est pas possible de supprimer.

b. Cas d'utilisations paquetage : Editer coordonnées villes

Cas d'utilisation : Afficher coordonnées villes

Chemin Principal

Le système affiche MainFrame. Le concepteur clique sur le bouton Emplacement des villes….

Le système crée le dialogue VillesDialog, il charge la liste des villes sélectionnées par le concepteur dans la collection SelectedVillesCollection, et affiche la liste de ces villes dans la liste lstVilles.

Chemins alternatifs

Le concepteur n'a plus besoin de la liste des coordonnées

Le concepteur clique sur le bouton btnFermer.

Cas d'utilisation : Modifier coordonnées ville

Chemin Principal

Le concepteur clique sur un élément de la liste des villes lstVilles. Le système récupère le numéro de l'élément sélectionné et initialise l'objet Ville ayant le même indice dans la collection SelectedVillesCollection, il crée le dialogue VilleEditDialog. Le système initialise le dialogue à partir de l'objet Ville.

Le concepteur effectue les changements désirés, il clique sur le bouton btnModifier. Le système initialise un objet Ville à partir des entrées du concepteur, il compare l'objet initial de la collection SelectedVillesCollection, s'ils sont différents le système écrase l'ancienne instance de l'objet et met à jour la liste lstVilles.

Chemins Alternatifs

Le concepteur veut annuler les changements

Il clique sur le bouton btnAnnuler

La valeur éditée et la valeur initiale sont égales

Le système n'effectue aucun changement dans SelectedVillesCollection

c. Cas d'utilisations paquetage : Editer horaires de prières

Cas d'utilisation : Afficher horaires de prières

Chemin Principal

Le système affiche le dialogue MainFrame. Le concepteur clique sur le bouton Edition des horaires de prière.

Le système crée le dialogue VilleChoiceDialog, il initialise ce dialogue par la liste des villes sélectionnées via SelectedVillesCollection.

Le concepteur sélectionne la ville désirée puis clique sur le bouton OK. Le système crée le dialogue PrayersTimesDialog, récupère le numéro de la ville sélectionné, transfère l'objet Ville ayant l'indice correspondant dans SelectedVillesCollection au dialogue et affiche les données du Ville en cours de consultation.

Le système retrouve les horaires de prières de la ville sélectionnée YearPrayersTimes, il initialise la liste lstPrayers avec les données de la collection.

Chemins alternatifs

Le concepteur clique sur annuler dans VilleChoiceDialog

Le cas d'utilisation est abandonné

Le concepteur n'a plus besoin d'éditer les horaires de prières

Il clique sur le bouton btnQuitter

Cas d'utilisation : Générer horaires de prières

Chemin Principal

Dans le dialogue PrayersTimesDialog le concepteur clique sur le bouton btnGenerer.

Le système demande la confirmation du concepteur. En cas de confirmation, le système régénère la liste YearsPrayersTimes à partir du calcul basé sur les coordonnés géographique et les données contenues dans l'objet Ville, il sauvegarde le résultat et réinitialise la liste lstPrayers avec les nouvelles valeurs.

Chemins alternatifs

Le concepteur ne confirme pas l'opération

Le système conserve les valeurs initiales des heures de prières

Cas d'utilisation : Modifier heures de prières

Chemin Principal

Le concepteur clique sur un élément de la liste des horaires lstPrayers. Le système récupère le numéro de l'élément sélectionné et initialise l'objet PrayersTimes ayant le même indice dans la collection YearPrayersTimes, il crée le dialogue PrayersEditDialog. Le système initialise le dialogue à partir de l'objet PrayersTimes.

Le concepteur effectue les changements désirés, il clique sur le bouton btnValider. Le système initialise un objet PrayersTimes à partir des entrées du concepteur, il compare l'objet initial de la collection YearPrayersTimes, s'ils sont différents le système écrase l'ancienne instance de l'objet et met à jour la liste lstPrayers.

Chemins alternatifs

Le concepteur veut annuler les changements

Il clique sur le bouton btnAnnuler

La valeur éditée et la valeur initiale sont égales

Le système n'effectue aucun changement dans YearPrayersTimes

d. Cas d'utilisations paquetage : Prayers Caller

Cas d'utilisation : Générer fichier Hex

Chemin Principal

Dans l'interface MainFrame, le concepteur clique sur le bouton Générer fichier Hex. Le système récupère la liste des villes sélectionnées SelectedVillesCollection, il récupère les YearPrayersTimes pour chacune des villes et génère le code Hex correspondant et enregistre le résultat.

Chemins alternatifs

Le concepteur n'a pas généré les horaires de certaines villes

Les horaires sont générés automatiquement par le système

3. Analyse

Dans la phase d'analyse il s'agit de dessiner les diagrammes de robustesses et les diagrammes de séquences pour chaque cas d'utilisation.

a. Diagrammes de robustesse

Les diagrammes de robustesses permettent de lever toute ambiguïté du texte des Use cases. L'enchainement dans ces diagrammes est décrit par le texte figurant dans une note UML à droite dans chacune d'elles.

Le diagramme de robustesse du cas d'utilisation : afficher villes sélectionnées (figure 38).

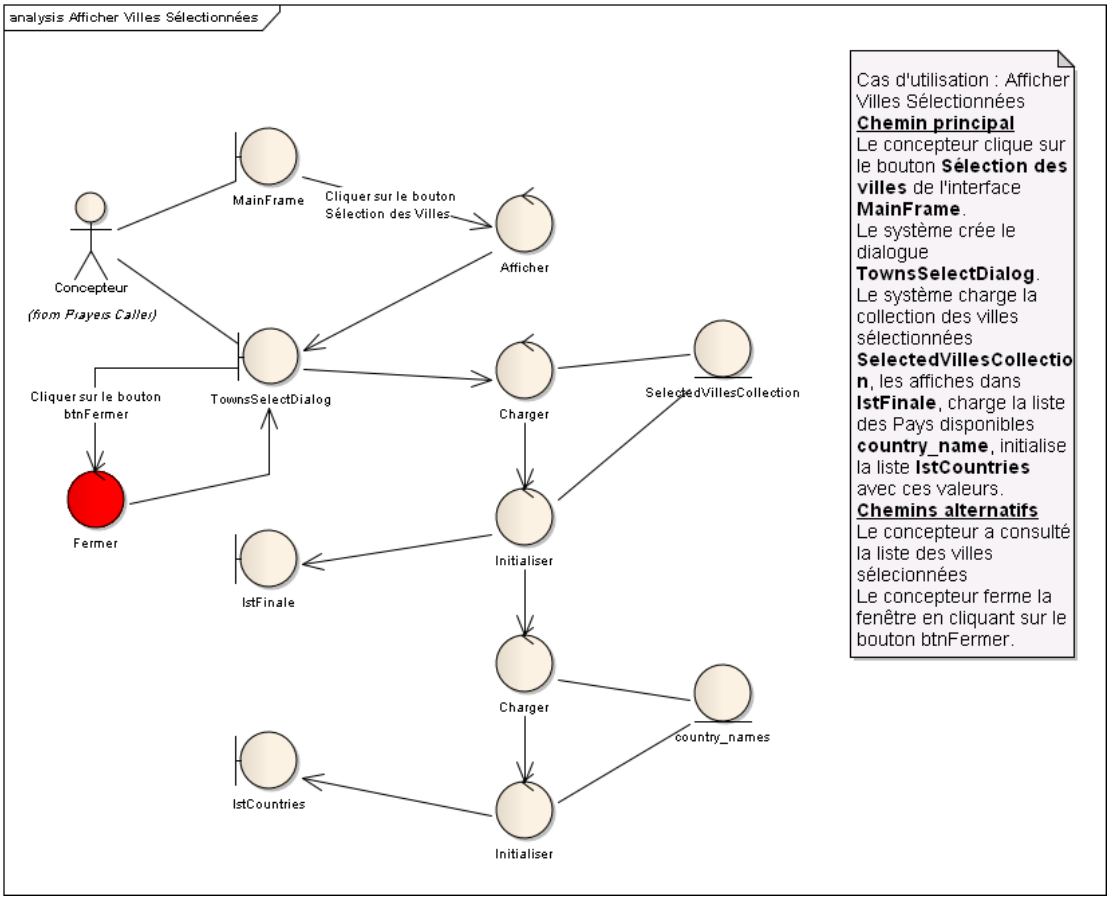


Figure 38, diagramme de robustesse Afficher villes sélectionnées

Le diagramme de robustesse pour le cas d'utilisation : Ajouter ville (figure 39).

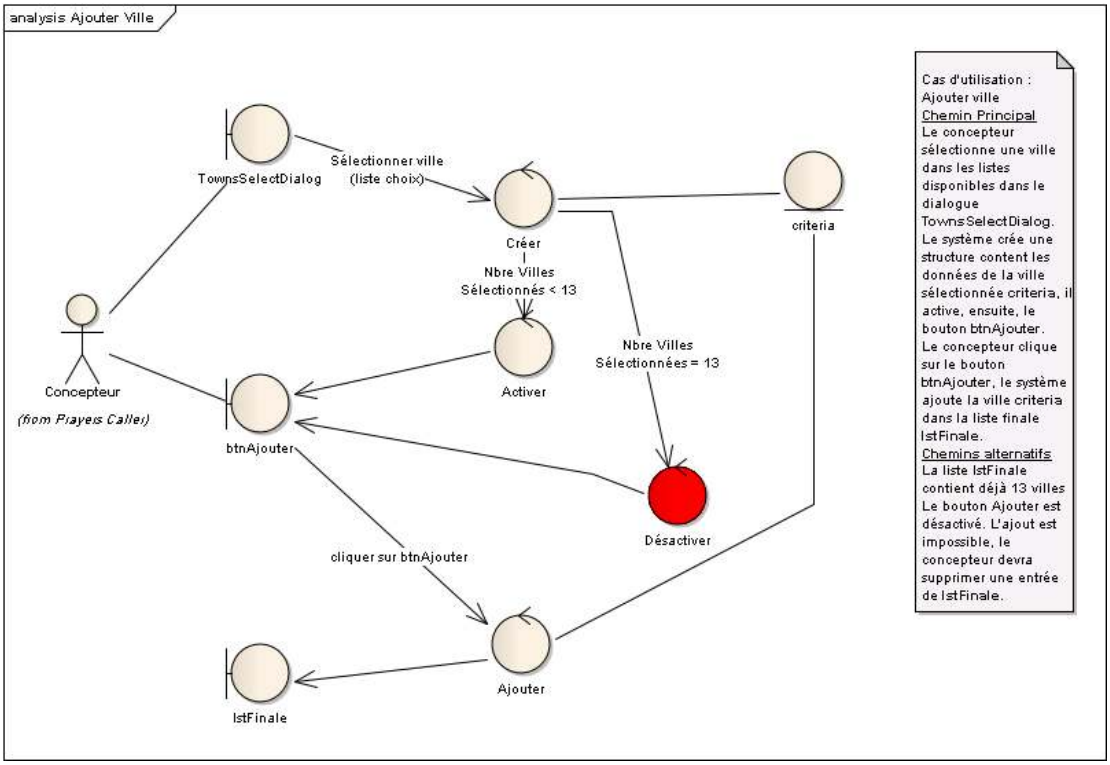


Figure 39, Diagramme de robustesse Ajouter ville

Le diagramme de robustesse du cas d'utilisation : Ordonner villes (figure 40).

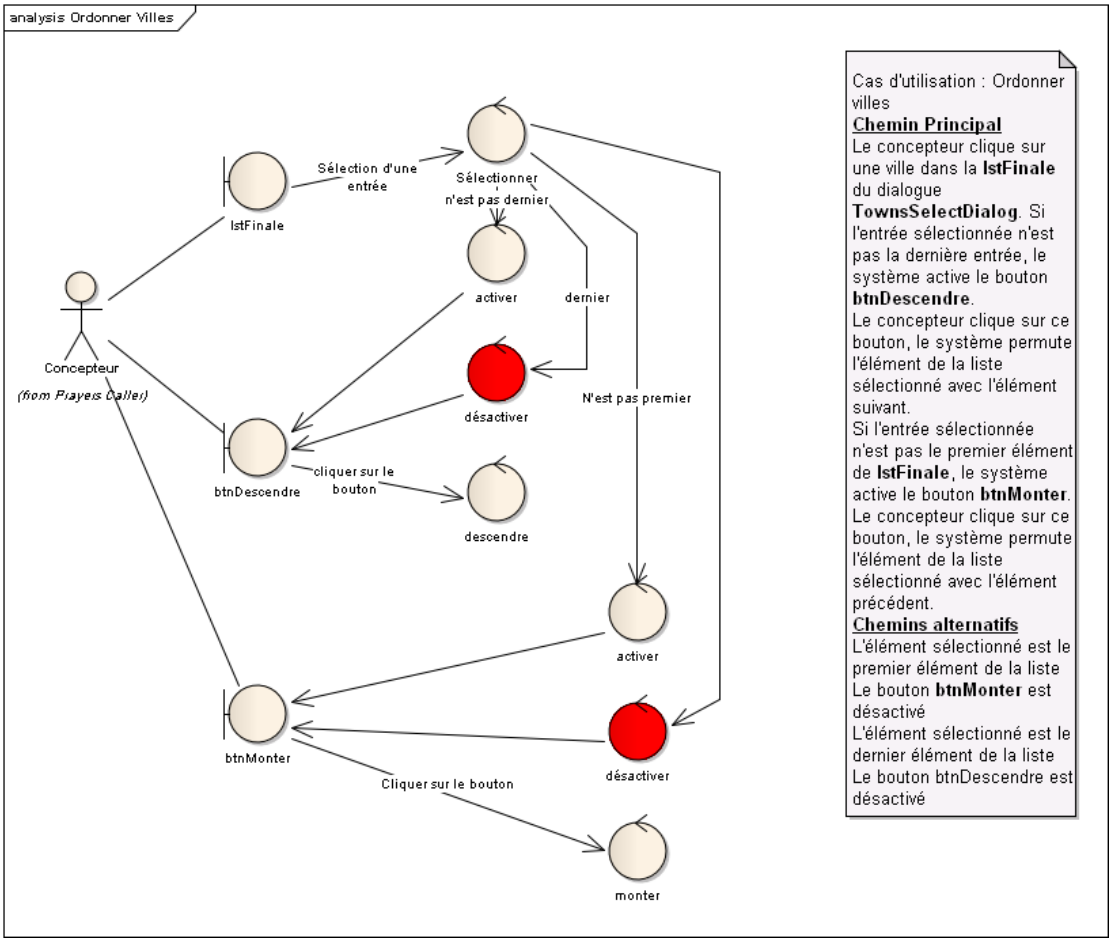


Figure 40, Diagramme de robustesse : Ordonner Villes

Le diagramme de robustesse du cas d'utilisation : retrouver coordonnées initiales des villes (figure 41).

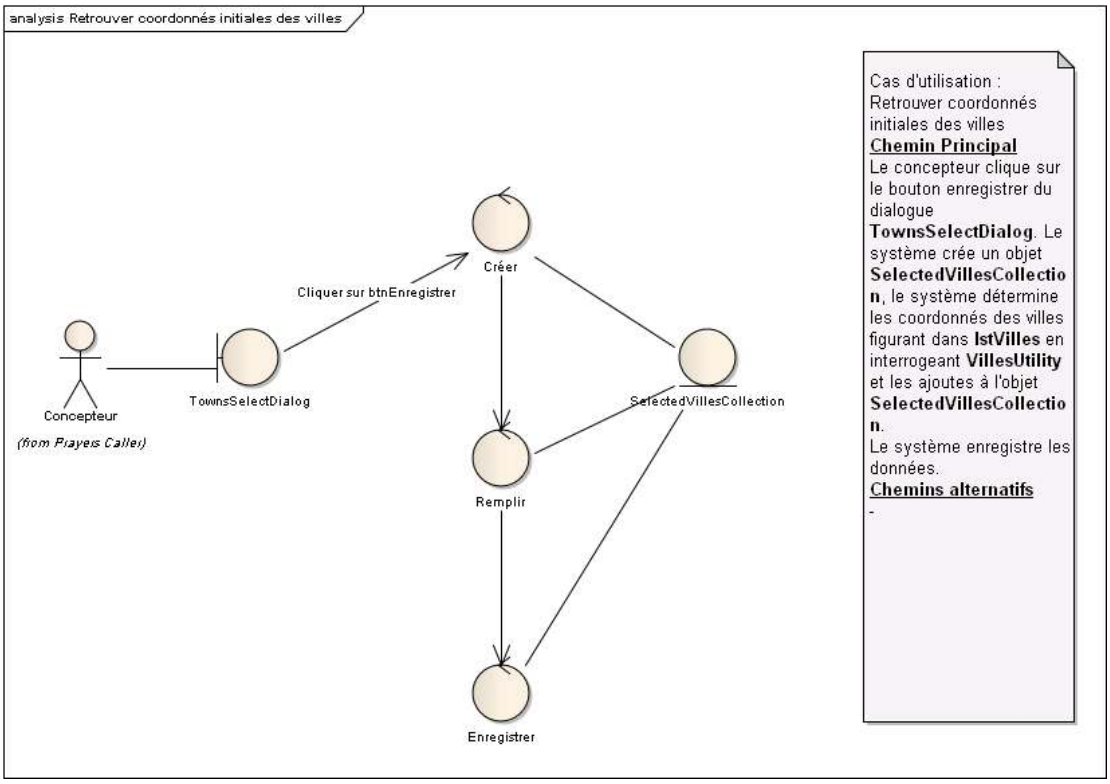


Figure 41, Diagramme de robustesse : retrouver coordonnées initiales des villes

Le diagramme de robustesse du cas d'utilisation : Supprimer ville (figure 42).

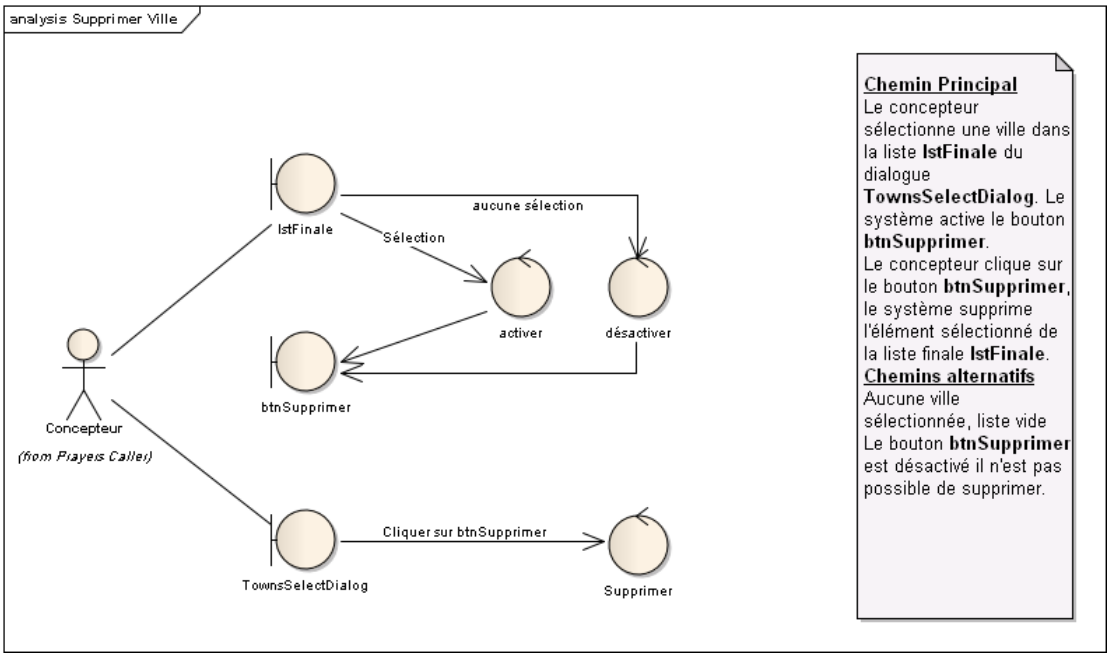


Figure 42, Diagramme de robustesse : Supprimer ville

Le diagramme de robustesse du cas d'utilisation Sélectionner ville (figure 43).

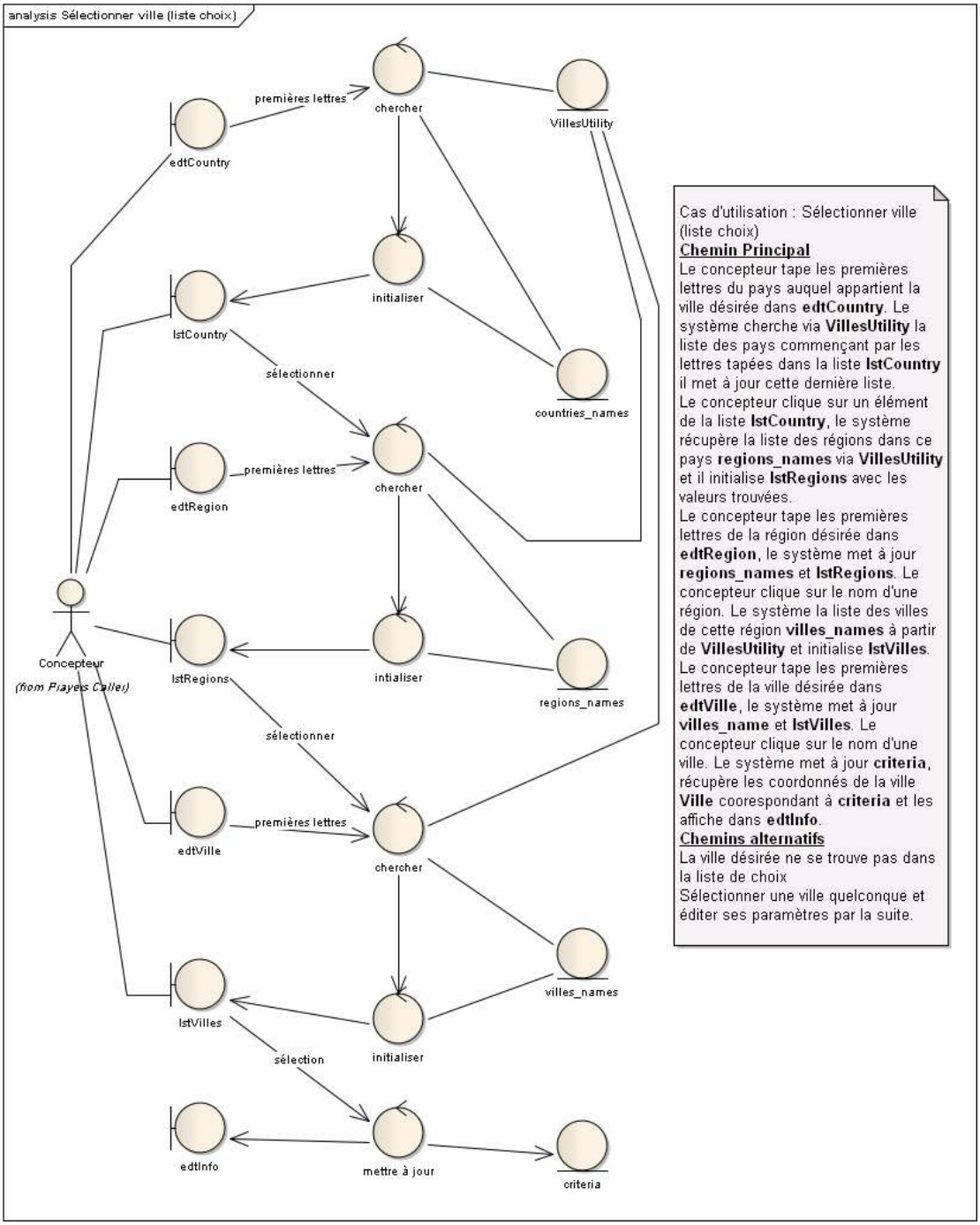


Figure 43, Diagramme de robustesse : Sélectionner ville (liste choix)

b. Diagrammes de séquence

Les diagrammes de séquence permettent d'allouer un comportement aux objets participant dans les différents Use cases.

Ces comportements deviendront des méthodes dans les classes correspondantes dans le diagramme de classe (figure 49).

Diagramme de séquence pour le cas d'utilisation Afficher villes sélectionnées (figure 44).

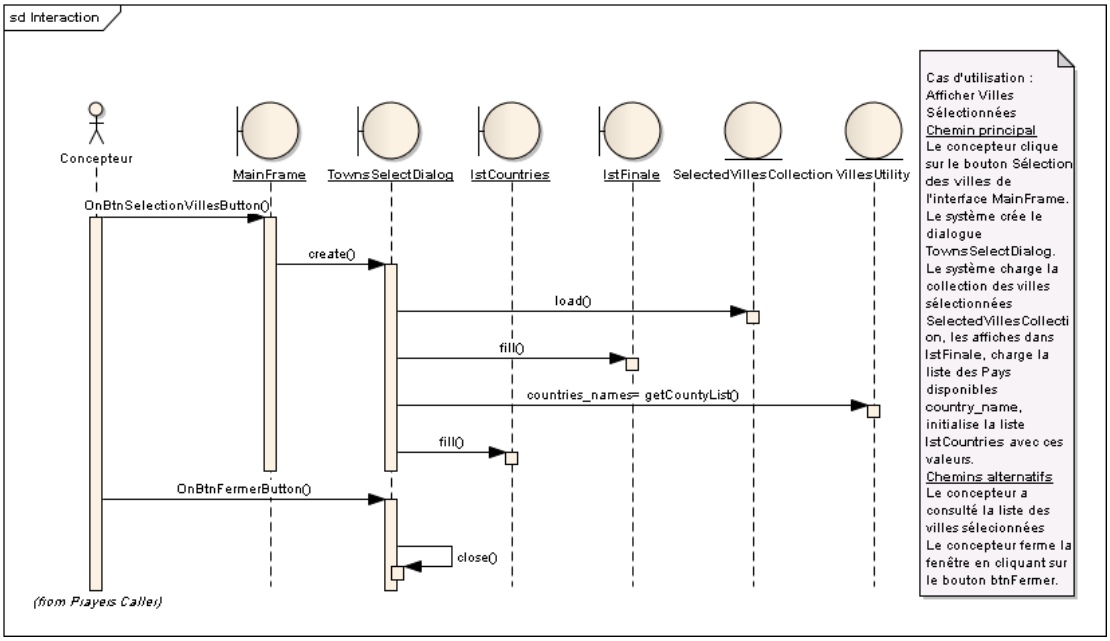


Figure 44, Diagramme de séquences : Afficher villes sélectionnées

Diagramme de séquence pour le cas d'utilisation Ordonner villes (figure 45).

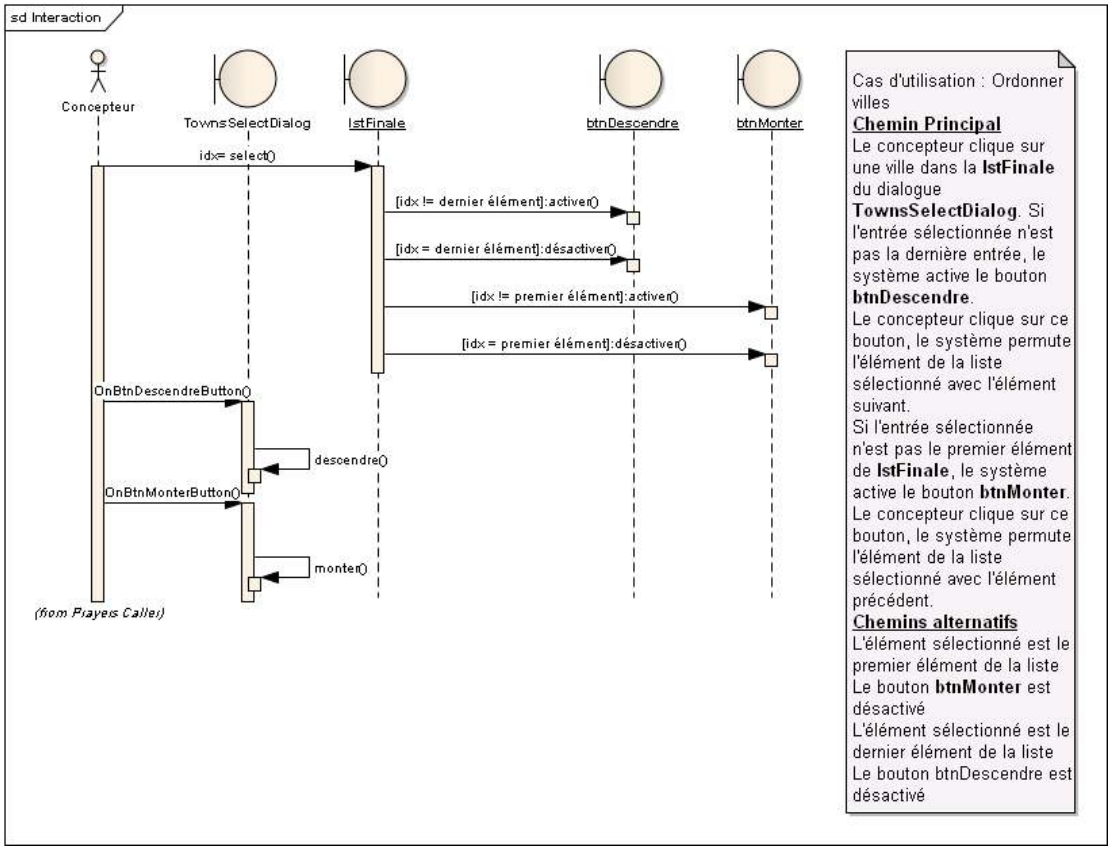


Figure 45, Diagramme de séquences : Ordonner villes

Diagramme de séquence pour le cas d'utilisation Retrouver coordonnées initiales des villes (figure 46).

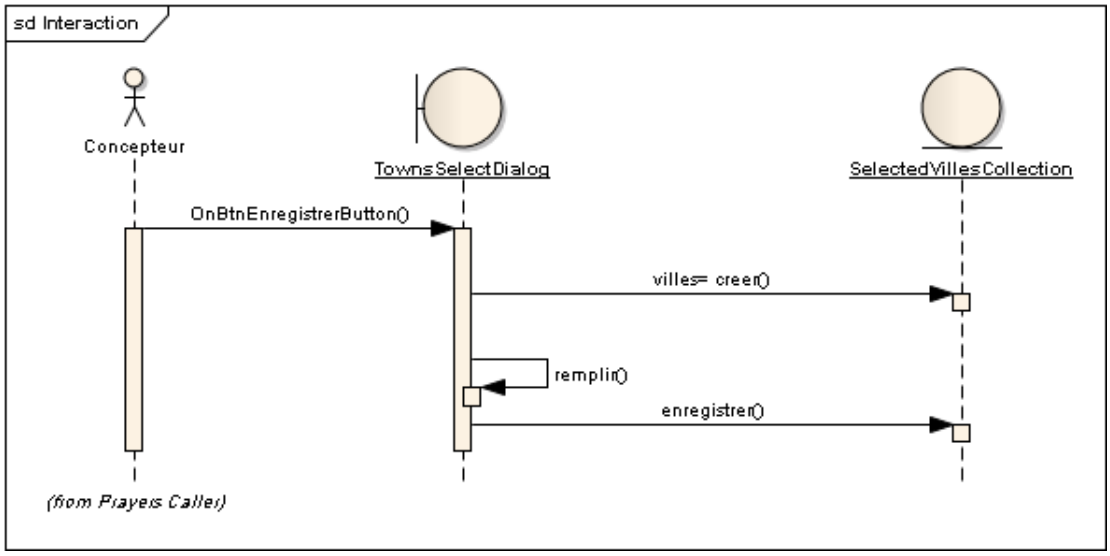


Figure 46, Diagramme de séquences : Retrouver coordonnés initiales des villes

Diagramme de séquence pour le cas d'utilisation supprimer ville (figure 47).

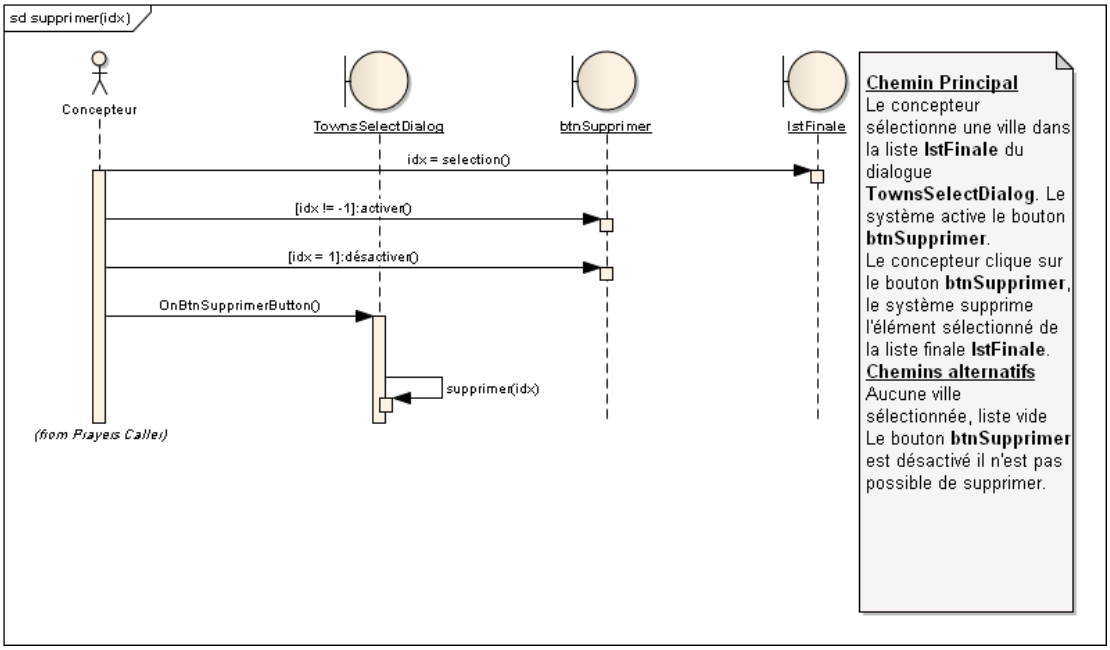


Figure 47, Diagramme de séquence : supprimer ville

Le diagramme de classes relatif au package Sélectionner ville est présenté dans la figure 47. Les autres diagrammes ne seront pas donnés dans ce rapport.

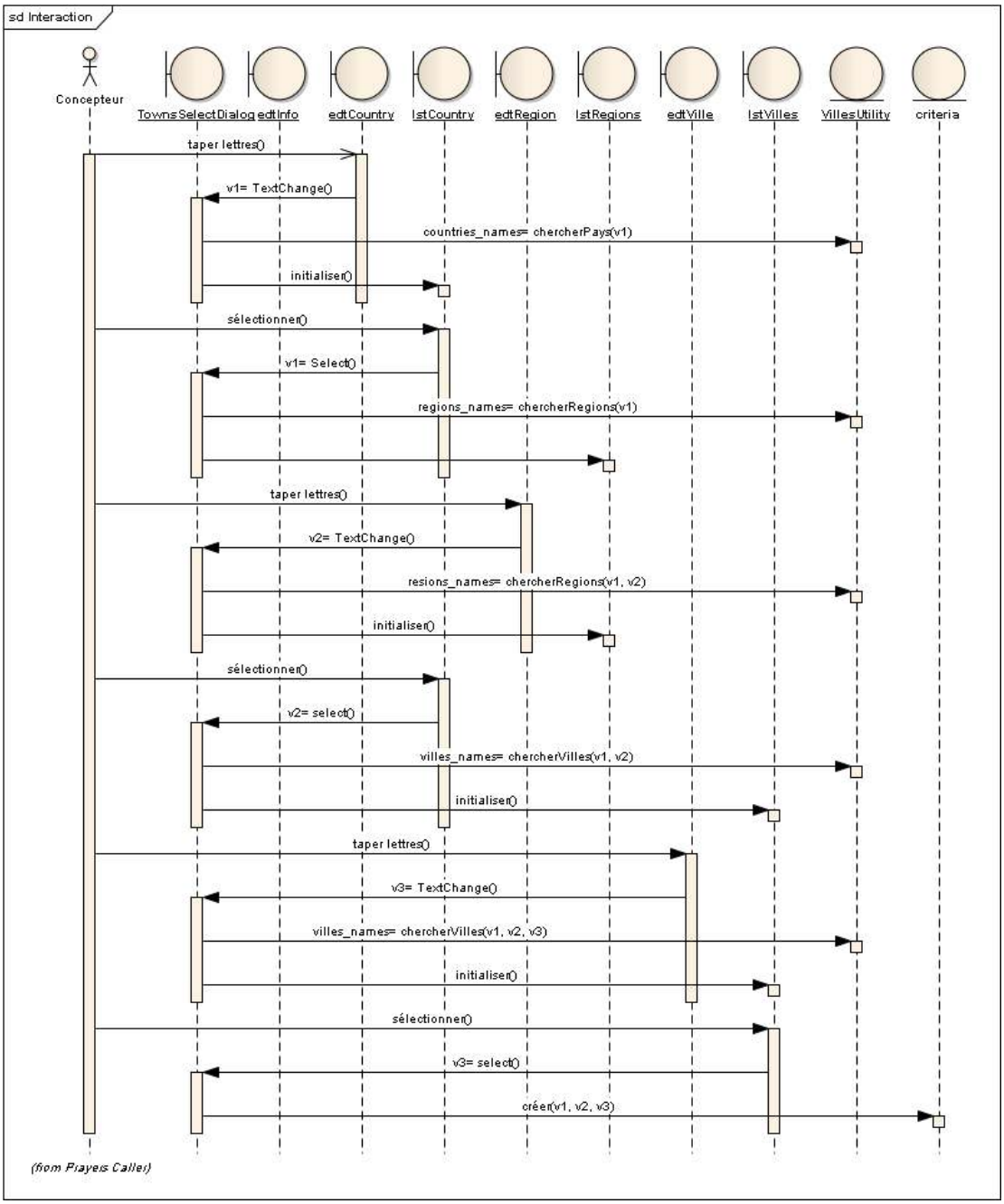


Figure 48, Diagramme de séquence : Sélectionner ville (liste choix)

4. Conception

"Analysis is about building the right system. Design is about building the system right. [9]"

L'analyse permet construire le bon système. La conception permet de construire le système correctement.

Pour se faire il faudra dessiner le diagramme de classes à partir du modèle de domaine (figure 49), des diagrammes de séquences, des diagrammes de robustesses et des cas d'utilisation.

Le nom des classes qui serviront de squelette pour notre application provient du modèle du domaine ainsi que quelques attributs.

L'interaction entre ces classes ainsi que les méthodes et les attributs qui les constituent sont déduits des diagrammes de robustesses.

L'enchaînement d'appel des méthodes est impliqué par les diagrammes de séquences.

[9] Use Case Driven Object Modeling with UML, Theory and Practice, Doug Rosenberg and Matt Stephens, Apress

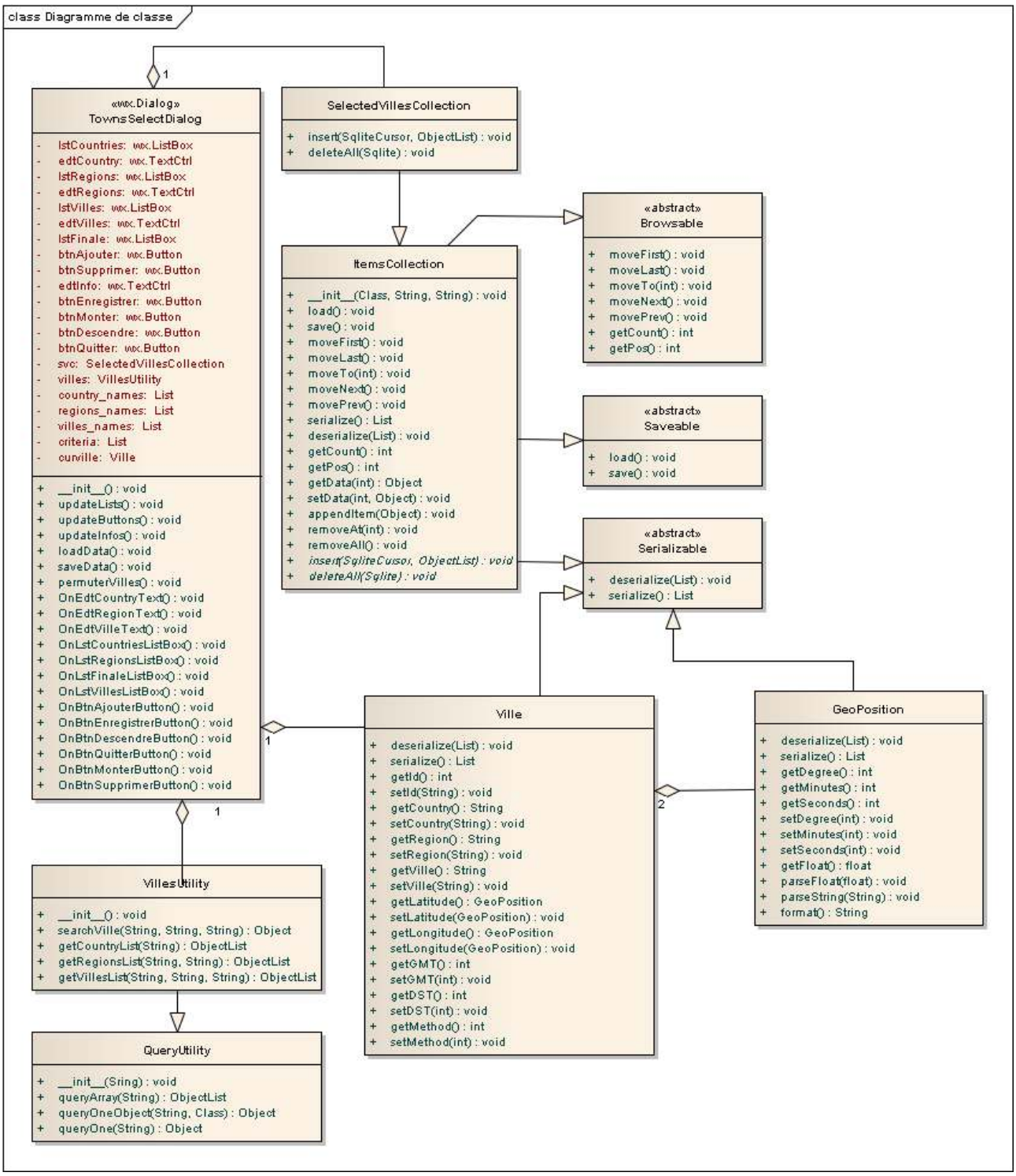


Figure 49, Diagramme de classes du paquetage : Sélectionner Villes

5. Implémentation

L'implémentation du programme sera réalisée avec le langage Python. Le code source est donné sur le support numérique.

Conclusion

Ce projet m'a donné l'occasion pour acquérir de nouvelles connaissances en électronique, utiles pour réaliser des montages à base de microcontrôleurs PIC et découvrir de nouvelles technologies : I_2C, RS232, mémoires EEPROM, horloge RTC, RC5, etc.

Durant ce projet j'ai eu l'occasion d'exploiter toutes ces technologies et de découvrir de nouvelles techniques, notamment, la réalisation des circuits imprimés, la soudure des composants, le calcul des résistances de polarisation des transistors, l'interfaçage d'un lecteur MP3, les mécanismes de protection de certains composants, etc.

J'ai, aussi, utilisé un nouveau processus de développement logiciel basé sur UML, ICONIX PROCESS. D'autant plus, j'ai essayé d'appliquer les phases de ce processus selon ma modeste compréhension de la langue anglaise.

J'ai, encore, utilisé le langage de programmation orienté objet Python, la bibliothèque graphique wxPython et le SGBD sqlite3 pour garantir le portage de mon travail sous un autre système d'exploitation.

J'avoue, enfin, qu'il reste encore du travail pour achever Prayer Caller. Il pourrait faire le sujet d'un autre mémoire de fin d'études.

Au terme de ce mémoire, je souligne l'importance de la mise en commun des sciences de la technologie et de l'informatique puisqu'elles ouvrent de nouveaux horizons de recherche et de développement.

Bibliographie

Use Case Driven Object Modeling with UML: Theory and Practice - Doug Rosenberg (Author), Matt Stephens (Author)

Webographie

Composants électroniques - datasheets - http://www.datasheetcatalog.net/

Informations sur les horaires de prières - http://www.namazvakti.com/fr.1.pdf

Iconix Process - http://iconixprocess.com/

Microcontroleur PIC - datasheets - http://www.microchip.com/

Python - http://python.org/

SQlite 3 - http://www.sqlite.org/

Techniquesde soudure - http://www.interface-z.com/conseils/soudure.htm

wxPython - http://www.wxpython.org/