PST 2A : RAPPORT DE PROJET SCIENTIFIQUE ET TECHNIQUE

PST REVEIL INTELLIGENT

L'équipe :

Paul BOSSIERES

Florent NIGET

Naïm TANY

Ibrahim LABIOD

Suivi par :

M. Sébastien GAGEOT

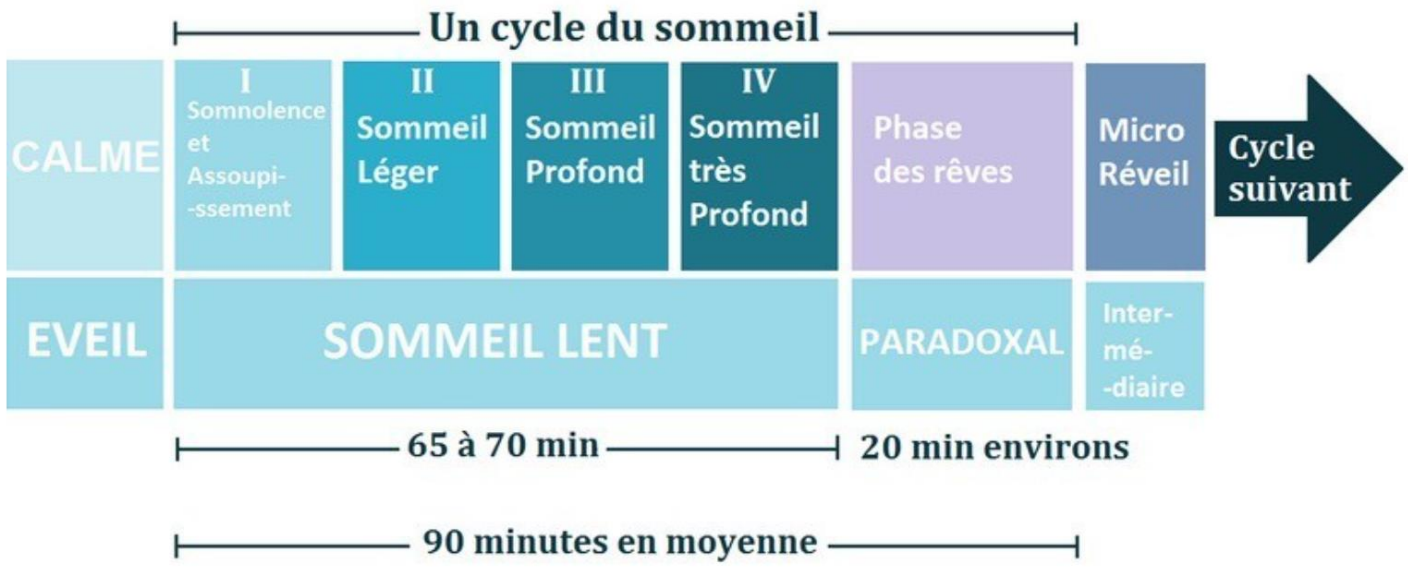
SOMMAIRE

PRESENTATION DU PROJET

Réveil Intelligent

Pourquoi avons-nous besoin de mieux nous réveiller ?

Dans notre société moderne, le sommeil est souvent négligé. On ne s'intéresse pas à ses mécanismes, on dort parce qu'il faut dormir. On pourrait pourtant améliorer la qualité de nos nuits sans pour autant dormir beaucoup plus. Beaucoup de personnes se réveillent à heure fixe, en fonction de leur temps de préparation le matin. Mais parfois, il vaut mieux dormir moins longtemps, pour se sentir plus en forme.



Le sommeil fonctionne par cycles, et dans ces cycles, il y a deux types de sommeil. Le profond et le léger. Le but de ce projet est de créer un réveil qui nous tire des bras de Morphée dans une phase adaptée. Le risque de faire sonner son réveil pendant la phase de sommeil profond est d'abord, de ne pas entendre le réveil, mais ensuite, de mettre du temps à émerger de son sommeil et de se sentir endormi une bonne partie de la journée.

Le projet

- La base du projet

L'idée est de créer un réveil, avec lequel on communique avant de se coucher. On prévient le réveil que l'on va se coucher, on donne une estimation de notre temps d'endormissement en fonction de notre niveau de fatigue. Ensuite, le réveil propose une multitude d'heures de lever en fonction de la durée de notre cycle de sommeil que l'on peut configurer auparavant. On choisit une heure de lever en fonction du temps que l'on a besoin le matin. Cela peut amener à se lever plus tôt. Ensuite le réveil sonne à l'heure choisie.

Le réveil utilise pour communiquer des boutons et un afficheur 7 segments. Une application mobile peut se synchroniser avec le réveil pour lui donner ces instructions. L'application permet aussi de présenter à l'utilisateur diverse données sur son sommeil et ses préférences.

- L'équipe

L'équipe autour de ce projet est constituée de quatre membres :

Paul Bossières

Florent Niget

Naïm Tany

Ibrahim Labiod

La partie conception du réveil en lui-même, gestion de l'heure, de l'affichage, des boutons, est attribuée à Paul Bossières et Naïm Tany. Le développement de l'application mobile liée au réveil est attribué à Florent Niget et Ibrahim Labiod.

ETAT DE L'ART

- Le sommeil

Le sommeil n'est pas continu, il est rythmé par plusieurs cycles successifs, environ 4 à 6 par nuit. Un cycle dure environ 90 minutes et est composé de 3 différents types de sommeil :

• sommeil lent léger
• sommeil lent profond (partie récupératrice)
• sommeil paradoxal (partie dans laquelle on rêve)

Le sommeil lent comporte donc différentes parties. Pendant le sommeil lent léger, l'activité du cerveau ralentit petit à petit pour arriver au sommeil lent profond. Pendant cette partie, l'activité cérébrale est faible, on dort profondément et il est dur d'être réveillé. Le sommeil lent dure environ 60 à 75 minutes. Ensuite arrive le sommeil paradoxal Pendant cette partie, l'activité cérébrale est forte et on peut observer des mouvements oculaires rapides. Les muscles du corps sont paralysés afin de ne pas reproduire les mouvements de nos rêves (même si ce système peut faire défaut, cela donne le somnambulisme). Le sommeil paradoxal dure lui entre 15 et 20 minutes.

Données tirées d'un article du Dr Jean Philippe Rivière et révisé par le Dr Jesus Cardenas.
http://www.doctissimo.fr/html/psychologie/bien_dormir/ps_6205_sommeil_cycles.htm

- Les réveils

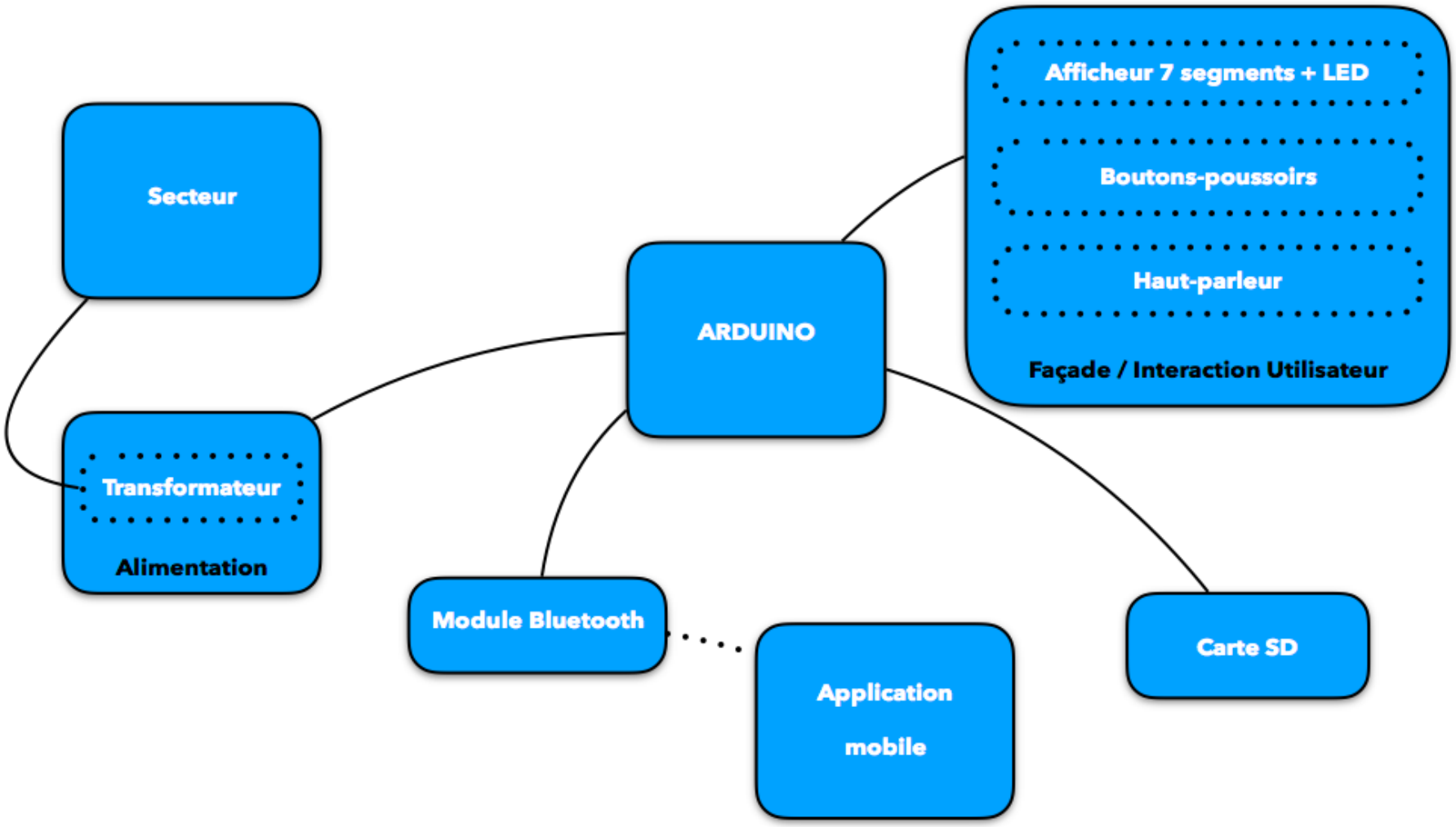
La plupart des réveils vendus dans le commerce permettent de nous réveiller à une heure qu'on fixe. Le problème c'est que si l'heure de réveil tombe pendant notre sommeil profond, il est possible qu'on ne l'entende pas, et si on l'entend, on met du temps à émerger et on a encore l'impression de dormir pendant une bonne partie de la journée. Le but de notre réveil intelligent est de prendre en compte les cycles de sommeil de utilisateur afin de lui proposer des heures de lever optimales.

Certains réveils utilisant en partie ce principe existent. Par exemple le « Iwaku » ou l' « Aura » de Withings. Ces derniers coûtent plus de 250€. Le premier fonctionne avec le téléphone comme capteur de mouvement pour déterminer les phases du sommeil de l'utilisateur. Il implique donc l'utilisation d'ondes or nous souhaitons les éviter pendant que l'utilisateur dort. Le second fonctionne avec un capteur de mouvement à placer sous le matelas. L'installation n'est pas forcément simple.

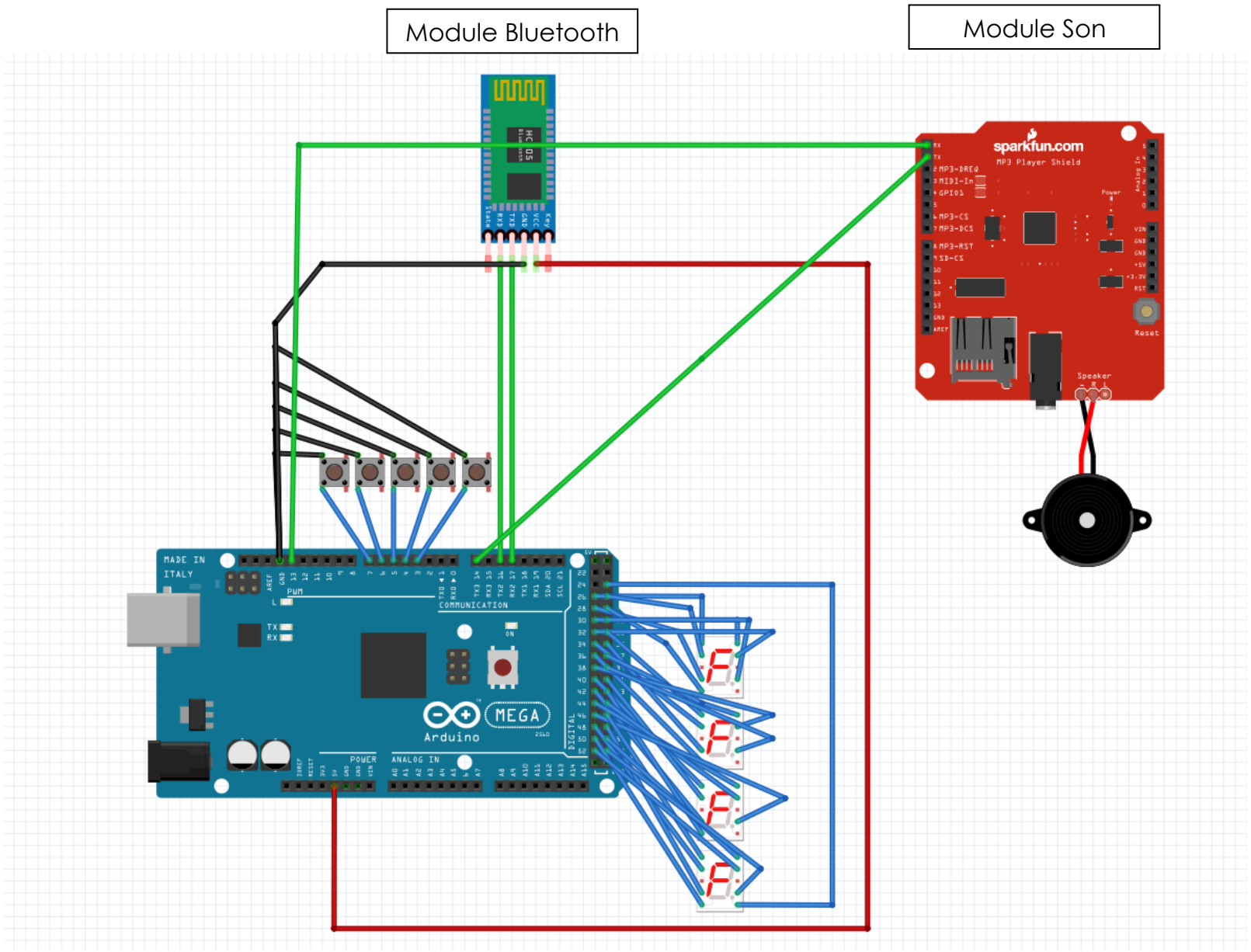
Notre réveil se veut plus simple et moins onéreux. Pour cela il se basera sur la durée du cycle de sommeil de l'utilisateur. De plus, il pourra se synchroniser avec une application mobile pour voir des données sur notre sommeil, déterminer la durée de son cycle de manière empirique, ou choisir une musique de réveil.

Informations tirées de sites marchands ou de "les numériques" pour les revues des réveils intelligents.

SCHEMA FONCTIONNEL



SCHEMA DU CABLAGE



On utilise en plus un Relais pour commander le module Bluetooth à l'aide d'un bouton.

PARTIE ELECTRONIQUE

Le programme Arduino a été conçu de façon modulaire et non bloquante, c'est à dire que la boucle principale ne s'interrompt jamais, même pour demander des informations à l'utilisateur. Ci-dessous une description simplifiée de son fonctionnement, le détail peut se parcourir sur l'annexe (voir clé USB).

- Variables de fonctionnement

```
//structures fonctionnement
Mode mode;
Date date;
unsigned char cycle= 90;
unsigned char endormissement = 0;
unsigned char choix=0;
Date alarme;
Date choixAlarme[NBR_CHOIX_ALARME];
unsigned char alarmeActive = 0;
```

Le réveil fonctionne avec des modes, comme le mode « affichage heure », « réglage alarme », « réglage cycle », et d'autres. Ces modes déterminent le comportement du réveil, car les boutons n'ont pas le même comportement à tout endroit de l'application. Par exemple le bouton – dans « réglage minute » ne fait pas la même chose que dans « réglage heure ». Il détermine aussi ce qui doit être afficher à l'écran et d'autres, c'est géré dans le fichier « interface.cpp », chaque mode amène à une « interface » (boutons + affichages + actions). Le réveil bien sûr stocke la date actuelle (année, mois, jour, heure, minute, seconde), la durée d'un cycle de l'utilisateur, la date de l'alarme, et si l'alarme est activée. Les variables « endormissement », « choix » , « choix alarme », servent à proposer les heures de lever lors du paramétrage de l'alarme.

```
typedef struct AppuiBouton_d
{
    char pin[NBR_BOUTONS];
    char appuiCourt[NBR_BOUTONS];
    char appuiLong[NBR_BOUTONS];
    unsigned long tAppui[NBR_BOUTONS];
}AppuiBouton;
```

Pour gérer les boutons, le programme comporte une structure qui permet de savoir si un bouton est appuyé par un appui long ou un appui court. Dans la case appuiCourt[3], on a l'information 1 si le bouton 3 a été appuyé brièvement, 0 sinon.

```c
typedef struct Afficheur7Seg_d
{
    char pin[NBR_PIN_7SEG];
    char etat[NBR_PIN_7SEG];
}Afficheur7Seg;
```

Pour gérer les afficheurs 7 segments, le programme utilise une structure intermédiaire pour chaque afficheur. Chaque partie du programme qui veut afficher une information à l'écran change un booléen dans le tableau « etat » qui contient les informations des segments (1 s'il faut l'allumer, 0 sinon). Bien sûr, pour simplifier la tâche, il y a une fonction qui permet de « décoder » un chiffre en segments. À chaque tour de boucle, une fonction prend les informations dans le tableau « etat » change l'état des pins correspondant.

```c
//gestion du temps
unsigned long nbRollover = 0;
unsigned long previousMillis=0;
unsigned long long previousDefiler=0;

//gestion bluetooth
String messageBluetooth;
char pinRelaisModuleBluetooth = 2;
unsigned char etatRelaisModuleBluetooth = 0;
unsigned long debutBluetooth = 0;

//gestion son
DFRobotDFPlayerMini son;
```

Dans la partie « gestion du temps », des variables nécessaires pour gérer le dépassement de mémoire de l'Arduino. Le problème est que pour compter le temps qui passe, on utilise le temps (en milliseconde) qui s'est écoulé depuis l'allumage de la carte. Or au bout d'un certain temps, il ne peut plus se stocker et il revient à 0. Cela arrive au bout de 50 jours environ, mais dans notre utilisation, le réveil doit pouvoir fonctionner correctement au-delà et ne pas se décaler, donc on détecte le dépassement de mémoire et calculons spécialement le temps passé.

D'autres variables servent à activer le module Bluetooth et recevoir les messages de l'application. La seule librairie externe du programme sert pour le son.

- Boucle principale

```
void loop()
{
    //gestion des boutons
    detectionAppuiBouton(&appuiBouton);
```

Au début de la boucle principale, on met à jour la structure des appuis boutons. Les appuis courts repassent à 0 si le bouton n'est plus réemployé. Le temps écoulé entre le début de l'appui et la fin de l'appui sert à déterminer si le bouton est appuyé longuement ou brièvement. Cela conditionnera les actions futures dans les différentes interfaces.

```
    //gestion affichage
    switch (mode)
    {
    case affichageHeure:
        interfaceAffichageHeure(&mode,da
        break;
    case reglageHeure:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageMinute:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageJour:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageMois:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageAnnee:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageAuto:
        interfaceReglageHeure(&mode,&dat
        break;
    case reglageCycle:
        interfaceCycle(&mode,&cycle,&pre
        break;
    case reglageEndormissement:
        interfaceReglageAlarme(&mode,cyc
        break;
    case reglageAlarme:
        interfaceReglageAlarme(&mode,cyc
        break;
    case sonne:
        interfaceSonne(&mode,date,&alarm
```

Ça passe par un simple switch-case, on se dirige vers l'interface du mode dans lequel on est. La fonction interface va interpréter les appuis boutons pour aller dans un autre mode spécifique ou changer la durée du cycle ou la date…

Par exemple l'interface pour le réglage du cycle, on commence par afficher le cycle à l'écran et le programme guette un appui sur un bouton. On passe dans cette fonction à chaque tour de boucle de l'algo principal tant qu'on est dans ce mode, on ne bloque pas en attendant les actions de l'utilisateur car il ne faut pas oublier que derrière le temps passe, qu'une alarme peut sonner ou recevoir une information de l'application mobile via bluetooth. Après l'appui sur le bouton de validation, il change lui-même le mode pour redonner la main.

```
void interfaceCycle(Mode *mode,unsigned char *cycle,unsigned long *previousDefileme
{
    affichageCycle(*cycle,aff);
    if(bouton.appuiCourt[3])
    {
        *mode=affichageHeure;
    }
    if(bouton.appuiCourt[2] && *cycle < MAX_CYCLE)
    {
        (*cycle)++;
    }
    if(bouton.appuiLong[2] && *cycle < MAX_CYCLE)
    {
        if(abs(millis()-*previousDefilementRapide > TEMPO_DEFILEMENT_RAPIDE))
        {
            (*cycle)++;
            *previousDefilementRapide = millis();
        }
    }
    if(bouton.appuiCourt[1] && *cycle > MIN_CYCLE)
    {
        (*cycle)--;
    }
    if(bouton.appuiLong[1] && *cycle > MIN_CYCLE)
    {
        if(abs(millis()-*previousDefilementRapide > TEMPO_DEFILEMENT_RAPIDE))
        {
            (*cycle)--;
            *previousDefilementRapide = millis();
        }
    }
}
```

Revenons à la boucle principale.

```
//vérification alarme
if(alarmeActive)
{
    if(datesEgales(date,alarme))
    {
        son.play(NUM_ALARME);
        mode=sonne;
        alarmeActive=0;
    }
}
```

```
//maj etat module bluetooth
majRelaisModuleBluetooth(appuiBouton,pinRelaisModuleBluetooth,&etat
//si le module est activé
if(etatRelaisModuleBluetooth)
{
    //véfification si message bluetooth arrivé
    if(receptionMessageBluetooth())
    {
        messageBluetooth = recuperationMessageBluetooth();
        if(abs(millis()-debutBluetooth)>TEMPO_RECEPTION)
        {
            //on passe l'alarme à 1
            alarmeActive = 1;
            //transformation de la chaine formatée en alarme
            alarme = dateHeureMinuteBluetooth(messageBluetooth);
            ajouterChampsAnneeMoisJour(&alarme,date);
        }
    }
}
```

À chaque tour de boucle, il faut bien sûr vérifier si une alarme doit sonner ou qu'un message Bluetooth arrive.

```
//affichage des afficheurs
afficher7Seg(*(afficheurs[0]));
afficher7Seg(*(afficheurs[1]));
afficher7Seg(*(afficheurs[2]));
afficher7Seg(*(afficheurs[3]));
```

On actualise les afficheurs.

```
//défilement du temps si pas en cours de réglage
if(mode!=reglageHeure&&mode!=reglageMinute&&mode!=reglageJour&&mode!=reglageMois&&mode!=reglageAnnee)
    defilerDate(&date,&previousDefiler,superMillis(&nbRollover,&previousMillis));
```

On met à jour l'heure et la date en fonction du temps qui s'est écoulé depuis le dernier appel de la fonction.

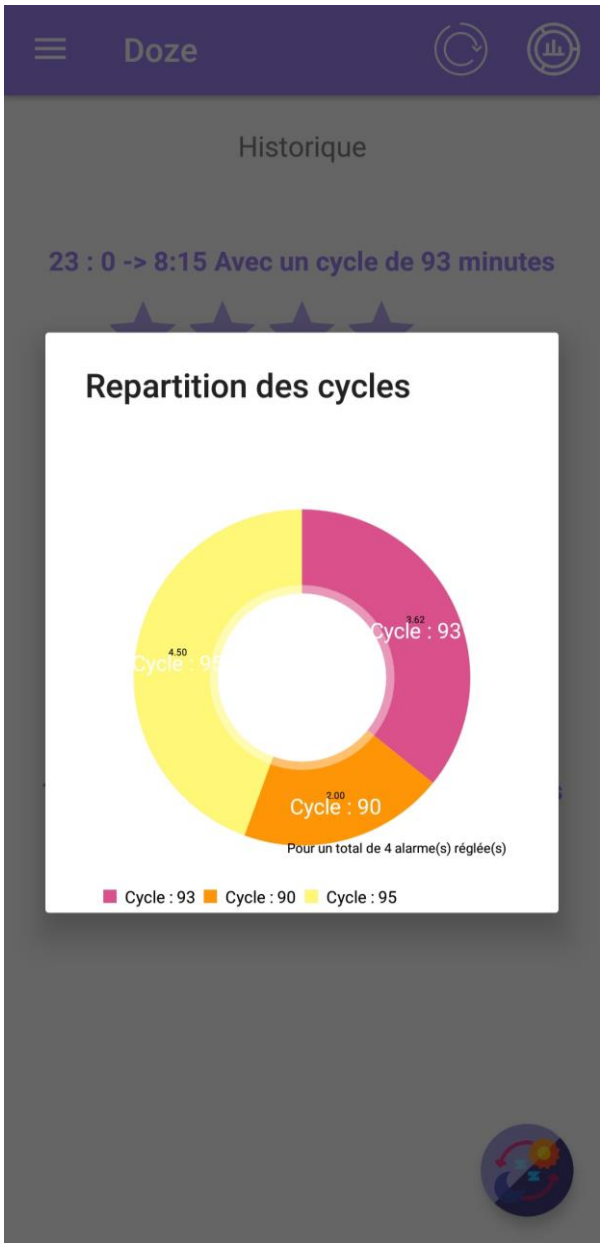
APPLICATION MOBILE

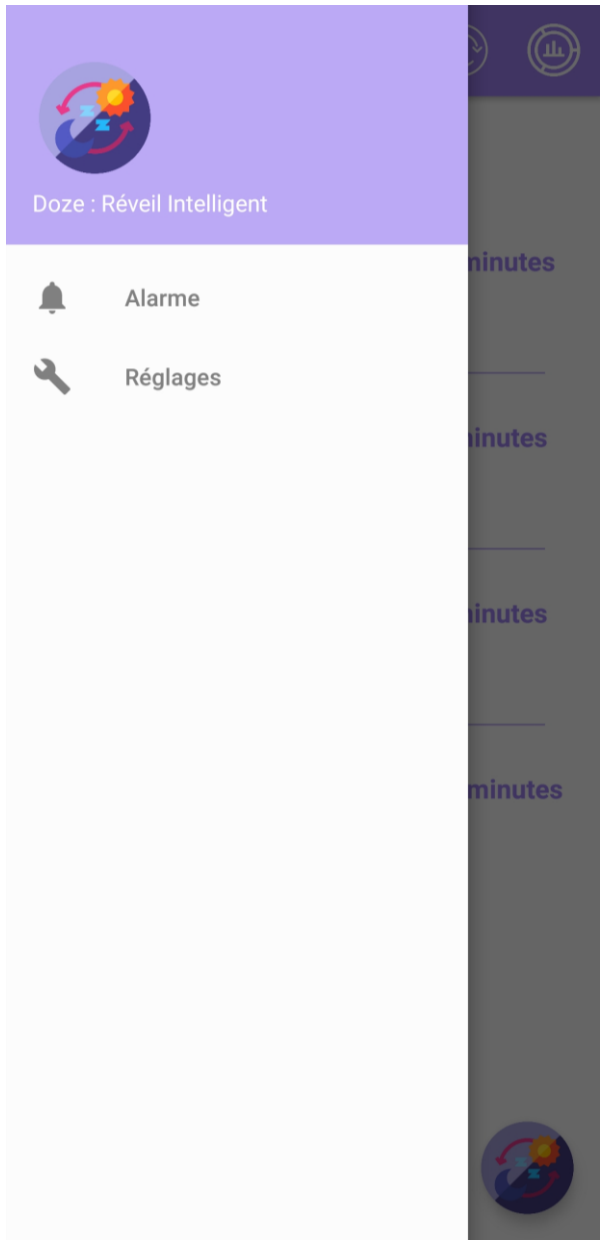
Le code source de l'application est disponible en annexe (voir clé USB).

L'application mobile a pour but de paramétrer le système réel, que ce soit le mode ou l'heure du réveil. Elle permet aussi de visualiser un historique des nuits précédentes et ses cycles favoris. L'application se compose de 3 pages :

- Page d'accueil

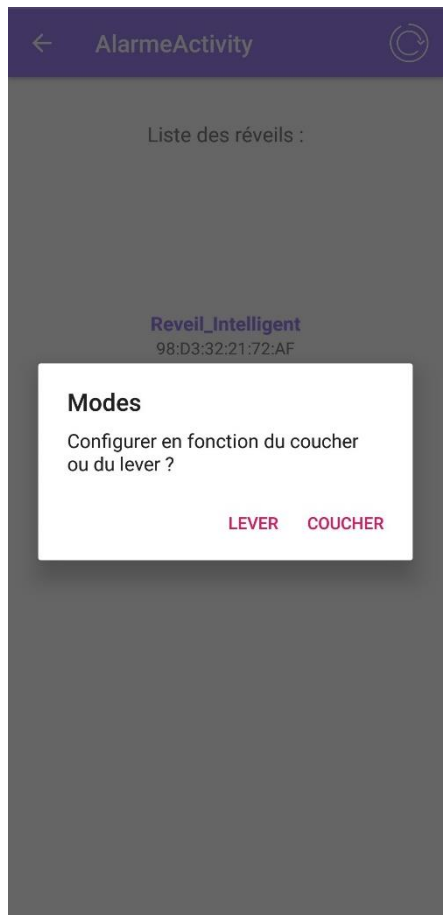
On retrouve sur cette première page l'historique des nuits précédentes on peut modifier la note attribuée à une alarme en cliquant sur la case correspondante. Cette note permet de mieux définir par la suite le cycle de l'utilisateur. Sur la barre d'en-tête, on retrouve respectivement de gauche à droite un bouton, pour naviguer vers le menu, puis pour réinitialiser la page et enfin pour afficher une représentation graphique de nos préférences. Le bouton présent en bas d'écran permet d'activer ou désactiver le mode sombre.

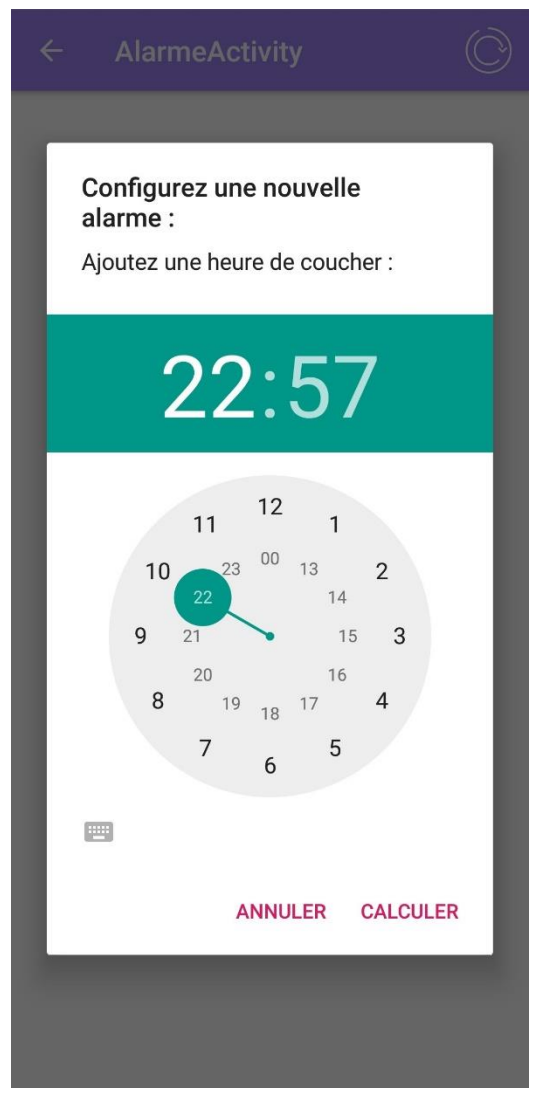

Doze
Historique
23 : 0 -> 8:15 Avec un cycle de 93 minutes
Repartition des cycles
Cycle : 93
3.62
Cycle : 90
2.00
4.50
Pour un total de 4 alarme(s) réglée(s)
Cycle : 93
Cycle : 90
Cycle : 95

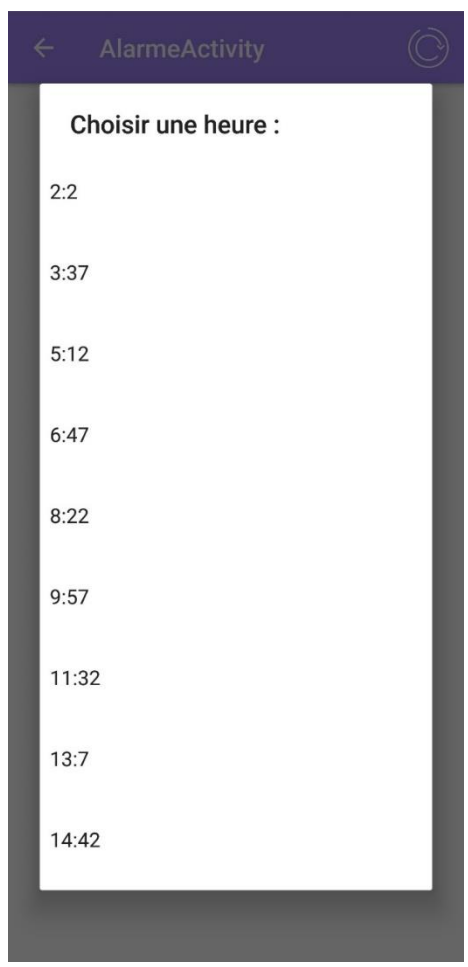

Doze : Réveil Intelligent
Alarme
Réglages

- Page pour paramétrer une alarme

Cette deuxième page permet de paramétrer une nouvelle alarme et de la transférer à l'appareil physique. On choisit d'abord entre deux modes, le premier permet de configurer l'heure du réveil à partir de l'heure du coucher et inversement pour le seconde mode. Le fait de paramétrer une alarme avec le téléphone, permet de laisser ce dernier éteint pour la nuit. On peut dormir sans se soucier des ondes néfastes et d'éventuelles notifications perturbatrices.







- Page de réglages

Cette page permet de régler; la longueur de son cycle, une estimation de notre temps d'endormissement et elle permet d'activer et de désactiver le mode sombre.

CONCLUSION

Nous sommes satisfaits d'avoir pu présenter un produit fonctionnel et travaillé visuellement. Malgré cela, des améliorations sont possibles. En effet nous avons manqué de temps pour ajouter un mode lumière du jour ou bien créer un système d'alimentation annexe ou encore créer une fonction qui permet de téléverser depuis son smartphone sa propre sonnerie. Par exemple, nous avons travaillé sur un module qui réceptionne l'heure légale mais cela n'a pas abouti pour cause le signal reçu était inexploitable.

Ce premier projet technique nous a été bénéfique tant sur le plan technique, car nous avons dû apprendre à utiliser un nouvel IDE (Android Studio) et son langage associé par exemple, que sur le plan humain, car nous avons dû nous organiser pour mener à bien ce projet sur un an. Nous avons pour cela utilisé des outils comme Trello, SharePoint ou Discord pour pouvoir nous distribuer les tâches et suivre notre avancement.

Enfin, si le projet devrait être reconduit, nous proposons d'ajouter les améliorations explicitées plus haut. Un système de commande vocale pourrait aussi être ajouté au réveil.