Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Rapport de projet d'Études et Réalisations

Le simulateur d'aube

Antoine BORDIER
Alexis BOURGUEIL
2e Année - Groupe Q2
Promotion 2011

Enseignants :

M Thierry LEQUEU
M Patrick PAPAZIAN

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Rapport de projet d'Études et Réalisations

Le simulateur d'aube

Antoine BORDIER
Alexis BOURGUEIL
2e Année - Groupe Q2
Promotion 2011

Enseignants :

M Thierry LEQUEU
M Patrick PAPAZIAN

Sommaire

Introduction

Pour notre projet d'Études et Réalisations du semestre 3, nous avons souhaité concevoir un simulateur d'aube. Un simulateur d'aube est un réveil dont la musique a été remplacé par une source lumineuse s'allumant graduellement, censée apporter un réveil plus doux et plus sain.

L'idée vient d'un projet commencé en première année avec M Frédéric Cayrel mais ayant échoué suite à divers problèmes dans la programmation et la réalisation. Ainsi, nous avons souhaité reprendre les bases du projet afin de l'emmener à son terme.

Pour cela, nous avons repris le cahier des charges et l'avons modifié afin de pouvoir résoudre quelques problèmes techniques et apporter des améliorations à celui-ci.

1. Présentation générale du projet

1.1. Cahier des charges

Afin de réaliser ce simulateur d'aube, plusieurs éléments doivent être réalisés :

◆ Réalisation d'un programme en langage C permettant d'effectuer les différentes tâches demandées

◆ Gestion de l'affichage LCD pour l'heure et de la conversion numérique/analogique pour les LED.

Les différentes fonctions à réaliser avec le programme sont les suivantes :

◆ Une entrée permettant de passer du mode normal au mode réglage de l'heure, puis réglage de l'heure de réveil, puis réglage de durée de rampe et enfin revenir au mode normal.

◆ Les différentes sorties pour la mise en marche de LED pour les différents modes et une pour signaler l'activation du réveil.

◆ Les sorties afin de récupérer l'information pour l'éclairage.

◆ Le calcul interne de l'heure de début de cycle afin que à l'heure de réveil la rampe soit au maximum afin d'être réveillé.

◆ Intégration d'une fonction stop afin de couper le cycle une fois réveillé.

La réalisation se compose donc d'un boîtier avec une carte comprenant l'alimentation des différents éléments du boîtier, l'ATMEGA8535, la LED, l'écran LCD et 6 boutons poussoirs.

1.2. Planning prévisionnel et réel

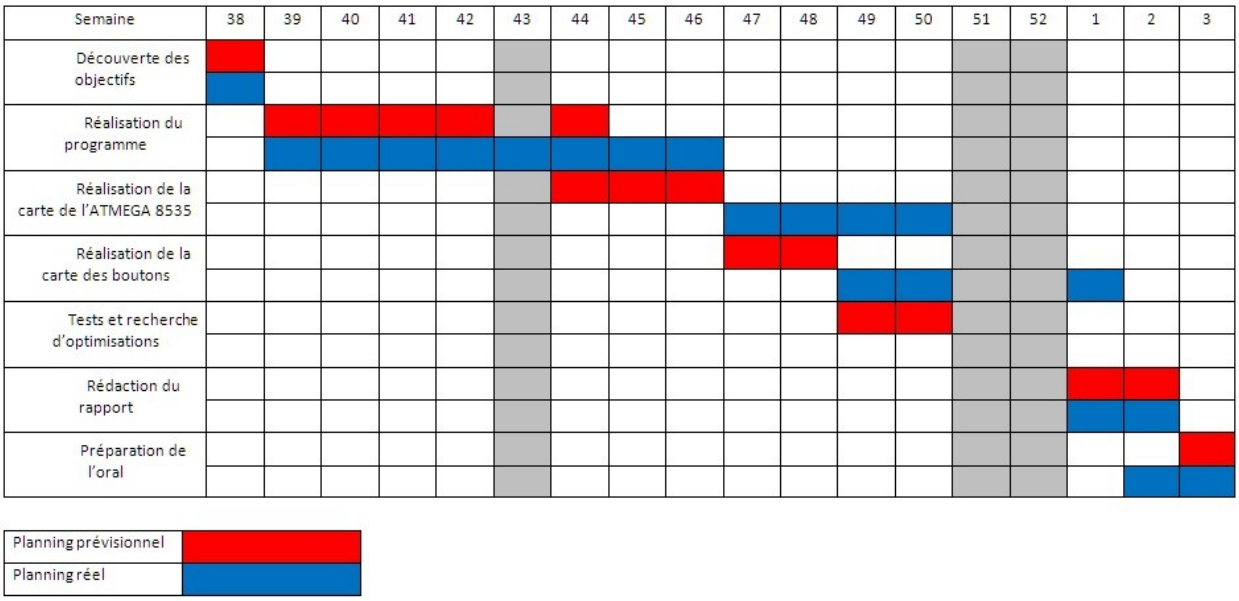


Illustration 1: Planning prévisionnel et réel

1.3. Synoptique du projet

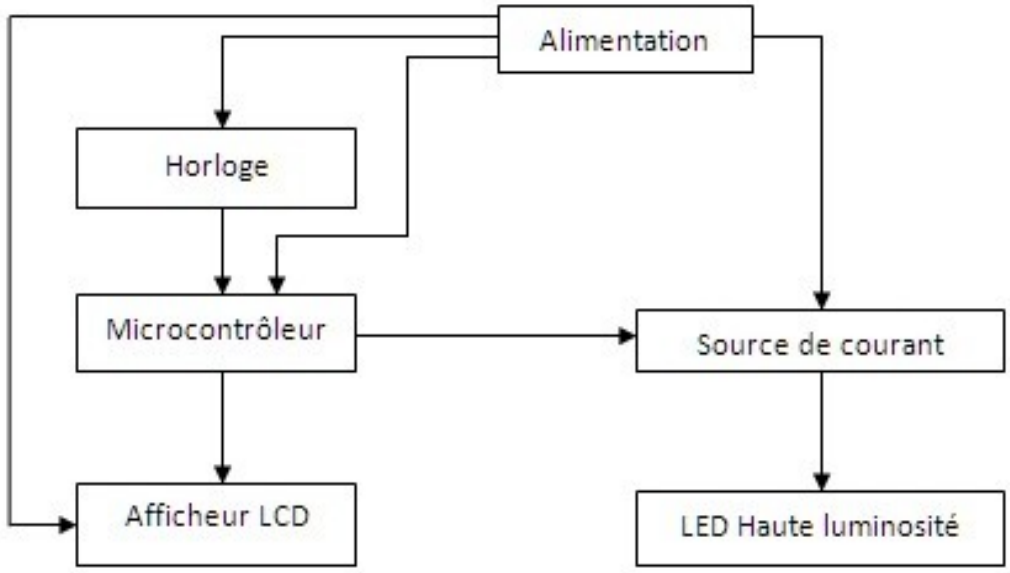


Illustration 2: Synoptique du projet

Mode normal

Ce mode est le mode de fonctionnement le plus courant dans l'utilisation du simulateur d'aube. Il affiche les informations utiles comme l'heure courante, l'état du réveil... Il ne permet aucun réglage.

Mode heure

Ce mode est celui permettant le réglage de l'heure de façon manuelle grâce aux boutons BpH et BpM. L'afficheur affichera sous ce mode l'heure que l'on souhaite enregistrer comme valable. Un appui sur BpH permettra de régler les heures en les incrémentant et le bouton BpM aura un fonctionnement identique pour les minutes.

La nouvelle heure réglée prendra effet dès un nouvel appui sur le bouton de mode (BpMode)

Mode réveil

Ce mode est celui qui permet la configuration de l'heure de réveil. De la même manière que pour le réglage de l'heure, les boutons BpH et BpM permettront de modifier l'heure de réveil voulue. Cette heure sera automatiquement enregistrée à la sortie du mode réveil.

Mode rampe

Ce mode permet le réglage de la durée qu'il faudra afin d'atteindre un éclairage maximum de la LED. Cette durée sera comprise entre 15 et 45 minutes et réglable par pas de 5 minutes. Chaque appui sur BpH ajoutera un pas tandis que BpM en retirera un.

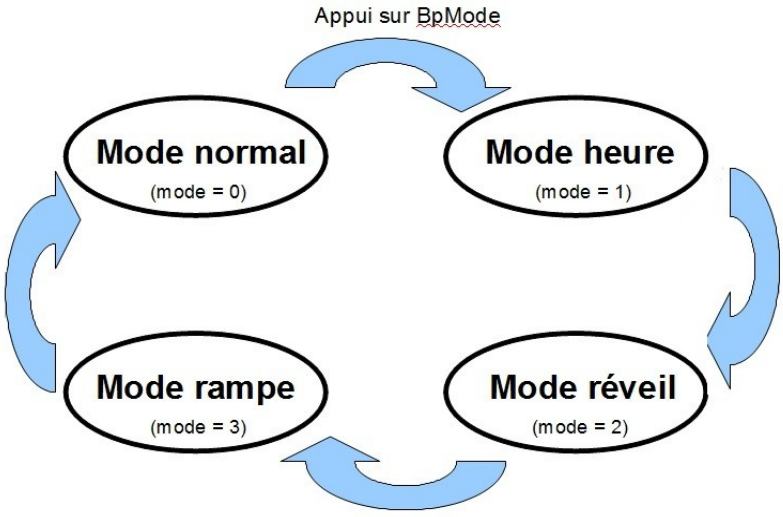


Illustration 3: Synoptique des modes

2. Programmation

Le programme est une des parties fondamentales de ce projet. Il va gérer toutes les fonctions d'horloge, de réveil et automatiser les procédures de calcul d'heure de départ et de rampe. Il va aussi permettre de gérer la MLI[1] sortant du micro-contrôleur.

2.1. Fonctionnement général du réveil

Le fonctionnement du réveil a été pensé de telle sorte que l'utilisateur ait les informations nécessaires permettant une utilisation aisée. L'utilisateur n'agit que sur les 4 boutons prévus pour le réglage du réveil. Au niveau du programme, le simulateur d'aube a été décomposé en 5 grandes parties :

- l'horloge qui permet d'avoir l'heure
- le réveil afin de pouvoir programmer un heure de réveil
- la gestion de la lumière pour le réveil
- la gestion des différentes interruptions et des conditions de fonctionnement
- l'affichage pour une utilisation simple

1 MLI : Modulation en Largeur d'Impulsion. Se dit d'un signal dont le rapport cyclique est variable

2.2. Ordinogramme

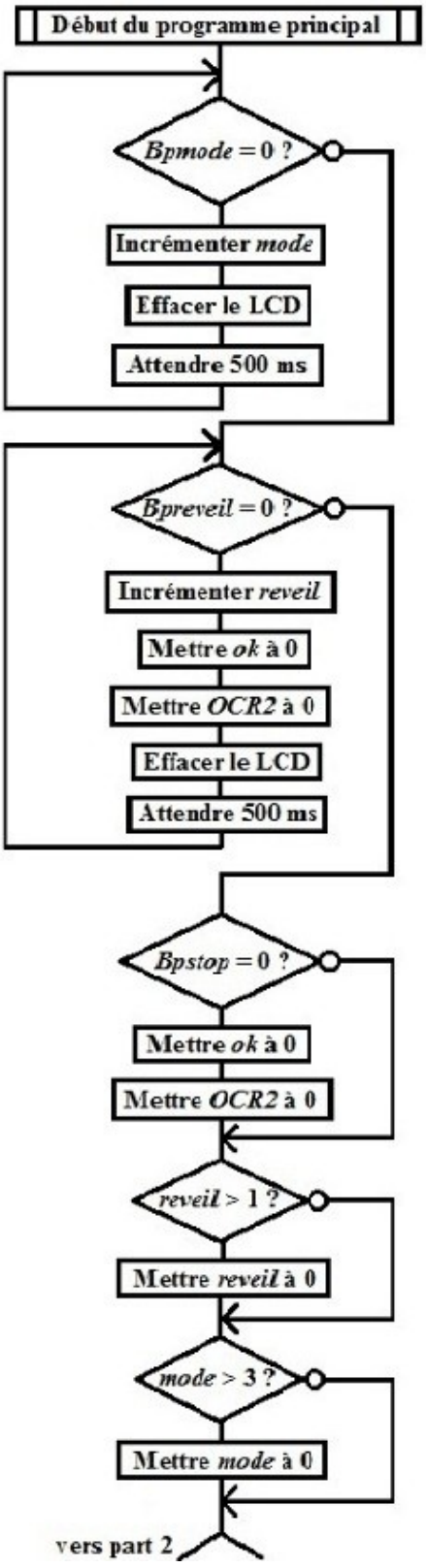


Illustration 4: Ordinogramme (1ère partie)

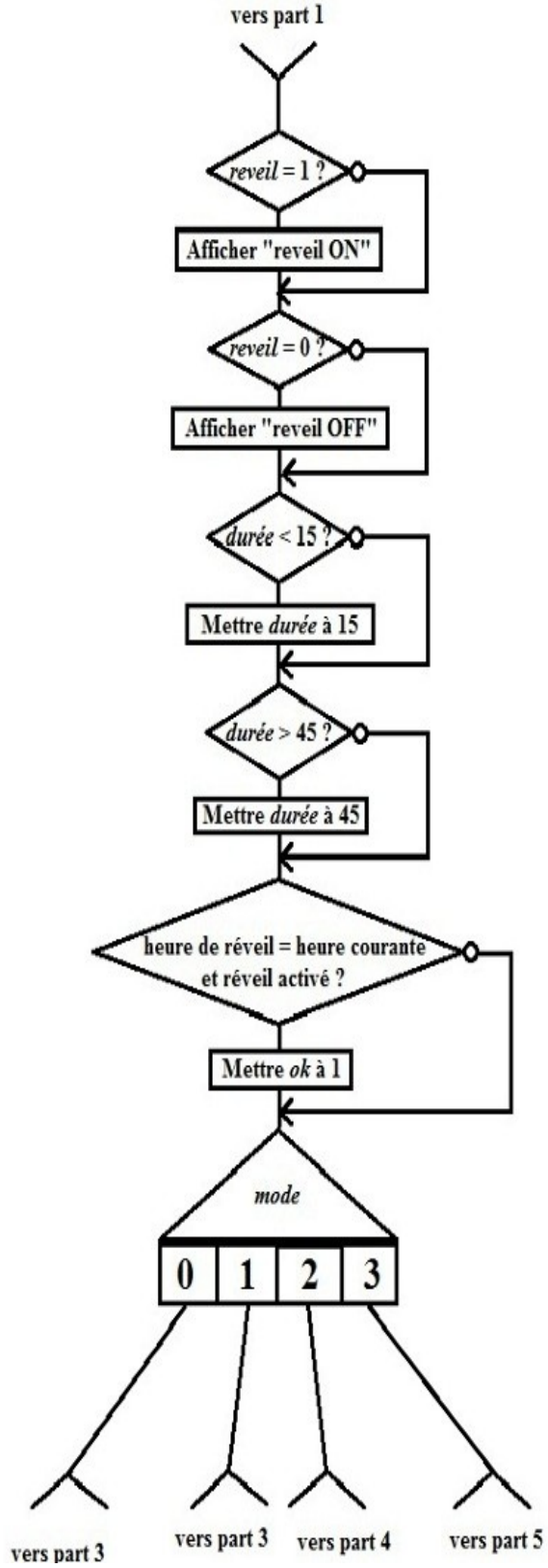
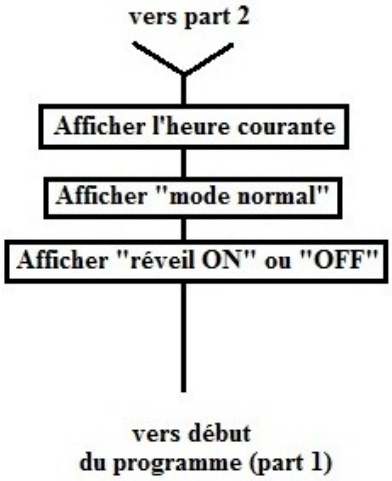


Illustration 5: Ordinogramme (2ème partie)



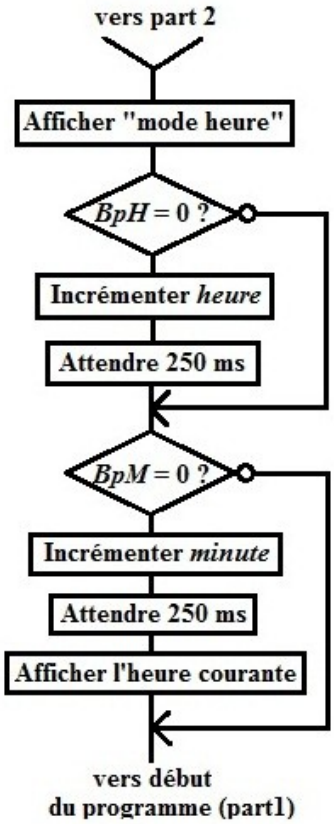


Illustration 6: Ordinogramme (3ème partie)

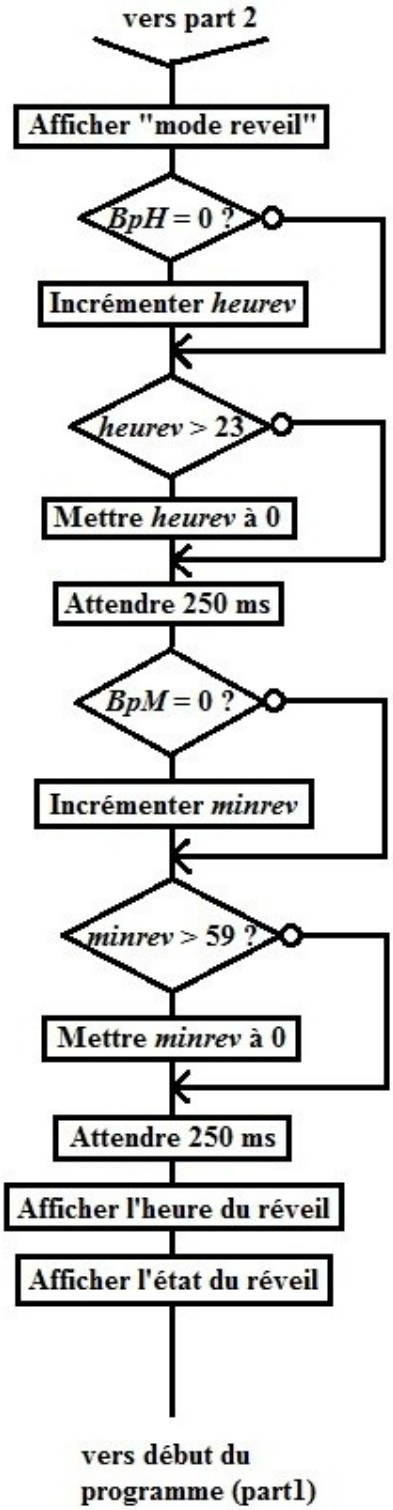


Illustration 7: Ordinogramme (4ème partie)

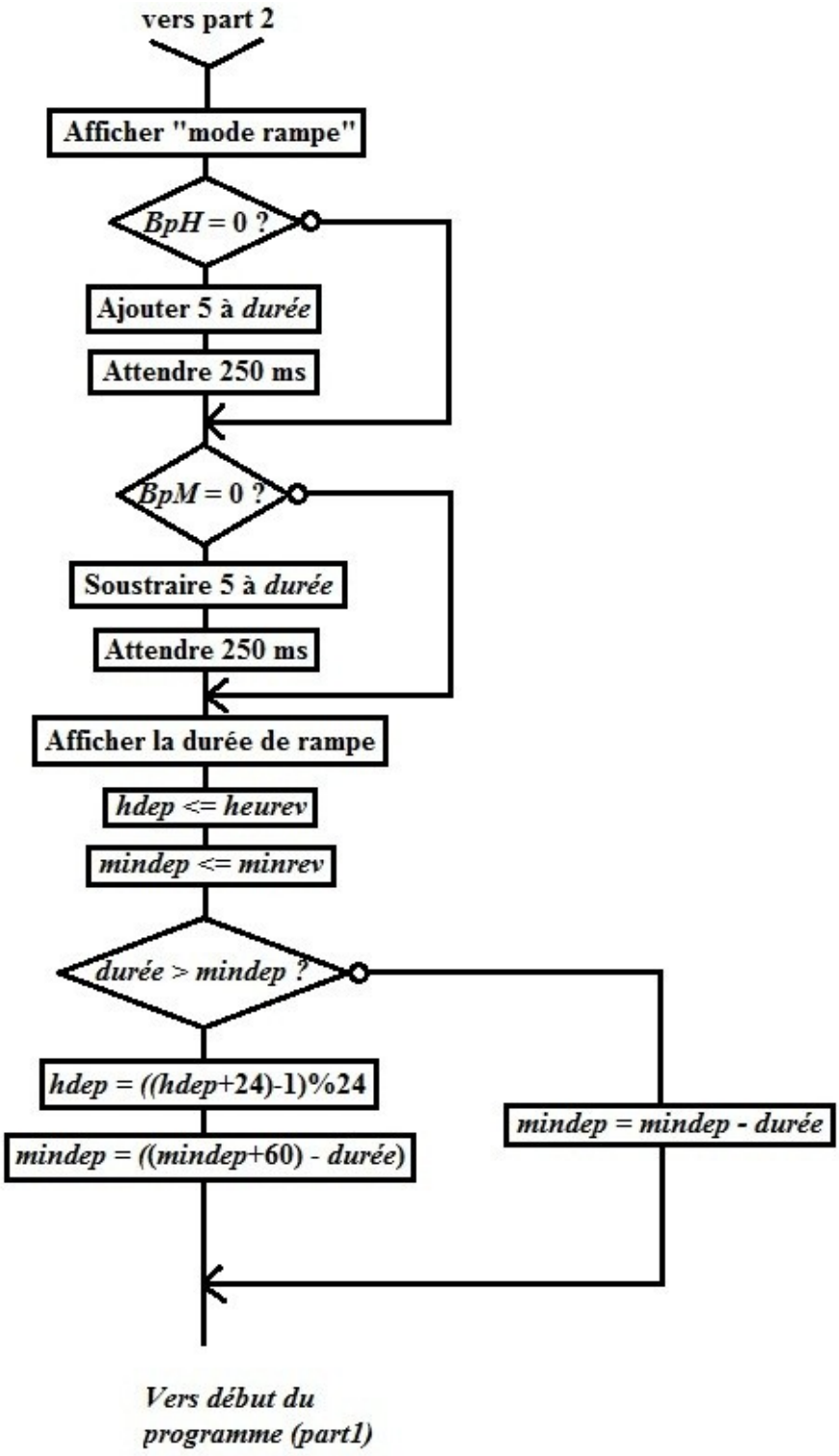


Illustration 8: Ordinogramme (5ème partie)

2.3. Paramétrage de l'ATMEGA8535

Afin de pouvoir créer une horloge et une interface homme machine, il a été nécessaire de paramétrer des entrées/sorties. Il y a donc 7 entrées (les 6 boutons poussoirs et le timer 1) et 2 sorties (le timer 2 et l'écran LCD).

2.3.1. Définition des boutons poussoirs

Dans cette partie, il est avant tout question des boutons poussoirs qui permettent à l'utilisateur d'interagir avec le micro-contrôleur et le programme. Il peut ainsi modifier certaines options du programme de manière simple. Ils sont au nombre de 6 et ont chacun un rôle spécifique suivant le mode du réveil.

Bouton / Mode	BpMode	BpH	BpM	BpReveil	BpStop
Normal	Passe au mode heure	N/A	N/A	Réveil On/Off	Arrête le réveil
Heure	Passe au mode réveil	Incrémente les heures	Incrémente les minutes	Réveil On/Off	Arrête le réveil
Réveil	Passe au mode rampe	Incrémente les heures	Incrémente les minutes	Réveil On/Off	Arrête le réveil
Rampe	Passe au mode heure	Incrémente de 5 min	Décremente de 5 min	Réveil On/Off	Arrête le réveil

Définition des boutons par rapport aux sorties de l'ATMEGA8535 :

```
#define BpMode PIND.2
#define BpH PIND.6
#define BpM PIND.5
#define BpReveil PIND.4
#define BpStop PIND.3
```

2.3.2. Timer 1

Le timer 1 a été utilisé en mode compteur pour créer l'horloge du réveil en prenant le signal depuis un quartz de 16MHz. L'utilisation se fait en incrémentant une variable à chaque fois que le compteur arrive à la valeur déterminée de 15625. Cela ayant pour but de créer une horloge qui change toutes les secondes.

Voici les paramètres du Timer 1 pour effectuer cette opération :

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x0D;            // Compteur bloqué 15625 (3d09 en hexadécimal) pour obtenir une
TCNT1H=0x00;            // interruption toutes les secondes pour incrémenter l'horloge
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3D;
OCR1AL=0x09;
OCR1BH=0x00;
OCR1BL=0x00;
```

Voici l'extrait du programme permettant d'obtenir l'horloge :

```
// Timer 1 output compare A interrupt service routine

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
        // Utilisation du Timer 1 pour créer une horloge qui incrémente
        l'horloge toutes les secondes

        seconde++;
        if (seconde>59)         // Au bout de 60 secondes, les secondes sont
        {                               remises à zéro et les minutes sont incrémentés
                seconde=0;
                minute++;
        }

        if(minute>59)                   // Au bout de 60 minutes, les minutes sont
        {                               remises à zéro et l'heure est incrémenté
                minute=0;
                heure++;
        }

        if(heure>23)// Au bout de 24h, l'heure est remise à zéro
        {
                heure=0;
        }
}
```

2.3.3. Timer 2

Le timer 2 est, lui, utilisé pour obtenir notre sortie en MLI afin de piloter notre LED pour avoir notre allumage progressif au moment du réveil. Pour cela, on utilise la fonction compteur du timer que l'on cadence au rythme des minutes de notre horloge, dans le but de respecter le temps demandé par l'utilisateur lors de ses réglages.

Voici le rappel des paramètres utilisés pour le timer 2 :

```
// Timer/Counter 2 initialization
```

```
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x61;   // Timer 2 configuré en MLI pour alimenter la LED
TCNT2=0x00;
OCR2=0x00;
```

Ainsi que le code pour incrémentant le compteur permettant d'obtenir la MLI :

```
if(ok==1)
{
        OCR2=OCR2+(255/duree); // Incrémentation du Timer 2 pour alimenter la
        if(OCR2>255)                                    LED
        {
                OCR2=255;
        }
}
```

2.3.4. Écran LCD

L'écran LCD est là pour amener un confort visuel à l'utilisateur en lui apportant un maximum d'informations nécessaires pour l'utilisation, selon le mode dans lequel il est. Il permet l'affichage d'un nombre d'informations importantes dans notre cas, puisqu'il s'agit d'un écran LCD 4 lignes x 16colonnes.

Il est initialisé par le code suivant :

```
// Alphanumeric LCD Module functions
#asm
        .equ __lcd_port=0x15 ;PORTC
#endasm

// LCD module initialization
lcd_init(16);
```

Voici les différents affichage selon les modes :

- mode 0 (Affichage de l'heure) :

```
sprintf(heure_txt,"%02d : %02d : %02d",heure,minute,seconde);
lcd_gotoxy(0,2);
lcd_puts(heure_txt);

sprintf(mode_txt,"mode normal");    // Mode affichage de l'heure
lcd_gotoxy(0,0);
lcd_puts(mode_txt);

lcd_gotoxy(6,3);
```

```
lcd_puts(reveil_txt);
```

- mode 1 (Réglage de l'heure) :

```
sprintf(mode_txt,"mode heure ");
lcd_gotoxy(0,0);
lcd_puts(mode_txt);

if(BpH == 0)
{       // Mode Réglage de l'heure avec affichage de l'heure réglée
        heure++;    // Un appui sur BpH incrémente l'heure de 1
        delay_ms(250);
}

if(BpM == 0)
{       // Un appui sur BpM incrémente les minutes de 1
        minute++;
        delay_ms(250);
}
sprintf(heure_txt,"%02d : %02d : %02d",heure,minute,seconde);
lcd_gotoxy(0,2);
lcd_puts(heure_txt);
```

- mode 2 (Réglage de l'heure de réveil) :

```
sprintf(mode_txt,"mode reveil ");
lcd_gotoxy(0,0);
lcd_puts(mode_txt);

if(BpH == 0)
{       // Mode de réglage du réveil avec affichage
        heurev++;  // Un appui sur BpH incrémente l'heure de réveil de 1
        if(heurev>23)
        {
                heurev=0;
        }
        delay_ms(250);
}

if(BpM == 0)
{       // Un appui sur BpM incrémente les minutes de réveil de 1
        minrev++;
        if(minrev>59)
        {
                minrev=0;
        }
        delay_ms(250);
}

sprintf(hrev_txt,"%02d : %02d ",heurev,minrev);
lcd_gotoxy(0,2);
lcd_puts(hrev_txt);
```

```
lcd_gotoxy(6,3);
lcd_puts(reveil_txt);
```

- mode 3 (Réglage de durée de la rampe) :

```
sprintf(mode_txt,"mode rampe");     // Mode de réglage de durée de rampe avec
lcd_gotoxy(0,0);                                affichage de la durée
lcd_puts(mode_txt);

if(BpH == 0)
{
        duree=duree+5;          // Un appui sur BpH augmente la durée de rampe de
        delay_ms(250);          5 min
}

if(BpM == 0)
{
        duree=duree-5;          // Un appui sur BpM réduit la durée de rampe  de
        delay_ms(250);          5 min
}

sprintf(rampe_txt,"%02d min",duree);
lcd_gotoxy(0,2);
lcd_puts(rampe_txt);
```

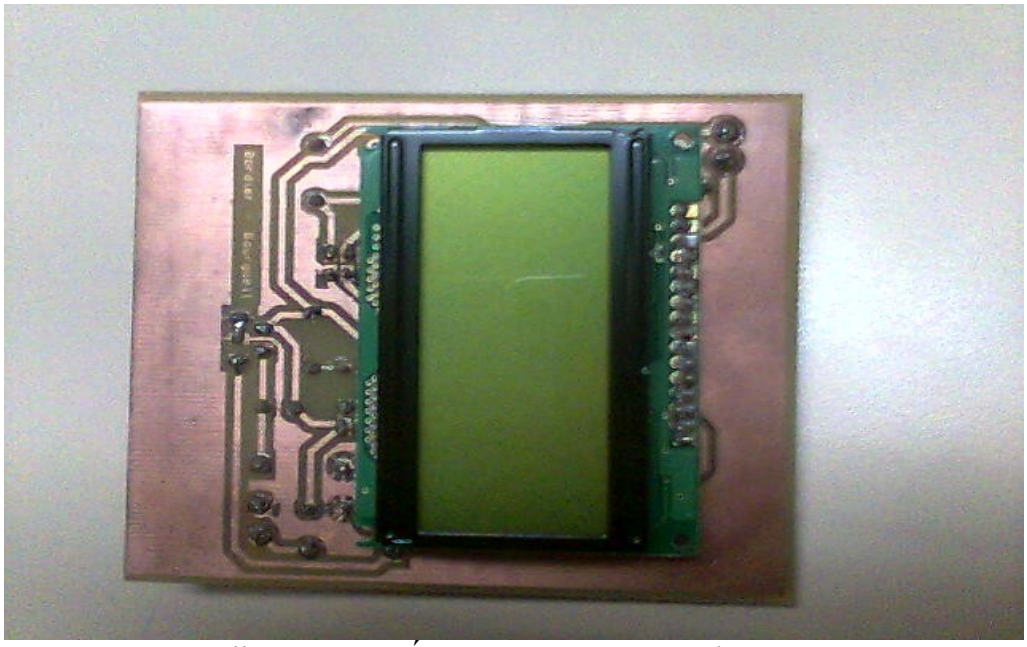


Illustration 9: Écran LCD présent sur la carte

3. Réalisation des cartes

Lors de ce projet, il nous est paru évident de faire 2 cartes. La raison est que les boutons sont à l'extérieur du boîtier pour que l'utilisateur puisse agir dessus, alors que le micro-contrôleur, lui, est au fond du boîtier, afin d'éviter qu'il soit détruit en cas de choc.

3.1. Schéma structurel

3.1.1. Carte ATMEGA8535

Le schéma structurel de la carte ATMEGA8535 suivant a été séparé en différents blocs différencies par des couleurs pour un repérage plus simple des fonctions de chaque bloc avec les composants associés.

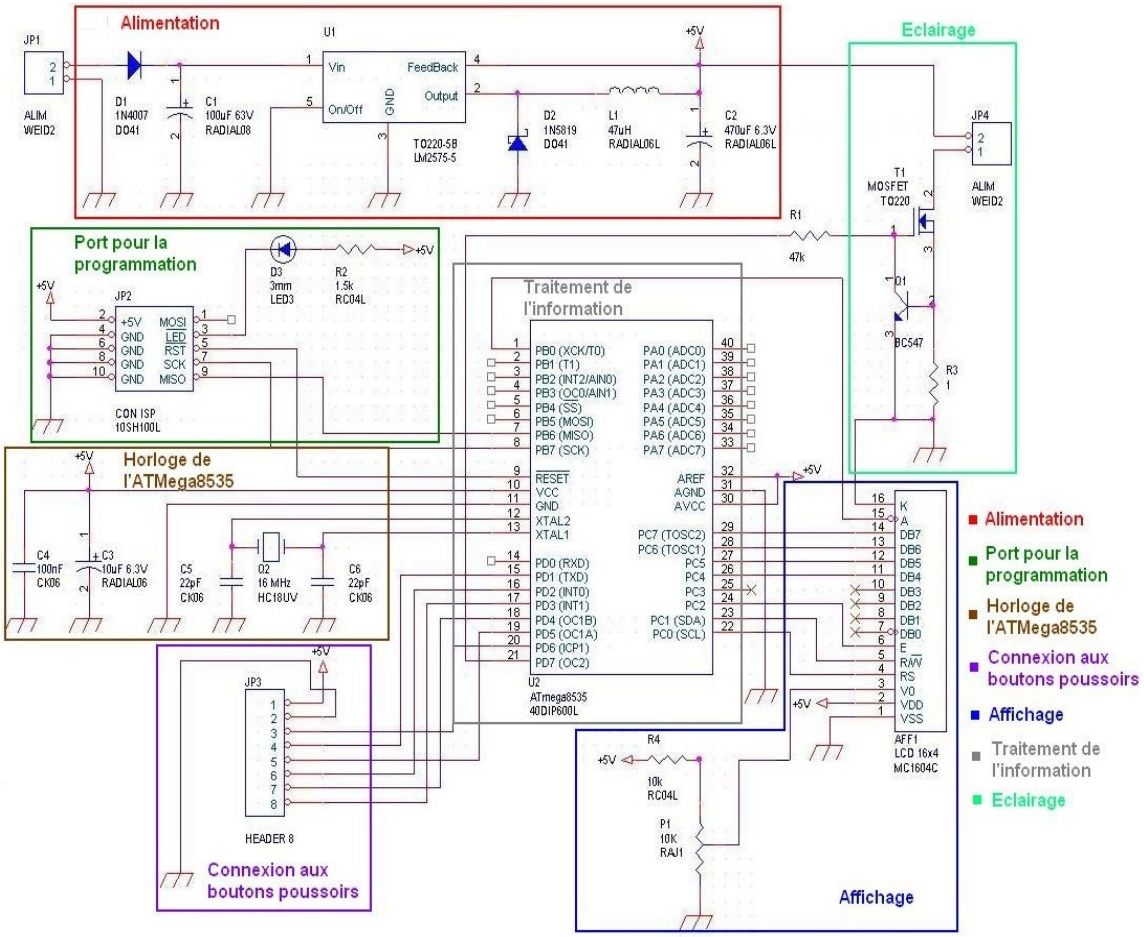


Illustration 10: Schéma structurel de la carte ATMEGA8535

Fonction de la carte

FPA : Fonction alimentation

En entrée de cette fonction nous avons le connecteur JP1 qui apporte l'énergie nécessaire au fonctionnement de la carte. Cependant la tension arrivant sur ce JP1 étant une tension de 12V (imposée par les transformateurs à disposition) il faut la réduire à 5V, tension de fonctionnement de l'ATMega8535 et de la LED de puissance. Cet abaissement de tension sera effectué par un hacheur abaisseur plutôt qu'un régulateur linéaire afin de minimiser les pertes d'énergie (rendement d'environ 90% contre 65%).

Notre choix s'est en premier porté sur le circuit intégré LM2574 qui permet de délivrer une tension de 5V et un courant maximum de sortie de 0,5A. Finalement afin de répondre à la demande de puissance de la LED d'éclairage et de l'ATMega nous avons pris le LM2575 qui délivre aussi une tension de 5V mais permet de fournir jusqu'à 1 A en courant de sortie.

Les composants disposés autour du LM2575 sont ceux nécessaires pour un fonctionnement correct de l'alimentation. Ces composants sont donnés dans la documentation constructeur du composant.

FP1 : Éclairage

L'éclairage sera effectué à l'aide d'une LED de puissance déjà achetée par l'IUT. Cette LED nécessitant un courant de 700mA au maximum et une tension de 4.5V, on comprend maintenant le choix du LM2575. Cette LED sera reliée directement au connecteur JP4.

Les deux transistors et la résistance de cette fonction FP1 constituent un montage permettant la régulation du courant dans la LED. Ce montage est issu de la revue "Elektor". Voir en annexe 2 l'article consacré à celui-ci.

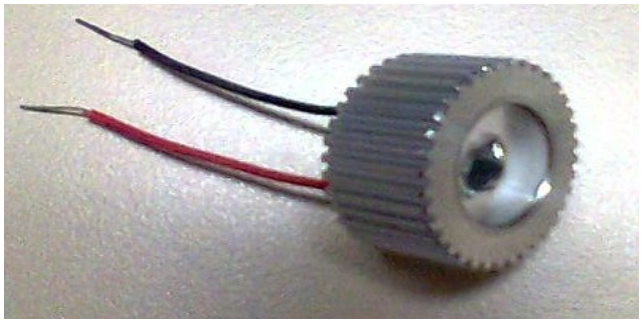


Illustration 11: LED de puissance utilisé pour l'éclairage

FP2 : Affichage

Cette fonction est essentielle non pas au fonctionnement du montage mais au dialogue utilisateur/système. Cette fonction sera effectuée à l'aide d'un afficheur LCD 16 caractères * 4 lignes.

On remarque en plus des pattes nécessaires au fonctionnement du LCD que les pattes 15, 16 sont reliées à l'ATMEGA8535 et à la masse. Le câblage de ces pattes permettra de rétro-éclairer l'afficheur si besoin par programmation.

Le montage à pont diviseur appliqué sur la patte 3 permet le réglage du contraste de l'afficheur par le potentiomètre P1.

FP3 : traitement de l'information

Cette fonction est assurée par un microcontrôleur ATMEGA8535. Ce composant est le centre de tout le système, c'est lui qui gère l'intégralité des interactions avec l'extérieur. C'est dans ce composant qu'est sauvegardé le programme entré par le concepteur qui décrit les actions effectuées par le système.

Le composant devra donc assuré les fonctions suivantes :

- - gestion des entrées (boutons poussoirs)
- - gestion de l'affichage sur le LCD
- - gestion de l'intensité lumineuse de la LED, grâce à sa sortie MLI.
- - gestion du programme permettant de définir l'heure courante, l'heure de réveil etc...

FP4 : Connexion aux boutons poussoirs

Cette partie n'est pas vraiment une fonction, c'est en fait un connecteur qui est relié aux boutons poussoirs qui sont déportés sur une autre carte.

FP5 : Horloge de l'ATMEGA8535

Ces composants sont imposés dans la documentation constructeur de l'ATmega8535. Ils permettent d'obtenir une fréquence stable afin de cadencer le travail de l'ATMega8535.

FP6 : Port de programmation

Ce port permet de rentrer dans le micro-contrôleur le programme voulu et donc de définir le comportement du système.

La programmation s'effectue par le biais du logiciel Avr Studio disponible sur les ordinateurs de l'université.

3.1.2. Carte bouton

Le schéma structurel suivant est celui de la carte des boutons.

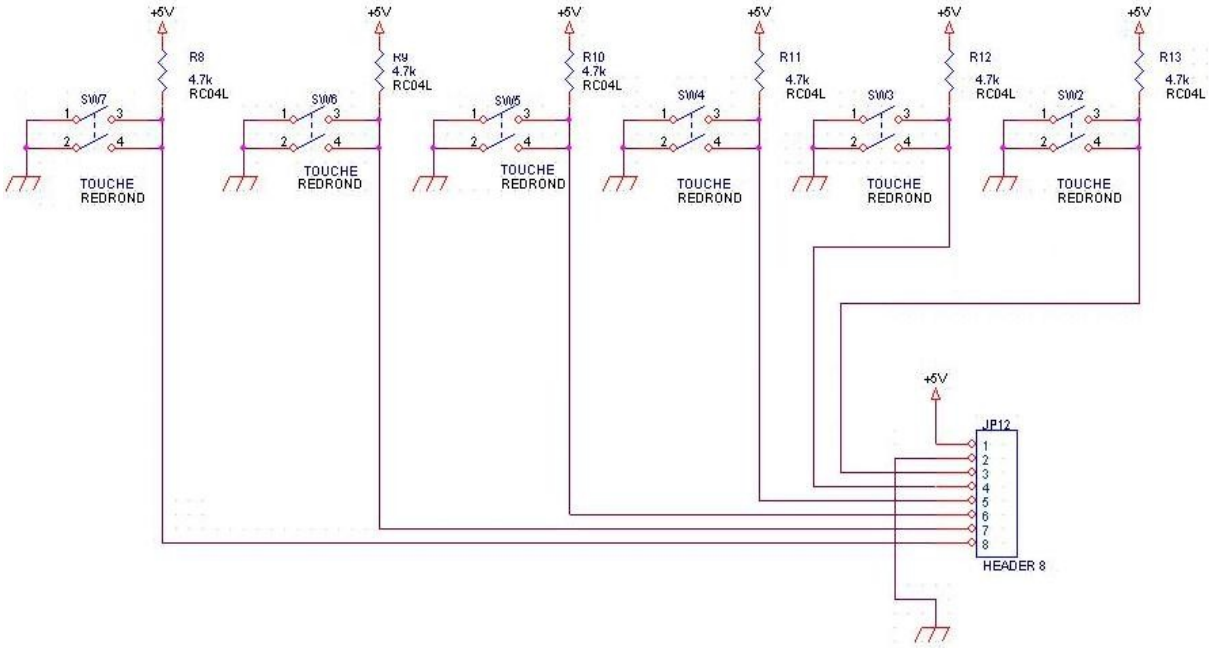


Illustration 12: Schéma structurel de la carte de boutons

3.2. Typon

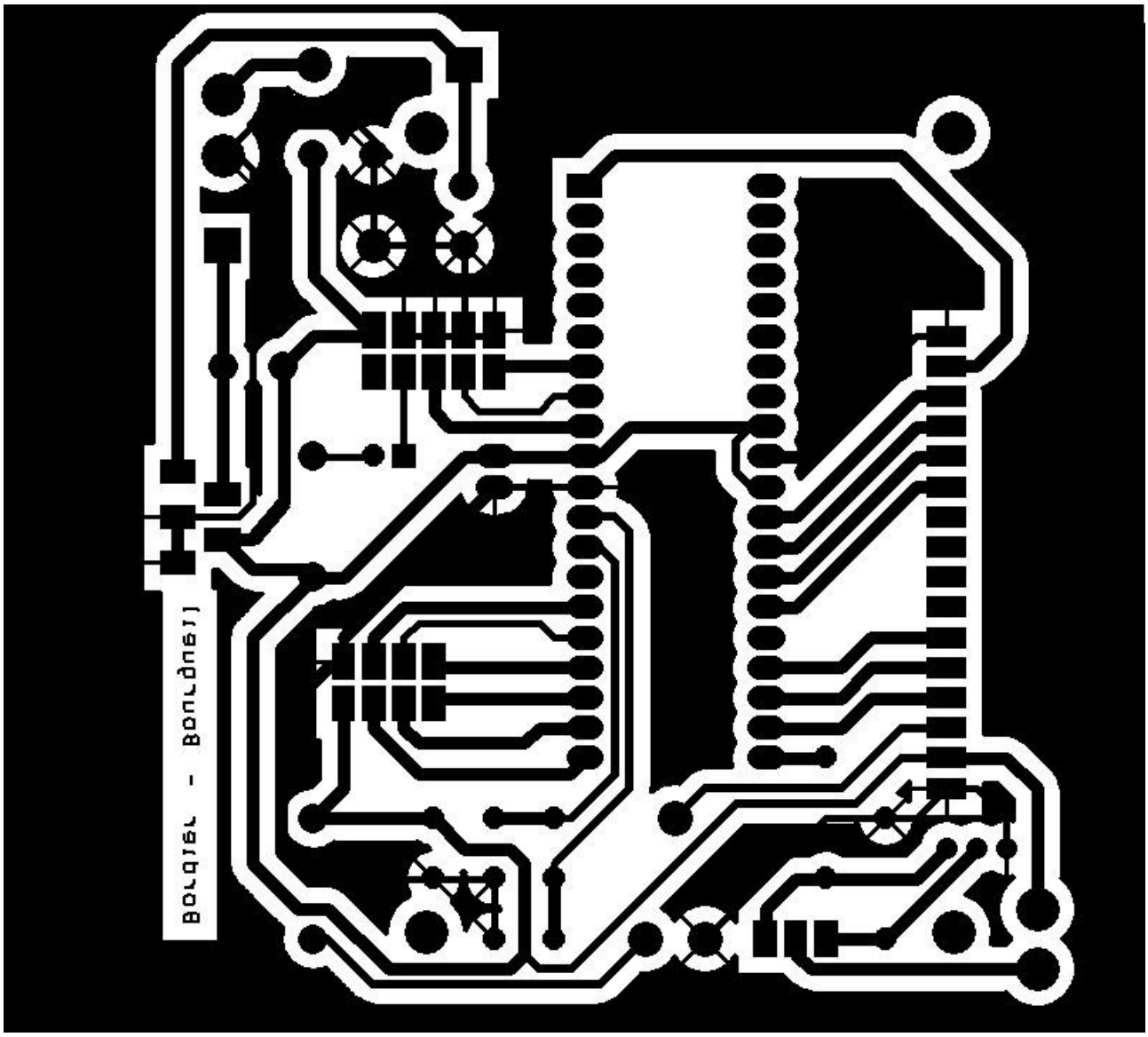


Illustration 13: Typon de la carte ATMega8535

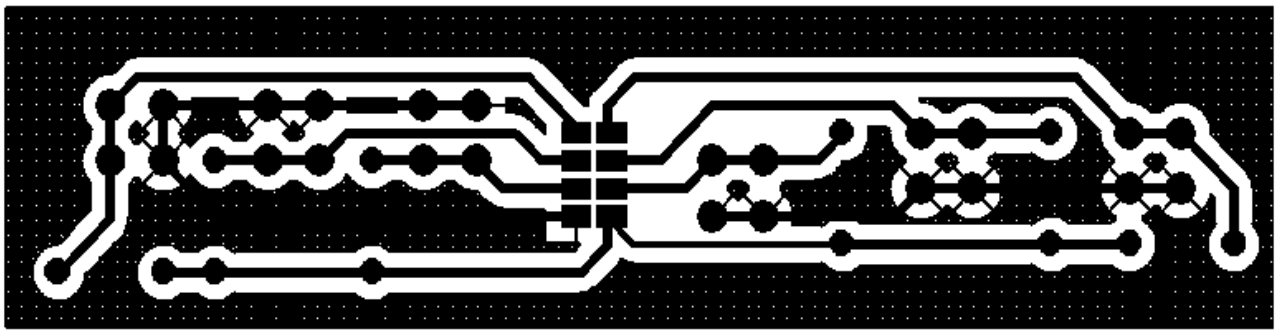
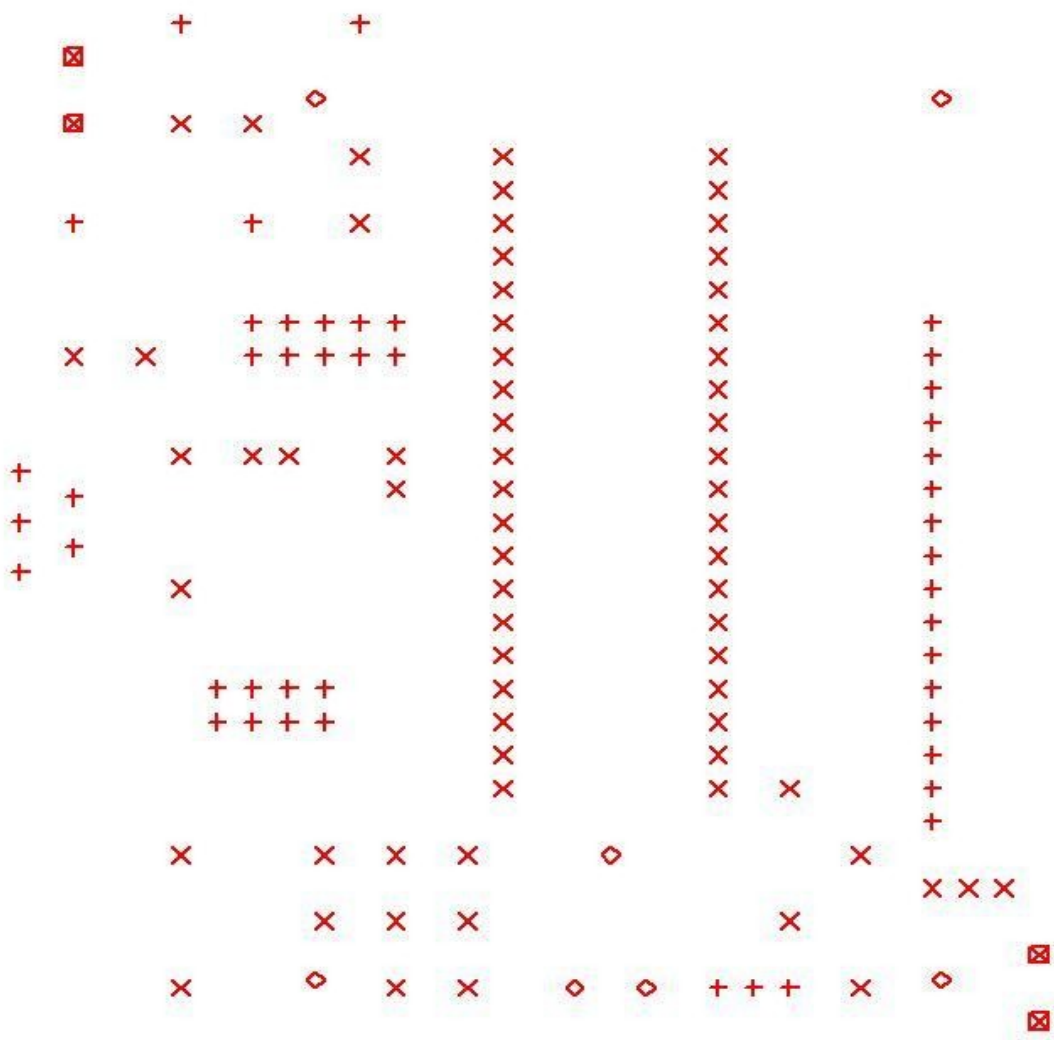


Illustration 14: Typon de la carte des boutons poussoirs

3.3. Perçage



DRILL CHART			
SYM	DIAM	TOL	QTY
×	0.787 mm		69
+	0.991 mm		46
⊠	1.000 mm		2
◇	1.194 mm		7
⊠	1.499 mm		2
TOTAL			126

Illustration 15: Plan de perçage de la carte ATMega8535

3.4. Implantation

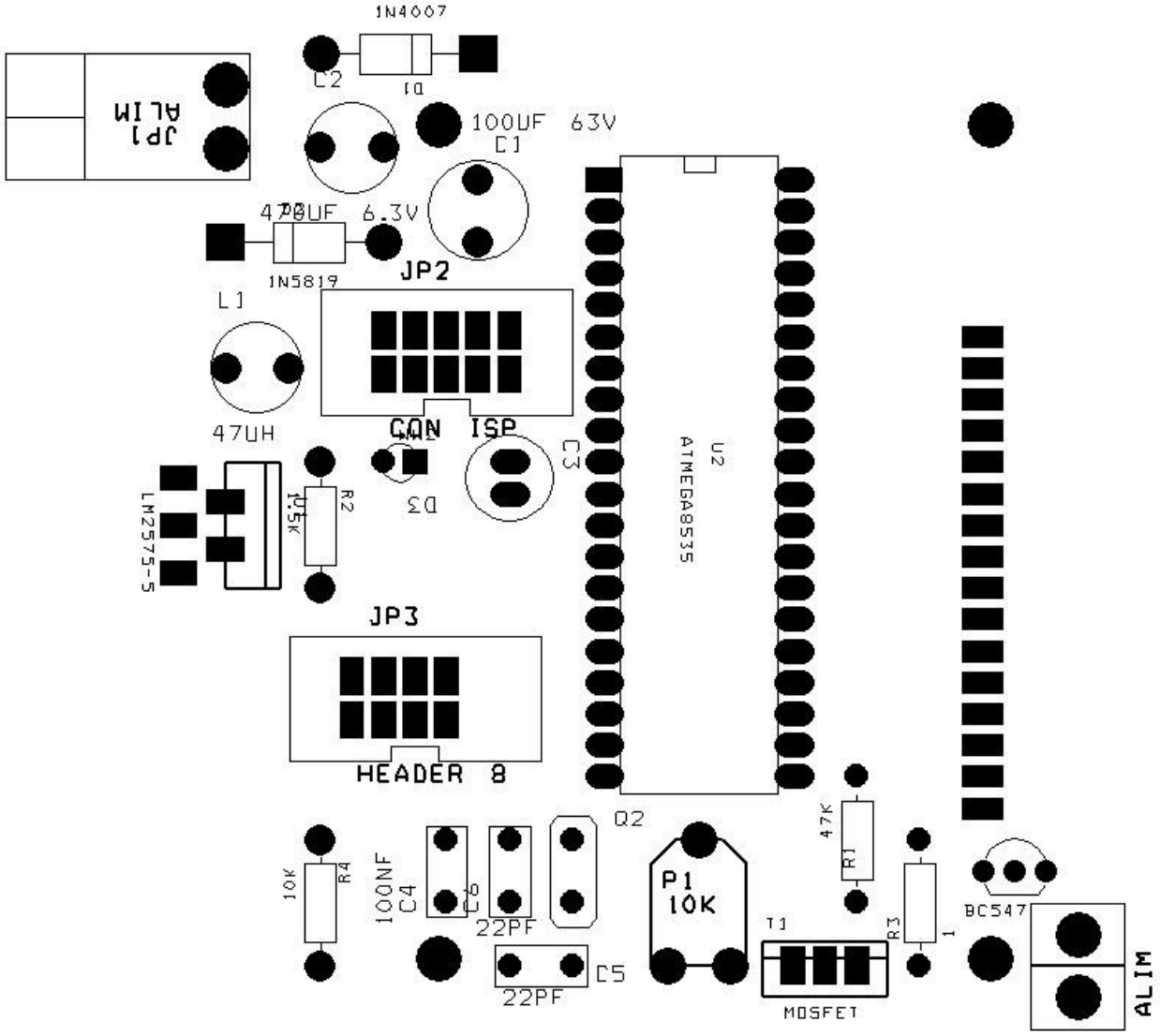


Illustration 16: Schéma d'implantation de la carte ATMega8535

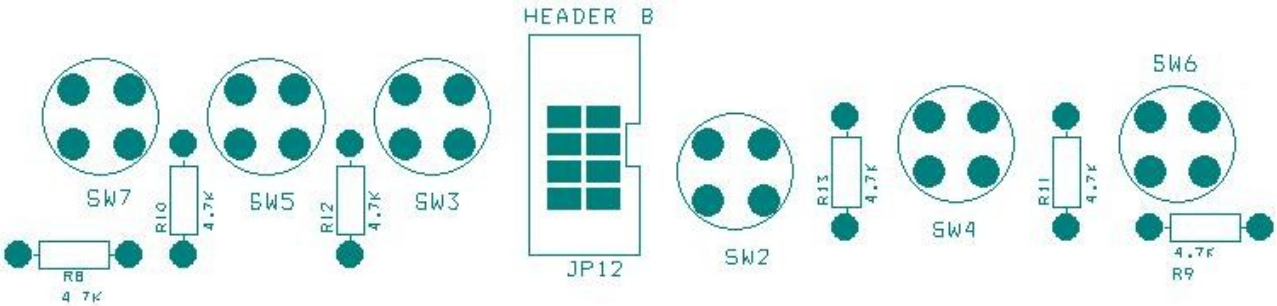


Illustration 17: Schéma d'implantation de la carte des boutons poussoirs

4. Conception et tests

Cette dernière partie est là pour traiter de la réalisation concrète du projet, ainsi qu'exposer les divers problèmes rencontrés lors de la réalisation et les tests effectués afin de vérifier le bon fonctionnement de chacun des éléments. Il sera aussi question des possibilités d'amélioration du projet par l'ajout de nouvelles fonctions dans le programme ou bien de nouveaux éléments dans le boîtier.

4.1. Étapes de conception

Dans cette partie il sera énuméré les étapes de conception qui ont mené à l'achèvement de la carte.

Le cahier des charges n'imposant pas de contraintes particulières sur la composition de la carte nous sommes partis dans l'idée d'utiliser un ATMega8535. Nous avons pris comme base la carte de test d'ATMega8535 réalisé par M Thierry LEQUEU à laquelle nous avons retirée les éléments qui n'étaient pas nécessaire à notre projet.

Une fois ces retraits effectués nous avons ajouté les éléments requis par notre projet. Ces éléments sont :

- La mise en place de la possibilité de rétro-éclairé l'écran.
- Le rajout de bouton poussoirs et un connecteur pour les relier de manière déporté.
- Le montage permettant l'alimentation en courant de la diode de puissance.
- Le remplacement du LM2574 par le LM2575 afin d'augmenter le courant maximal en sortie.

Toutes ces modifications ont été saisies sous le logiciel Orcad CaptureCIS qui permet de créer des schémas électriques. Une fois le schéma structurel achevé tout les paramètres de composants et leurs empreintes ont été validés afin de pouvoir créer la Netlist du circuit.

Cette Netlist qui vient d'être créée est ensuite ouverte sous le logiciel Orcad Layout. Layout permet de réaliser les éléments amenant à la création physique de la carte. Ces éléments sont le typon, le plan de perçage et le plan d'implantation. L'impression de ces trois documents permet d'insoler, percer et souder les composants sur la carte et donc de compléter la réalisation de la carte.

Afin de vraiment finaliser le projet il ne manquera plus qu'à insérer le tout dans une boîte une fois que les tests auront été effectués et la carte validée.

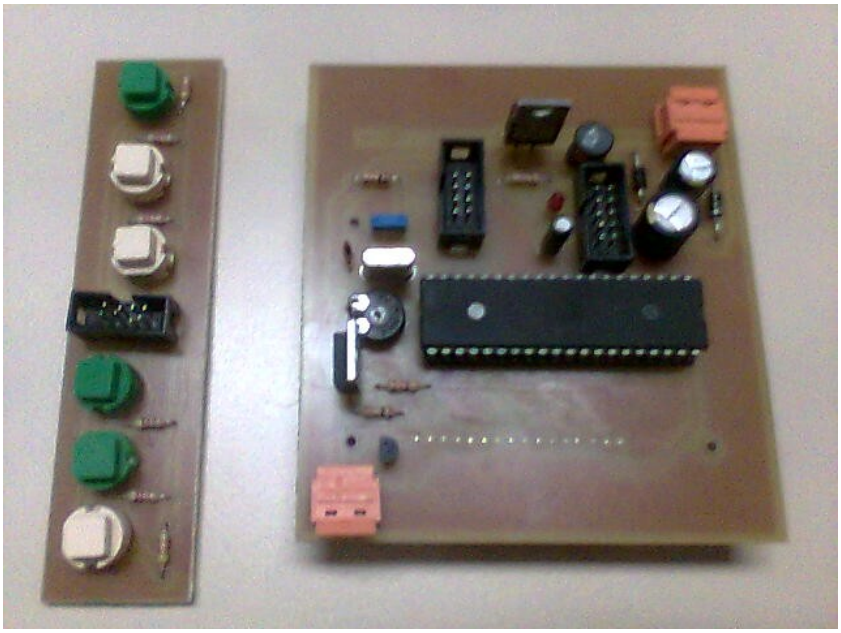


Illustration 18: Photo de la carte réalisée

4.2. Problèmes rencontrés

Lors de la réalisation de ce projet, aucun problème majeur n'a été rencontré. Cependant voici quelques petits aléas qui sont tout de même intervenus :

- ◆ - échec de l'insolation de la première carte d'ATmega8535
- ◆ - réalisation d'un strap entre deux pattes de l'ATmega8535 puisque le routage n'a pu être complété de par sa complexité.
- ◆ - les empreintes utilisées sous Layout pour les boutons poussoirs n'étant pas tout à fait les bonnes il a fallu vérifier la correspondance des 4 pattes de ceux ci afin d'éviter des dysfonctionnement.
- ◆ - il a fallu sur la carte des boutons poussoirs inverser la face de la carte où était soudé le connecteur puisque celui-ci se retrouvait à l'extérieur du boîtier tout comme les boutons.

4.3. Tests

Une fois la carte réalisée selon le cheminement décrit précédemment il est nécessaire de s'assurer de son bon fonctionnement à l'aide d'une série de tests.

Dans un premier temps il est de rigueur de commencer des tests qui sont commun à tout projet. Ces tests sont la vérification de l'absence de coupure dans les pistes de la carte ou au contraire de la présence de cour-circuit.

Dans notre cas la deuxième série de test porte sur l'alimentation, celle-ci s'effectue par mesure de précaution sans le micro-contrôleur afin de lui éviter toute destruction en cas de problèmes. Il a fallu vérifier que le hacheur fournissait bien une tension de 5V avec un courant suffisant et que cette tension était bien acheminée aux endroits de la carte où elle est nécessaire.

Les derniers test s'effectuent en ajoutant l'ATMega8535. L'ajout de ce composant permet de vérifier le fonctionnement général du montage. Ce test du fonctionnement global n'est valable qu'à la condition que le programme n'est pas de dysfonctionnement. En effet si le programme est incorrect il serait possible de croire que le problème vient de la carte alors que c'est seulement logiciel.

Afin d'éviter cette confusion entre le logiciel et le matériel, le programme que nous avons créé, a été auparavant testé sur une autre carte.

Si lors de l'ajout de l'ATMEga8535 et d'un programme fonctionnel il y a encore des problèmes alors il faudra tester les origines possibles de ceux ci. Par exemple si la LED ne s'allume pas de manière croissante il peut être utile de vérifier le signal de sortie de la MLI de micro-contrôleur.

Conclusion

Le projet doit être testé dans son ensemble, néanmoins chaque partie doit l'être de manière séparée. Suite aux tests des différentes parties, aucune anomalie n 'est à signaler. Le but premier de ce projet étant de pouvoir réveiller une personne à l'aide d'une lumière progressive, afin d'imiter un réveil naturel, il reste à savoir si cela suffit ou si l'ajout d'un signal sonore pourra être un élément à étudier dans le futur. De plus, le projet peut encore être amélioré par l'ajout d'une LED supplémentaire via une option que peut choisir l'utilisateur ou simulé un réel lever de soleil à l'aide de LED de couleur.

Résumé

Le but de ce projet était de créer un simulateur d'aube, réveil ayant une lumière progressive au lieu de musique, avec une durée de réveil variable. Pour cela, nous avons divisé le projet en 3 grandes tâches : le programme, la carte comportant le micro-contrôleur, l'alimentation, la source de courant pour la LED et l'écran LCD et une carte de boutons pour créer une interface avec l'utilisateur.

Le programme permet de gérer toutes les opérations de calculs, d'affichage et les entrées/sorties. Nous avons utilisé deux fonctions timer afin de créer l'horloge de base et l'allumage progressif de la LED, 7 entrées/sorties comprenant les 2 timers et les boutons poussoirs et l'écran LCD afin de permettre à l'utilisateur de régler le simulateur de manière simple.

Le projet est constitué de 2 cartes : Une carte comprenant le micro-contrôleur, l'alimentation, le générateur de courant de la LED et l'écran LCD et une carte comprenant les 6 boutons permettant une interface avec l'utilisateur pour les réglages et la navigation entre les menus.

Durant le conception, nous avons rajouté la fonction du rétro-éclairage de l'écran LCD et nous avons remplacé le LM2574 par un LM2575 pour fournir assez de courant au montage. Nous avons aussi du faire face à quelques problèmes comme des défauts d'insolation, un strap sur la carte de l'ATMEGA8535 du à la complexité du typon et enfin l'empreinte des boutons poussoirs sous Layout qui n'existait pas. Les tests effectués sont le test du programme sur une carte de test, l'alimentation de la LED et l'alimentation globale du montage. Le dernier test sera de faire fonctionner les divers éléments ensemble.

Index des illustrations

Annexe 1 : Article tiré de la revue Elektor

Source de courant simplifiée pour LED

Rainer Schuster (Allemagne)

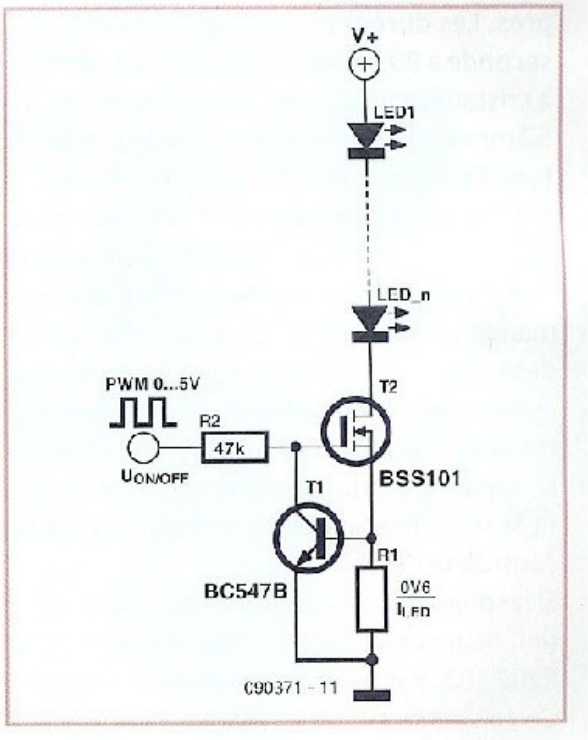
Actuellement sortent à tout bout de champ des dispositifs de plus en plus sophistiqués dédiés à l'activation des LED à courant constant. Ceux qui préfèrent les choses simples et bon marché trouveront leur bonheur dans le circuit présenté ici. Le courant traversant la diode provoque une chute de tension sur R1, qui à partir des 0,6 V de la tension base-émetteur sur T1 abaisse la tension grille-source de T2, de sorte qu'un courant constant de I = 0,6 V/R1 traverse les LED.

L'entrée de commande permet d'allumer les LED avec une plage de tension de l'ordre de 5 V à 12 V, ou de les éteindre avec une tension de 0 V. Il est également possible de contrôler la luminosité des LED en appliquant sur cette entrée une tension rectangulaire modulée en largeur (MLI). La tension d'alimentation des LED en série peut être presque aussi élevée que l'on veut, mais tant que la tension maximale admissible drain-source n'est pas dépassée !

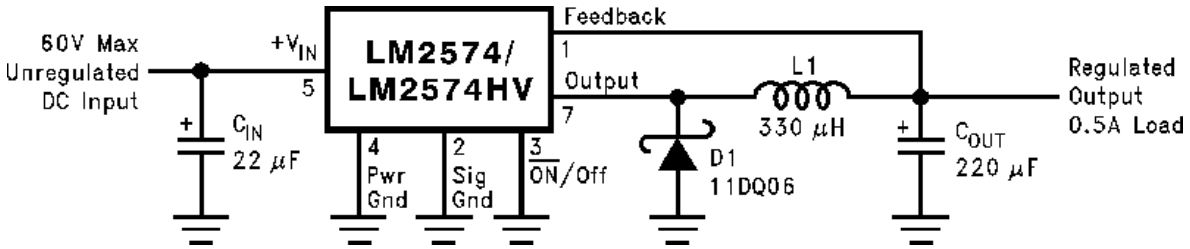
Lors du choix de T2 et du calcul d'un éventuel refroidissement, il faut prendre en compte la perte subie :

(Tension d'alimentation *moins* Tension sur les LED) x I_{LED}

(090371-I)



Annexe 2 : Schéma de montage du LM2574



Annexe 3 : Le programme complet du simulateur d'aube

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project : Aube
Version : 1.0
Date    : 27/09/2010
Author  : GEII
Company : IUT Tours
Comments: Projet d'études et réalisations de Antoine Bordier et Alexis Bourgueil


Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************************/

#include <mega8535.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <stdio.h>
#include <delay.h>

// Définition des boutons par rapport aux sorties de l'ATMEGA8535

#define BpMode PIND.2
#define BpH PIND.6
#define BpM PIND.5
#define BpReveil PIND.4
#define BpStop PIND.3

// Déclaration des variables

int seconde;
int minute;
int heure;
unsigned char heure_txt[13];
```

```
int minrev;
int heurev;
unsigned char hrev_txt[16];

int mode;
unsigned char mode_txt[16];

int reveil;
unsigned char reveil_txt[16];

int duree;
unsigned char rampe_txt[16];

int hdep;
int mindep;

int ok;


// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here

// Utilisation du Timer 1 pour créer une horloge qui incrémente l'horloge toutes les secondes

   seconde++;
   if (seconde>59)   // Au bout de 60 secondes, les secondes sont remises à zéro et les minutes sont incrémentés de 1
   {
            seconde=0;
            minute++;
   }

   if(minute>59)  // Au bout de 60 minutes, les minutes sont remises à zéro et l'heure est incrémenté de 1
   {
            minute=0;
            heure++;

                     if(ok==1)
                     {
                     OCR2=OCR2+(255/duree); // Incrémentation du Timer 2 pour alimenter la LED
                     if(OCR2>255)
                              {
                              OCR2=255;
                     }
            }
   }

   if(heure>23)   // Au bout de 24h, l'heure est remise à zéro
   {
            heure=0;
```

```
   }
}

// Declare your global variables here

void main(void)
{


// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x80;  // OCR2 en sortie !

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
```

```
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x0D;    // Compteur bloqué 15625 (3d09 en héxadécimal) pour obtenir une interruption
TCNT1H=0x00;    // toutes les secondes pour incrémenter l'horloge
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3D;
OCR1AL=0x09;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x61;   // Timer 2 configuré en MLI pour alimenter la LED
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

while (1)
      {
      // Place your code here
      while(BpMode == 0)
      {
```

```
  mode++;
        lcd_clear();     // Incrémentation de la variable "mode" pour naviguer entre les différents modes
        delay_ms(500);
        break;
}

 while(BpReveil == 0)
{
  reveil++;
  ok=0;
  OCR2=0;         // Mise en marche ou à l'arrêt du réveil
        lcd_clear();
        delay_ms(500);
        break;
}

if(BpStop == 0)
          {
        ok=0;           // Extinction du réveil
        OCR2=0;
}

if(reveil >1)
          {
        reveil = 0;    // Remise à zéro de la variable "réveil"
}

if(mode>3)
          {
        mode=0;          // Remise à zéro de la variable "mode"
}

if(reveil == 1)
          {
        sprintf(reveil_txt,"reveil on");
}
                                  // Affichage de l'état du réveil
if(reveil == 0)
          {
        sprintf(reveil_txt,"reveil off");
}

if(duree<15)
          {
        duree=15;
}               // Blocage de la durée de rampe entre 15 et 45 minutes

if(duree>45)
          {
        duree=45;
}
```

```
if((hdep==heure)&&(mindep==minute)&&(reveil == 1))
        {
        ok=1;      // Démarrage du révéil a l'heure prévue
}

switch(mode)
        {
        case 0 :
                sprintf(heure_txt,"%02d : %02d : %02d",heure,minute,seconde);
                lcd_gotoxy(0,2);
                lcd_puts(heure_txt);

                sprintf(mode_txt,"mode normal");    // Mode affichage de l'heure
        lcd_gotoxy(0,0);
        lcd_puts(mode_txt);

        lcd_gotoxy(6,3);
        lcd_puts(reveil_txt);

                break;
        case 1 :
        sprintf(mode_txt,"mode heure ");
        lcd_gotoxy(0,0);
        lcd_puts(mode_txt);
        if(BpH == 0)
                                        {                       // Mode Réglage de l'heure avec affichage de l'heure réglée
        heure++;            // Un appui sur BpH incrémente l'heure de 1
        delay_ms(250);
        }
         if(BpM == 0)
                                        {           // Un appui sur BpM incrémente les minutes de 1
        minute++;
        delay_ms(250);
        }
        sprintf(heure_txt,"%02d : %02d : %02d",heure,minute,seconde);
                lcd_gotoxy(0,2);
                lcd_puts(heure_txt);
                break;
        case 2 :
        sprintf(mode_txt,"mode reveil ");
        lcd_gotoxy(0,0);
        lcd_puts(mode_txt);

        if(BpH == 0)
                                        {                       // Mode de réglage du réveil avec affichage
        heurev++;           // Un appui sur BpH incrémente l'heure de réveil de 1
        if(heurev>23)
                                                {
                heurev=0;
        }
```

```
        delay_ms(250);
        }
         if(BpM == 0)
                                        {               // Un appui sur BpM incrémente les minutes de réveil de 1
        minrev++;
        if(minrev>59)
                                                        {
                        minrev=0;
        }
        delay_ms(250);
        }
        sprintf(hrev_txt,"%02d : %02d ",heurev,minrev);
                        lcd_gotoxy(0,2);
                        lcd_puts(hrev_txt);

                        lcd_gotoxy(6,3);
        lcd_puts(reveil_txt);

        break;
case 3 :
        sprintf(mode_txt,"mode rampe"); // Mode de réglage de durée de rampe avec affichage de la durée
        lcd_gotoxy(0,0);
        lcd_puts(mode_txt);

        if(BpH == 0)
                                        {
        duree=duree+5;          // Un appui sur BpH augmente la durée de rampe de 5 min
        delay_ms(250);
        }

        if(BpM == 0)
                                        {
        duree=duree-5;          // Un appui sur BpM réduit la durée de rampe de 5 min
        delay_ms(250);
        }

        sprintf(rampe_txt,"%02d min",duree);
                        lcd_gotoxy(0,2);
                        lcd_puts(rampe_txt);

                        hdep=heurev;
        mindep=minrev;

                        if(duree>mindep)                        // Calcul de l'heure de départ du réveil
                                {
                                hdep=((hdep+24)-1)%24;
                                mindep=((mindep+60)-duree);
                        }

                        else
                                {
```

```
                    mindep=mindep-duree;
                }

            break;
            }

        }
    }
```