Université François Rabelais de Tours
Institut universitaire et technologique de Tours
Département Génie Électrique et Informatique Industrielle

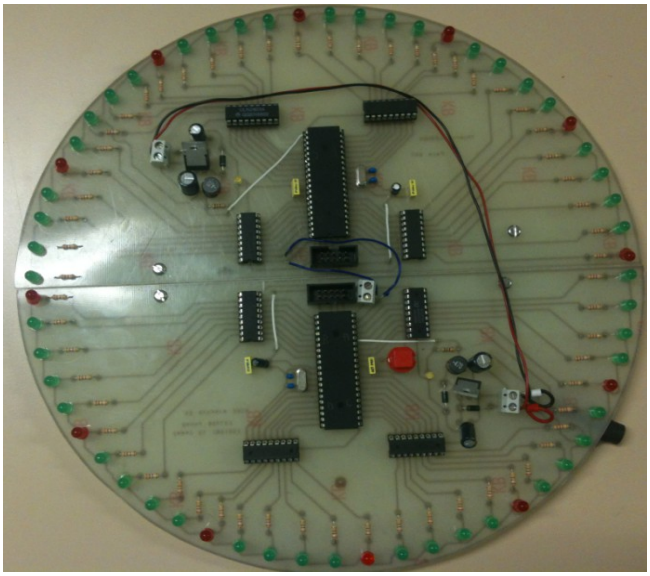


Illustration 1: Horloge à Led

Horloge à LED piloté par micro-contrôleur

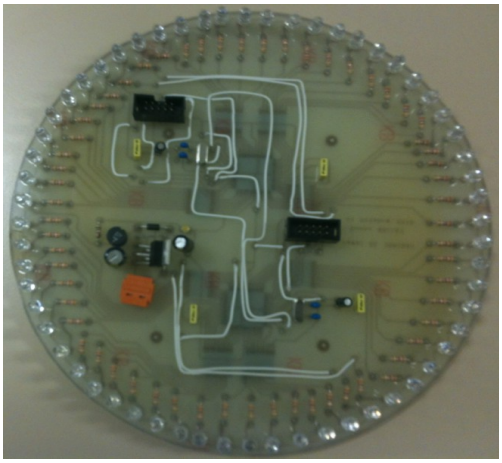


Illustration 2: Horloge à LED CMS

Étude et réalisation
ER-ISI & ER-GE

Sonny BELLET
Jamal EL IDRISSI
K4A
Promotion 2010/2013

Enseignants
M. Thierry LEQUEU
M. Philippe AUGER

Université François Rabelais de Tours
Institut universitaire et technologique de Tours
Département Génie Électrique et Informatique Industrielle

Horloge à LED piloté par micro-contrôleur

Sonny BELLET
Jamal EL IDRISSI
K4A
Promotion 2010/2013

Enseignants
M. Thierry LEQUEU
M. Philippe AUGER

Remerciements

Au nom de notre équipe de travail, nous tenons à remercier toutes les personnes qui ont contribué à la réalisation de notre projet tuteuré.

Nos remerciements s'adressent particulièrement à Monsieur Thierry LEQUEU et Monsieur Michel ZAPATA, pour leurs précieux conseils, mais également pour leur écoute et leur disponibilité.

Nous remercions le corps enseignant de l'Institut Universitaire de Technologie de Tours, pour les connaissances transmises au cours des deux dernières années.

Nous adressons également, nos remerciements à M. Richard VAUTIER, magasinier, pour sa collaboration.

Sommaire

Introduction

Au cours du semestre 4 de notre D.U.T. nous avions à réaliser un projet mettant en œuvre un micro-contrôleur. Pour cela nous avions 8 semaines à raison de 2 fois 3h30min d'étude et réalisation par semaine pour réaliser ce projet, de plus, des séances en autonomie ont été misent en place afin de nous préparer au travail seul, comme les conditions du stage en entreprise qui se déroulera à la fin de notre semestre.

Composée de Jamal EL IDRISSI et Sonny BELLET, notre équipe en binôme a donc choisi de faire fonctionner une horloge à LED par le biais d'un ATmega8535, car nous trouvions cette idée intéressante pour apprendre la programmation sur un micro-contrôleur et faire la réalisation de typon plus complexe.

Dans un premier temps nous présenterons notre projet avec ses objectifs et ses limites, puis nous parlerons des principaux composants utilisés dans notre montage et leur fonction, ensuite nous montrerons la réalisation et l'évolution de nos cartes électroniques. Enfin nous finirons par expliquer la programmation de notre horloge avec une présentation du logiciel et des fonctions utilisées.

1.Cahier des charges

1.1.Principe de fonctionnement

Le but de notre projet est de faire fonctionner une horloge à LED en contrôlant les LED représentant les minutes et les heures par un ATmega8535. Pour cela nous devrons concevoir une carte électronique ayant la forme d'une pendule et nous devrons ensuite concevoir un programme qui permettra de contrôler ces LED. Nous utiliserons 60 LED pour représenter chaque minute et nous utiliserons sur ces 60Leds, 1 LED sur 5 qui sera d'une autre couleur pour représenter les heures en la faisant clignoter.

1.2.Contraintes du projet

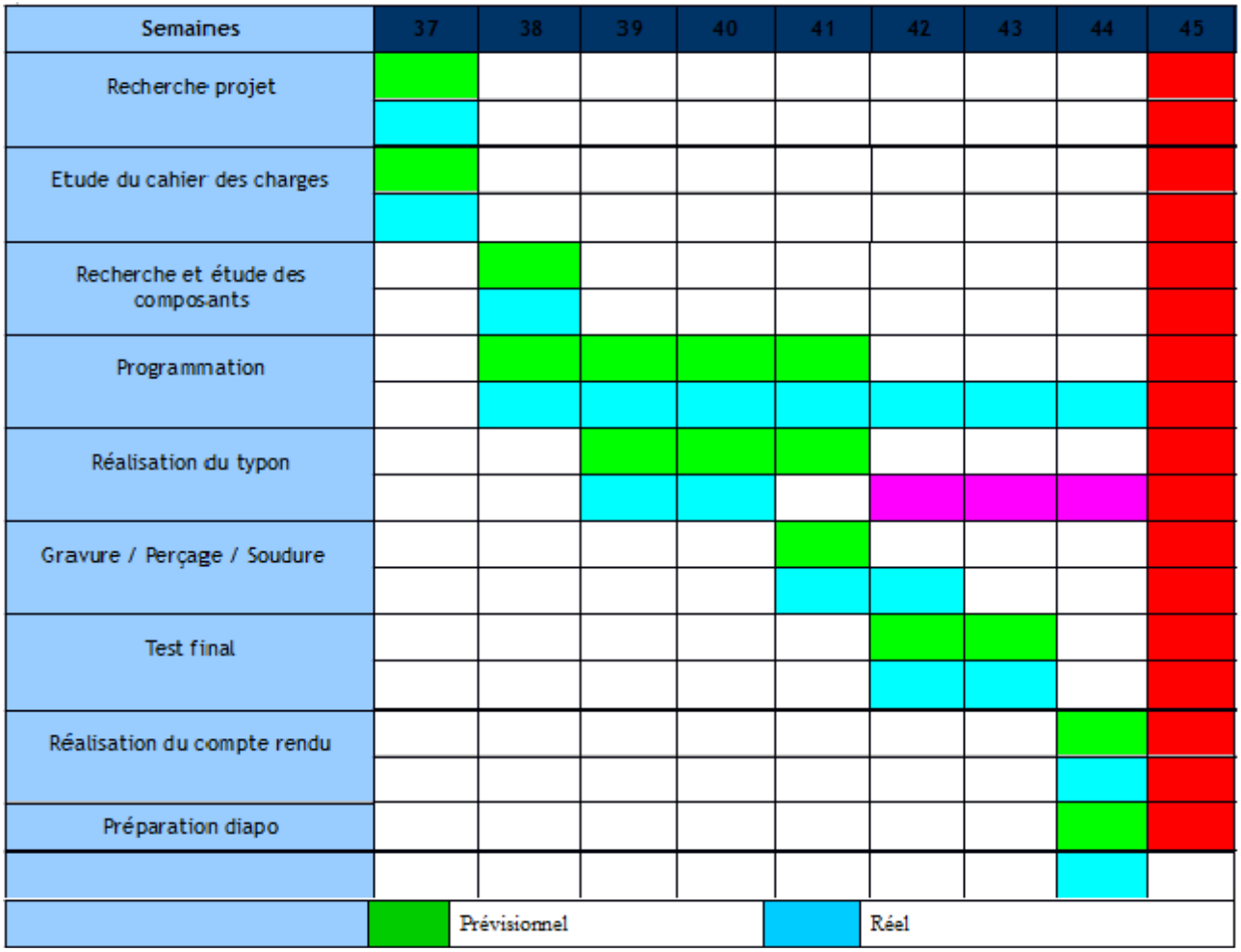
Afin que le projet se déroule dans de bonne condition et qu'il puisse fonctionner à la fin des 8 semaines, nous avions des contraintes, telles que les dates à respecter pour rendre le rapport et faire un exposé oral. Ainsi que les heures de projet qui nous étaient consacrées pour réaliser notre horloge à LED. Nous devions donc nous répartir le travail à effectuer sur ce projet, pour cela Jamal s'est occupé en grande partie de la partie réalisation de la carte électronique et Sonny s'est plus penché sur la programmation de l'horloge.

De plus nous avions une contrainte de budget, nous ne devions pas avoir un coût trop excessif pour un projet d'étudiant en 2° année.

Nous avions aussi une contrainte matériel qui est qu'un ATmega8535 ne peut commander que 32 entrées/sorties. Nous devions donc prendre 2 micro-contrôleurs afin de disposer de 64 entrées/sorties pour gérer les 60 LED qui représentent les minutes et les heures et nous pourrons ainsi faire communiquer les 2 ATmega.

1.3.Planning prévisionnel et réel

Pour l'avancement de notre projet, nous avions réalisé au début des séances d'études un planning prévisionnel qui nous permettait de nous projeter dans l'avenir et qui nous a permis de nous fixer des objectifs à réaliser.



On peut voir sur le planning ci-dessus les différentes étapes que nous avons dû réaliser lors de notre projet avec les différents délais de chaque étape du projet. On peut aussi voir qu'il y a différentes couleurs sur notre planning qui correspondent au temps que nous pensions passer sur chaque étape, et au temps que nous avons réellement passé sur chacune des étapes et enfin les vacances et les actions réalisées en plus, car il nous restait du temps. Il y a de plus les cases roses qui correspondent à la réalisation de la carte électronique avec les composants en CMS qui ne figuraient pas sur notre projet de départ car nous ne pensions pas la concevoir, hors Jamal ayant trouvé le temps nécessaire à la conception, il l'a réalisé.

On peut voir sur ce planning qu'il y a des différences entre le planning prévisionnel et le réel ce qui est tout à fait normal, car on ne peut pas savoir quels seront les inconvénients que nous rencontrerons lors de la réalisation de notre projet, telle que la prise en main du logiciel de programmation et la communication entre les 2 ATmega qui a été plus longue que prévu, ainsi que les erreurs de câblage de la carte électronique.

2.Les principaux différents composants utilisés et leur fonction

2.1. Alimentation 5V par régulateur LM2574

Pour pouvoir créer notre horloge nous aurons besoin d'une source d'alimentation continue pour alimenter nos LED ainsi que l'ATmega8535 pour gérer l'allumage de celle-ci.

C'est pourquoi nous avons mis en place une alimentation 5V qui nous a permis de réduire notre tension d'entrée 12V à l'aide d'un montage que nous avons vu au cours du semestre 2. Ce montage est composé d'un régulateur à découpage LM2574N-5.0 qui est contrôlé par la tension d'entrée qui peut être variable en fonction des conditions d'alimentation.

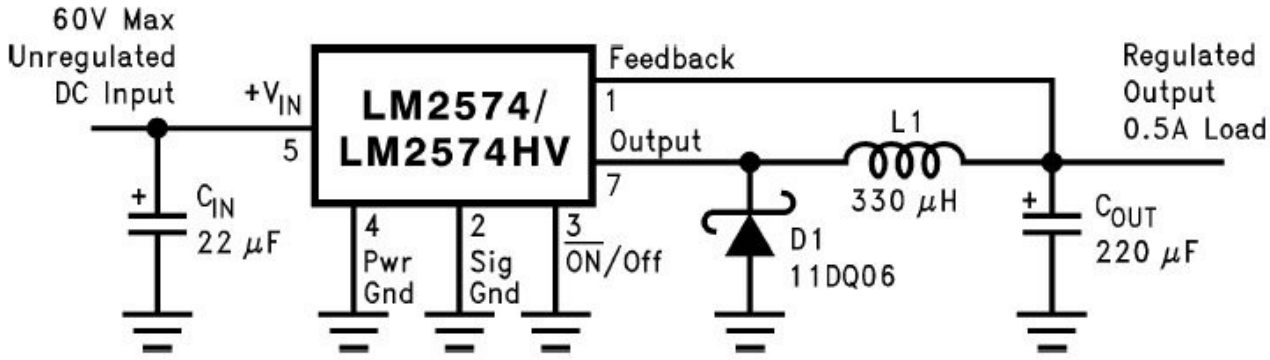


Illustration 3: régulateur LM2574

Cette tension entre dans la patte V_{in} du régulateur et passe directement dans un transistor intégré au régulateur qui agit comme un interrupteur. Cette valeur sort à la patte V_{out} du régulateur et passe dans une diode qui elle aussi agit comme un interrupteur mais de façon complémentaire au transistor. Lorsque le transistor est passant, la diode est bloquée[1]. Cette tension passe aussi dans un circuit qui la filtre afin d'obtenir la tension la plus lisse et continue possible. Ce circuit est composé d'une inductance et d'un condensateur, ce qui crée un filtre passe-bas[2]. Le transistor suivi de la diode avec l'inductance « L1 » et le condensateur « Cout » peut représenter un montage de type hacheur Buck. Son rôle est d'abaisser la tension par découpage.

Le découpage est réalisé avec la fermeture et l'ouverture successif du transistor et de la diode. Ils agissent comme des interrupteurs contrôlés à très haute fréquence.

1 Passant signifie comme un interrupteur fermé, le courant peut donc passer et Bloqué signifie comme un interrupteur ouvert, le courant ne circule donc pas.

2 Filtre passe-bas : permet de laisser passer seulement les basses fréquences.

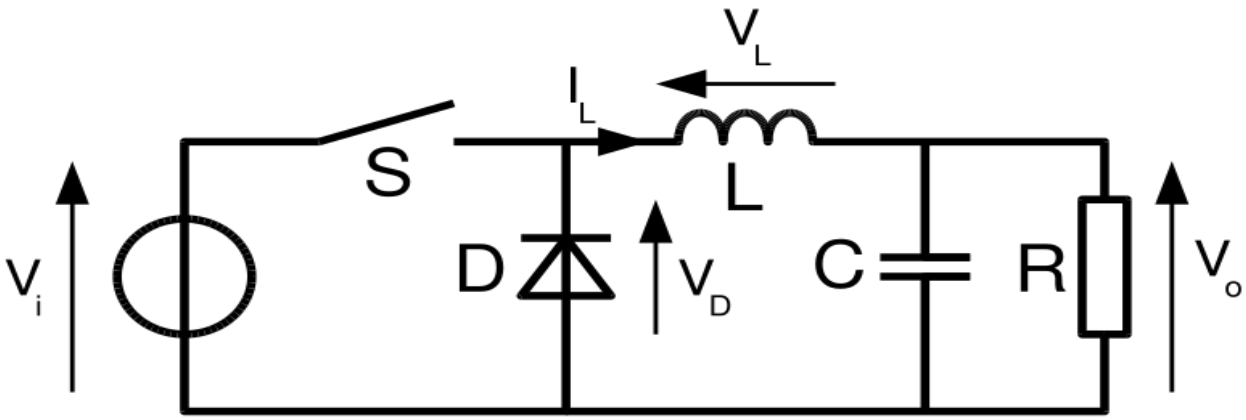


Illustration 4: Hacheur Buck

2.2. Micro-contrôleur Atmega8535

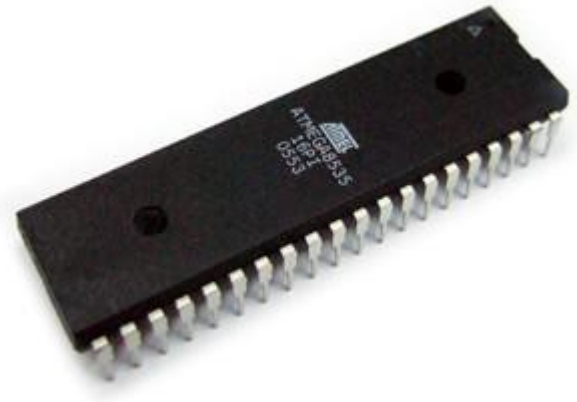


Illustration 5: Composant ATmega8535

L'ATmega8535 est un micro-contrôleur. C'est un circuit intégré qui rassemble la gestion en tension, les interfaces d'entrées/sorties et le contrôle de la vitesse de transmission des données.

Il dispose de 4 ports différents : A, B, C, D. C'est 4 ports peuvent être réglés en entrée ou en sortie en fonction du besoin de notre projet. Pour notre projet, nous avions besoin de mettre les 4 ports en sortie. Il nous suffira juste de configurer un bit d'un port en entrée, pour faire communiquer les 2 ATmega, ainsi les micro-contrôleurs pourront envoyer et recevoir une information.

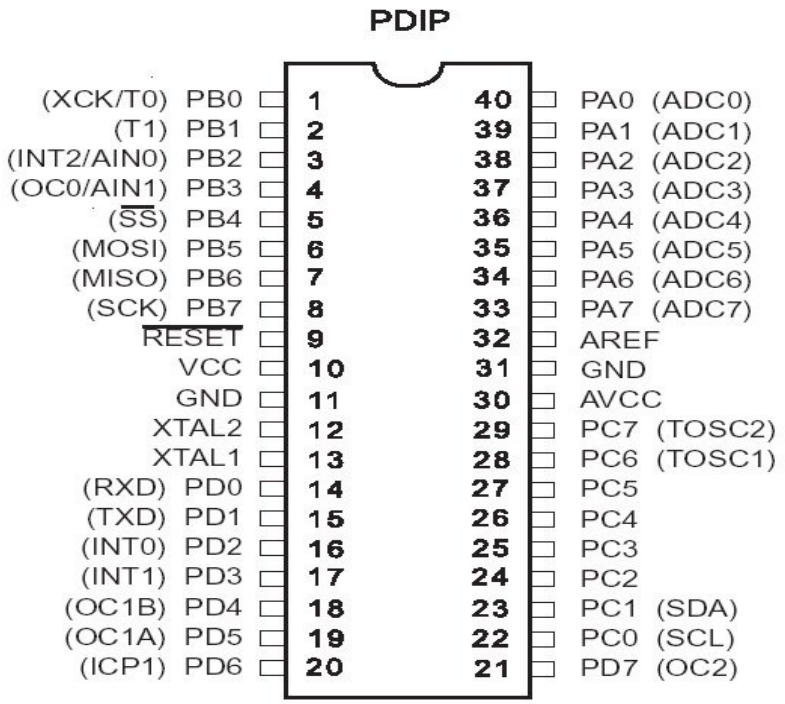


Illustration 6: ATmega8535

Ces quatre ports ont des fonctions différentes selon si ceux-ci sont configurés en entrée ou en sortie. Par exemple le port A est uniquement dédié à la conversion analogique-numérique, tandis que le port D lui sert au transfert d'information et au comparaison de valeurs sur les pins 18 et 19 de l'ATmega8535.[TL1]

L'ATmega8535 possède 40 broches, les 4 différents ports comportent chacun 8 broches d'entrées ou de sorties selon les besoins du programmateur(au total 32 broches pour la programmation). Il reste 8 autres broches qui vont nous servir à plusieurs fonctions. Tout d'abord il y a la broche 10 « Vcc » qui va nous permettre d'alimenter le composant par l'intermédiaire du connecteur ISP, les broches 11 et 31 « GND » qui vont nous servir à relier les masses, la broche 9 « Reset » qui nous sera utile si nous voulons supprimer le programme en cours d'exécution en remettant à 0 les données de l'ATmega8535. Nous avons ensuite les broches XTAL1 et XTAL2 qui vont nous permettre d'y relier notre Quartz.

Le Quartz va nous être utile lors de la partie programmation, car il va permettre de gérer la fréquence d'horloge de l'ATmega, la vitesse à laquelle on transmet les données. Et enfin, il y a 2 broches (AREF et AVCC) mais qui ne sont pas utile dans notre cas, car ces broches permettent de gérer la partie convertisseur analogique-numérique.

Les broches 6 « MOSI », 7 « MISO » et 8 « SCK » seront expliquées au point 2.4 de ce rapport. Elles sont reliées au connecteur ISP sur les mêmes broches de ce connecteur.

2.3. Transistors Darlington ULN2803a

Illustration 7: Composants ULN 2803

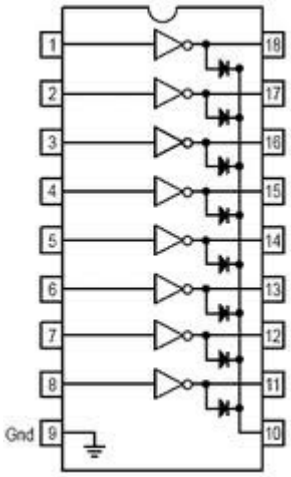


Illustration 8: ULN2803a

Le circuit ULN2803 est un réseau d'amplification Darlington (double amplification par transistors). C'est une combinaison de deux transistors bipolaires de type semblable (NPN ou PNP). Le gain en courant de ce montage sur chaque broche est égal au produit du gain des deux transistors. Il permet donc à notre circuit de disposer de plus de courant et ainsi alimenter plus de sortie.

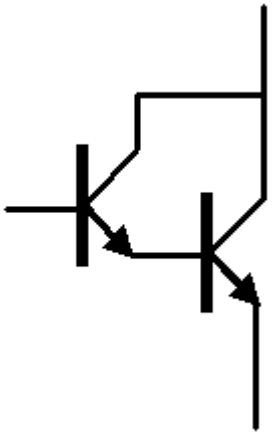


Illustration 9: transistor Darlington PNP

Sur notre projet, là où l'ATmega8535 ne pouvait pas fournir assez de courant pour allumer toutes les LED simultanément, avec l'ULN2803a, on peut maintenant faire fonctionner toutes les LED avec un courant suffisant fourni par ces transistors à Darlington. On reliera donc une à une chaque broche de l'ULN2803a aux broches des ports de l'ATmega8535.

Dans le composant, une diode est mise en série aux deux transistors afin de les protéger contre des retours de courant.

2.4.Le connecteur ISP

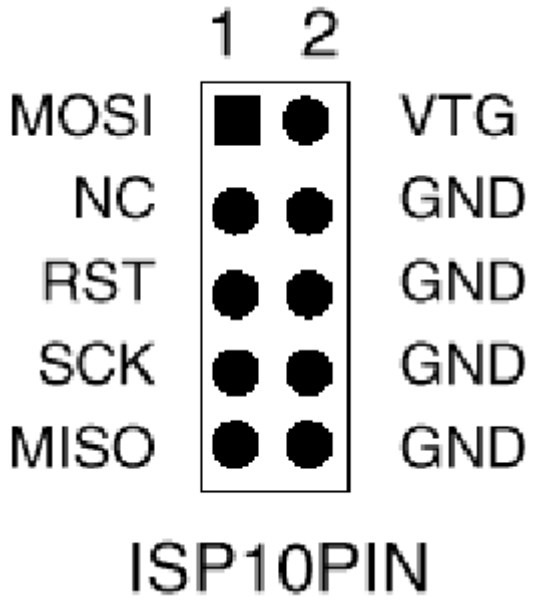


Illustration 10: Connecteur ISP 10 broches

Le connecteur ISP est la liaison qui communiquera avec le port parallèle de l'ordinateur par l'intermédiaire du 74LS245, ce sont 8 « buffers[3] bidirectionnel Tri-State non inverseur » et à pour rôle l'échange de données.

3 Tampons

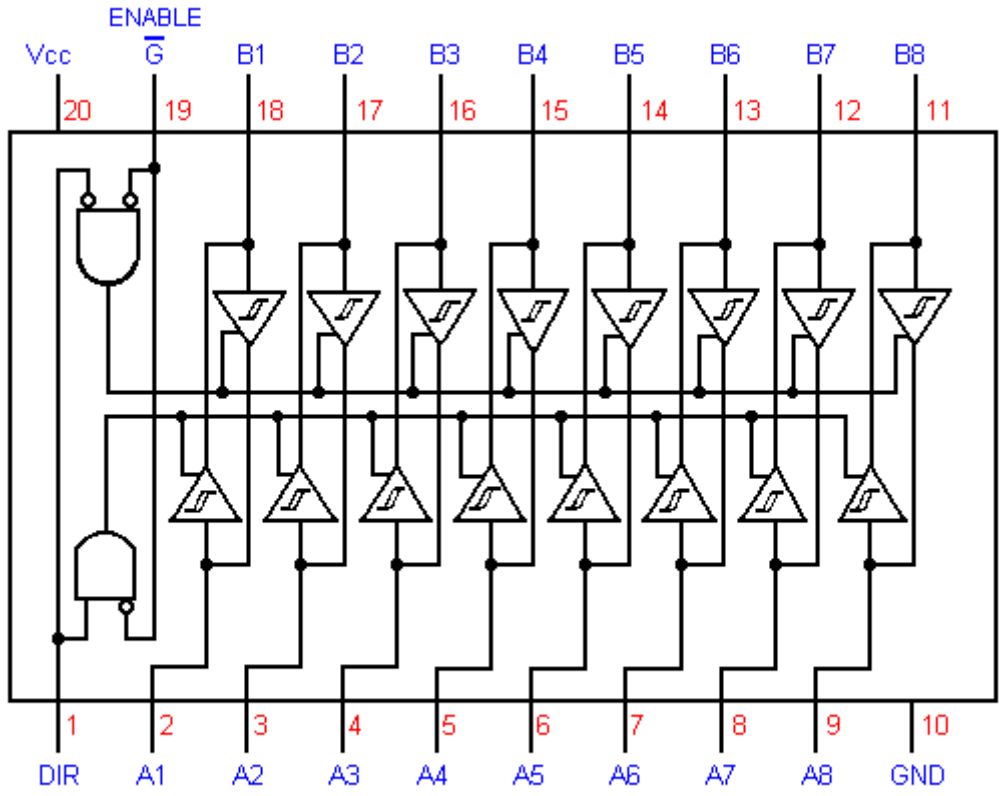


Fig. 14. - Schéma interne du circuit intégré 74LS245
(huit buffers bidirectionnels TRI-STATE).

Illustration 11: 74LS245

Ce circuit intégré est utilisé car le connecteur ISP consomme beaucoup de mémoire lors de ces transferts d'informations et le 74LS245 dispose d'une technologie qui fait que la sortie ne dépend pas que de l'état de l'entrée. Elle dépend aussi d'une autre variable qui indique que la transmission est possible seulement si celle-ci est à « 1 », un peu comme un interrupteur. Comme on peut le voir sur l'illustration ci-dessus, les informations sont bidirectionnelles et sont aussi contrôlées par les portes logiques ET qui sont reliées à la broche 1 et 19 qui envoie ou non un signal permettant l'envoi de données.

Voici le symbole d'un AOP à trois états :

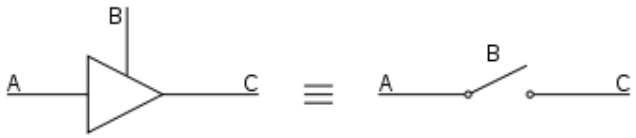


Table de vérité :

Entrées		Sortie
A	B	C
0	0	Z
1		
0	1	0
1	1	1

Illustration 12: sortie à 3 états

Nous allons utiliser un connecteur ISP 10 broches sur notre montage. Il est composé de 4 broches qui représentent la masse « **GND** », d'une broche « **VTG** » qui est le +5V. Il y a ensuite « **MOSI** » qui signifie « Master Out Slave In » qui envoie des données de l'ordinateur(maître) vers notre micro-contrôleur(esclave), puis la broche « **NC** » qui signifie que la broche est non-commandée ici. Ensuite il y a le « **RST** », le Reset permet de réinitialiser le système. Il y a le « **SCK** » qui est le système d'horloge de notre système qui contrôle la vitesse de transmission de données. Et enfin le « **MISO** » qui signifie le « Master In Slave Out » , cette broche envoie des données du micro-contrôleur(maître) vers l'ordinateur(esclave).

On voit donc que ce composant va permettre la communication entre l'ATmega8535 et l'ordinateur pour la programmation de notre projet en reliant chaque broche à celle correspondante sur le micro-contrôleur.

3.Réalisation du montage

3.1.Le carte électronique normal

Pour réaliser notre carte, nous avons travaillé tout d'abord sur le logiciel Orcad Capture CIS pour pouvoir dessiner notre montage et réaliser les différentes connections entre les composants. Nous avons mis en haut à gauche de notre feuille de travail dans le cadre rouge la partie alimentation 5V pour pouvoir facilité son repérage. Puis pour simplifier la lecture de notre schéma nous avons mis en place des groupes de 8 LED pour pouvoir différencier les différents ULN2803a et les LED associées.

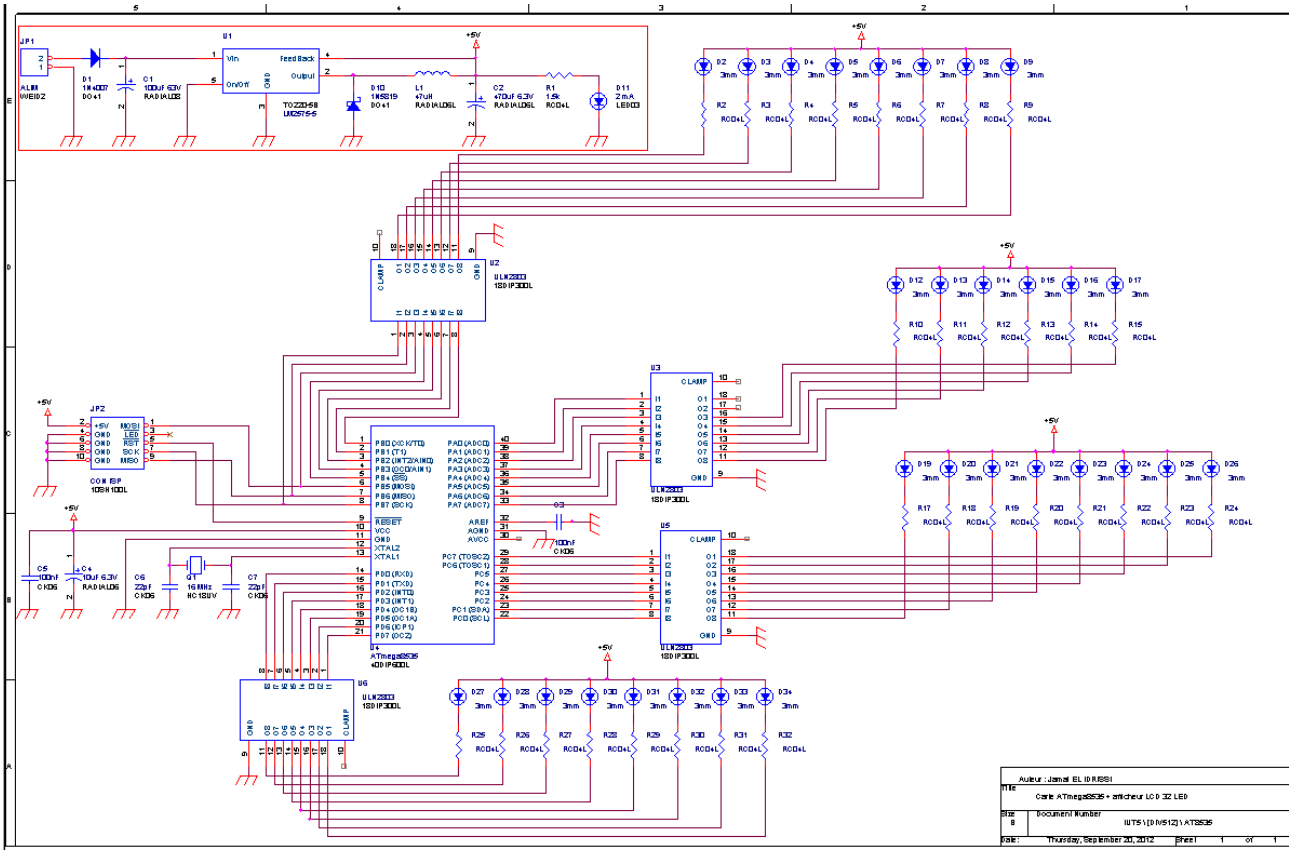


Illustration 13: Schéma Kicad du montage

Lors de la réalisation de la première horloge avec des composants de taille normale, nous avons dû réfléchir un minimum avant de se lancer dans la réalisation du typon, car nous devions réaliser notre typon de façon à obtenir une horloge ronde et donc penser au placement des 60 LED pour obtenir un rond idéal.[TL2]

Nous avons dû calculer à l'aide d'un logiciel de calcul, « LibreOfficeCalc », la position de chaque LED et pour simplifier nos calculs nous avons mis en place un tableau à 2 dimensions comportant les données sur la taille de la carte et l'écart entre chaque LED pour pouvoir les placer correctement.

(cf. Annexe 3)

Sachant qu'un quart de cercle parcours un angle de 90 degrés et que sur une horloge normale il y a 15 arrêts d'aiguilles pour afficher les minutes et les heures, nous avons pu en déduire que nous aurions besoin de 15 LED pour un quart de l'horloge, nous avons donc séparé l'angle du cercle sur lequel nous déposerons les LED divisées par le nombre de LED concernées est nous obtenons un angle de 6 degrés. Il nous a donc fallu mettre un écart de 6 degrés entre chaque LED pour que notre cercle de LED ressemble à une vraie horloge. Mais avant de pouvoir placer nos LED sur la carte nous avons dû calculer l'emplacement exact de chaque LED, pour cela nous avons utilisé les formules concernent les positions en x et en y dans un cercle.

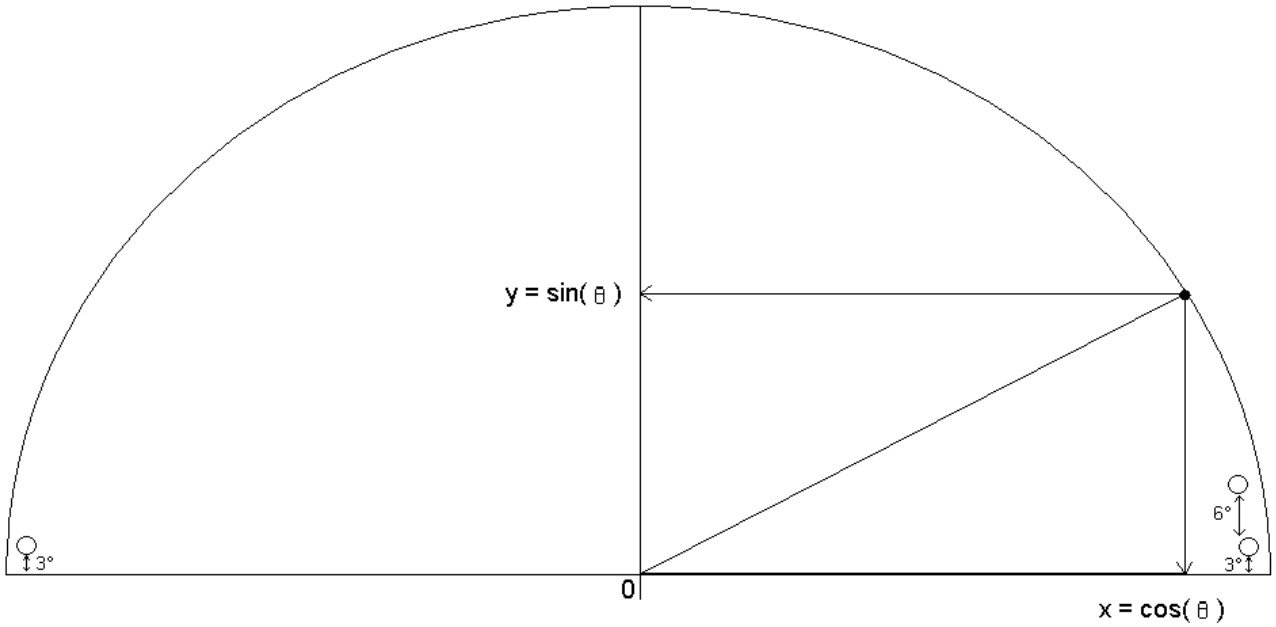


Illustration 14: Schéma de positionnement des LED sur la carte électronique

Lors du démarrage de notre carte nous nous sommes renseigné au prêt du magasinier M. VAUTIER pour savoir s'il pouvait nous donner une carte suffisamment grande pour contenir tous nos composants mais il nous a dit que la taille maximale qu'il avait en stock était de 200 × 300 cm. Or il n'était pas envisageable pour nous de faire notre carte sur une aussi petite taille comme nous devions utiliser 60 LED et 2 ATmega, nous avons décidé de mettre en place deux cartes composées chacune d'un ATmega, de 4 ULN2803a, de 30 LED et d'une alimentation 5V pour alimenter nos LED.

Nous avons donc décidé de réaliser sur chaque carte une demie horloge pour cela nous avons créé sur le logiciel Orcad un cercle qui serait la base de notre carte que nous avons ensuite séparée en deux parties et nous avons donc travaillé sur une seule partie, un demi-cercle

Après avoir sorti notre carte, nous avons dû tout d'abord faire plusieurs découpes avant de souder nos composants, car la carte que nous possédions était de forme rectangulaire et nous voulions un cercle nous avons alors dû passer du temps sur la découpe de la carte. Nous avons ensuite limé le contour de la carte pour obtenir une surface lisse et bien arrondie.

Grâce aux calculs réalisés dans notre tableau et à une option de placement du logiciel Orcad, nous avons pu mettre en place nos 30 premières LED de façon quasi automatique. Nous avons dû rentrer à la main chaque coordonnées des LED, mais les LED n'ont pas été disposées de façon aléatoire nous avons dû mettre en place les LED par groupe de 8, car nos ULN2803a permettent de gérer 8 LED à la fois et nos groupements sont mis de façon à obtenir les ports de l'ATmega dans l'ordre pour faciliter la partie programmation.

Pour le placement des LED nous avons mis chaque LED espacée de 6 degrés les unes des autres mais en ce qui concerne la première LED qui est au bord de la carte, elle a été placé à seulement 3 degrés du bord, car nous travaillons sur une seule partie de l'horloge. Ainsi lorsque l'on réunie les deux morceaux de l'horloge l'écart de 3 degrés de chaque cartes s'additionne et nous donne donc bien un total de 6 degrés entre la LED de la première carte et celle de la seconde. Nous avons par la suite pu disposer le reste des composants.

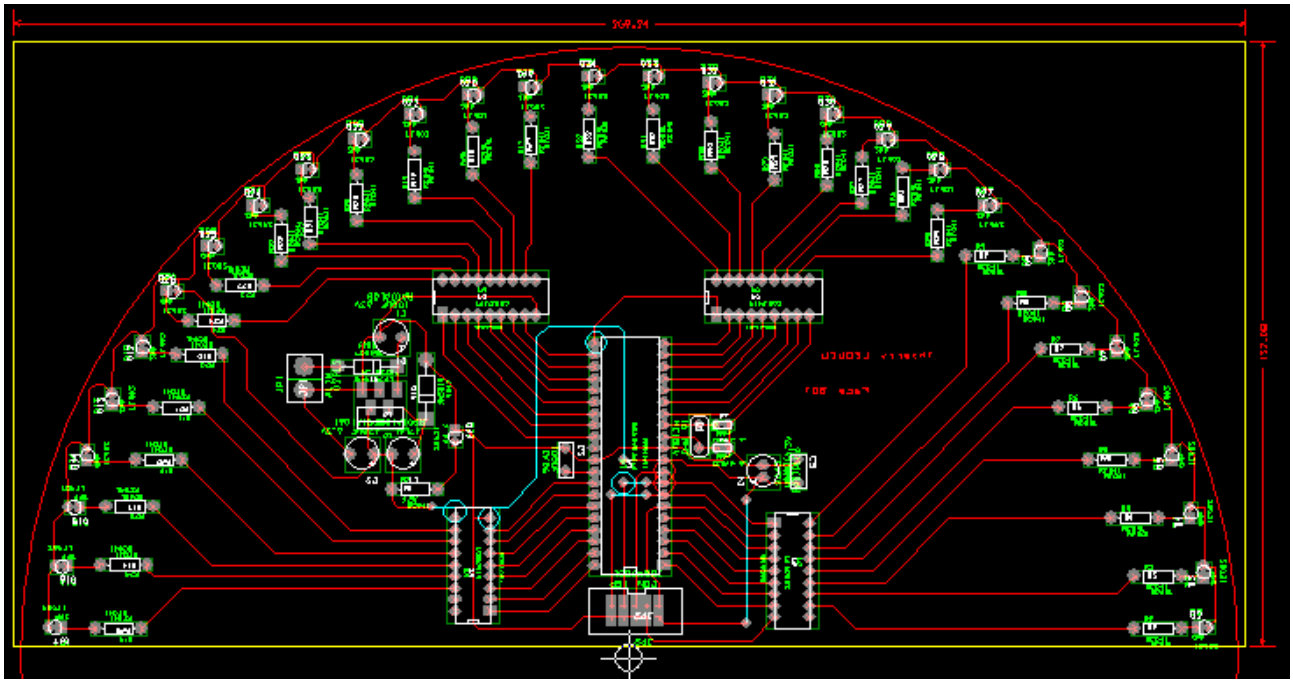


Illustration 15: Câblage de la moitié de carte électronique sur PCB Kicad

Après avoir réalisé notre carte, nous avons testé la partie alimentation 5 Volts pour s'assurer du bon fonctionnement grâce à une LED jaune qui nous permet de voir si notre alimentation fonctionne correctement. Nous avons ensuite envoyé un programme simple dans l'ATmega grâce au logiciel CodeVision AVR pour tester si toutes les LED allumaient et s'il n'y avait pas de court-circuit entre les différents composants.

3.2.Le carte électronique en CMS

Après avoir testé le bon fonctionnement de notre carte et comme ils nous restait encore 3 semaines de projet, nous avons décidé avec l'accord de notre professeur M.Thierry LEQUEU de réaliser une seconde carte électronique mais avec des composants de taille réduite que l'on appelle composant monté en surface (CMS). Cette seconde carte nous a permis de réduire la taille de la carte de façon à pouvoir la réaliser sur une seule carte de 200 x 300 cm. Mais l'inconvénient du CMS est la mise en réalisation du typon, car les composants sont beaucoup plus petit donc la taille des broches sont plus petites et moins espacées les une des autres, nous devions donc rétrécir la taille des pistes pour qu'elle ne se touche pas et pour que le routage de la carte soit plus simple.

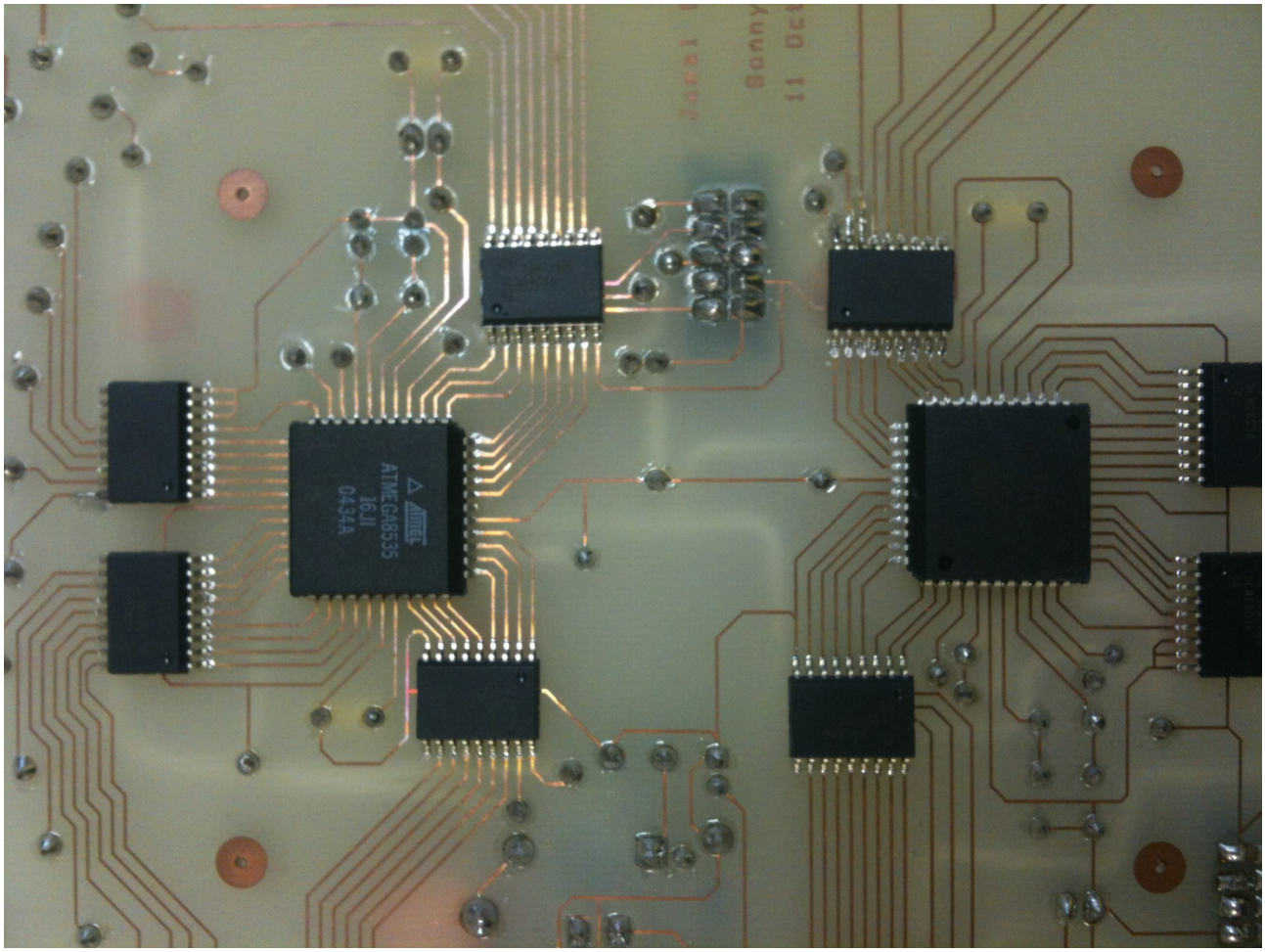


Illustration 16: Composants CMS de la carte électronique

L'avantage de ces composants en CMS sont la réduction de la taille des composants ce qui va nous permettre de réaliser notre typon sur une seule carte et mettre en place une seule alimentation 5 Volts. Nous avons dû rajouter des modifications à notre montage, car les empreintes des composants en CMS sont différentes, nous avons dû rajouter des masses supplémentaires et des borniers d'alimentation, ce qui nous a donc obligé à refaire la partie routage de la carte.

Lors de la réalisation de la carte, les composants nous ont posé quelques problèmes de routage, car les ATmega possédaient plus de pattes et moins d'espace pour le routage des composants qui entourent les micro-contrôleurs, nous avons donc dû mettre un grand nombre de straps[4] (fil qui traverse la carte du coté composant et non pas du coté des pistes cela nous permet de faire passer des fils à des endroits où le passage n'est pas possible soit, car il n'y a pas assez de place soit parce que les pistes aux alentours nous bloquent l'accès au composant voulu.)

4.Programmation de l'ATmega8535

4 Fil qui relie 2 composant par un câble volant et non par une piste cuivrée

4.1.CodeVisionAVR

Illustration 17: Logo CodeVisionAVR

Le logiciel CodeVisionAVR est un logiciel de programmation adapté spécialement pour programmer les micro-contrôleurs. Il nous permet de prendre en main rapidement la programmation avec une aide de différentes fonctions en nous créant automatiquement les fonctions que le l'on souhaite réaliser et utiliser en fonction du micro-contrôleur que l'on sélectionne par « Chip » et on règle la vitesse de fonctionnement de l'ATmega en fonction de la valeur de notre quartz dans « Clock »(comme sur l'illustration ci-dessous). Lorsque l'on générera un projet sous CodeVisionAVR, le logiciel ouvre cette fenêtre qui nous permettra d'ajouter à notre programme des fonctions. Il ne nous restera plus qu'à remplir les fonctions avec ce que l'on souhaite.

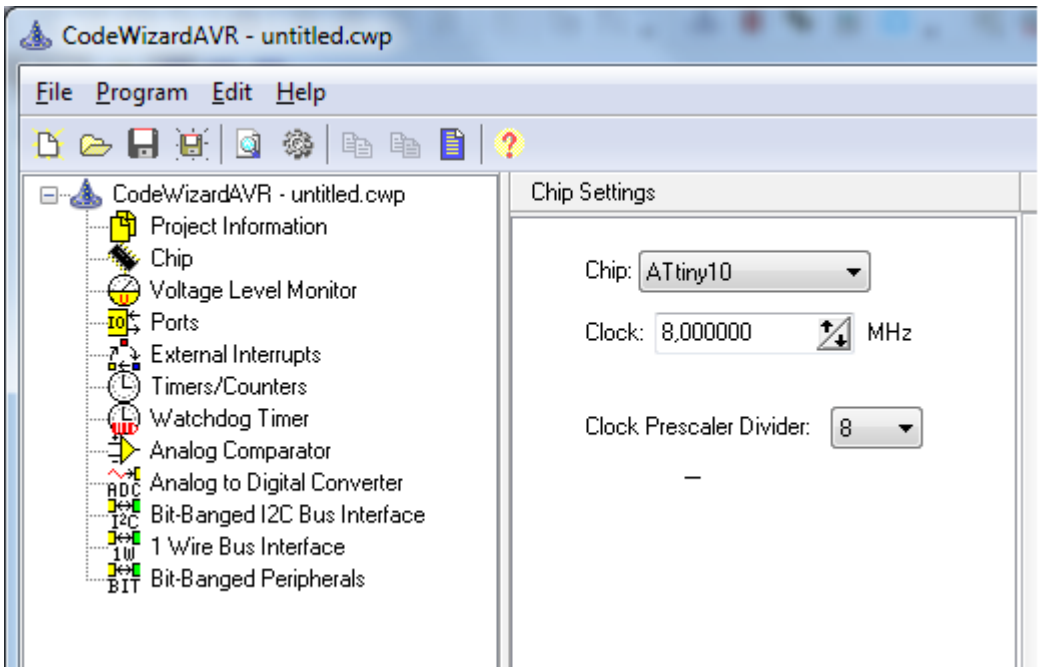


Illustration 18: Fenêtre de création d'un projet

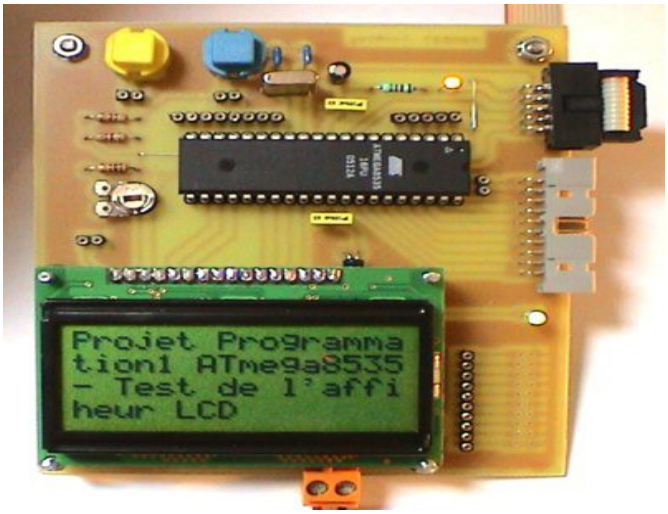


Illustration 19: Carte de programmation de l'ATmega8535 avec afficheur LCD

Pour débuter, nous avons tentés d'afficher des messages sur l'afficheur LCD de la carte de programmation fourni par l'IUT, avec la fonction écrite par le logiciel pour afficher sur l'écran LCD. [TL3]

```
/* initialise l'afficheur LCD pour 4 lignes & 16 colonnes */
lcd_init(16);

/* fonction qui indique à l'afficheur ou commencer l'écriture sur son écran */
lcd_gotoxy(0,0);
lcd_putsf("Projet Programmation1 ATmega8535 - Test de l'afficheur LCD");
```

Illustration 20: indication pour faire fonctionner l'écran LCD

Avec la première fonction, on initialise l'afficheur LCD en lui indiquant l'espace disponible pour l'écriture d'un message, nous pourrons donc afficher « 4*16=64 caractères » sur cet écran. Car l'afficheur dispose de 4 lignes d'écritures

Puis avec la deuxième fonction, on ordonne à l'afficheur le caractère de départ du message que l'on souhaite afficher à l'écran.

Illustration 21: Exemple d'affichage de l'heure sur un écran LCD

Nous avons ensuite essayés d'afficher sur l'écran LCD de la même carte de programmation un compteur pour que cela ressemble à une horloge à affichage numérique. Cela nous permettait de nous préparer à l'incrémentation de nos secondes, nos minutes et de nos heures.

Afin d'utiliser l'afficheur LCD, on doit configurer le programme pour mettre un PORT de l'ATmega8535 en sortie.

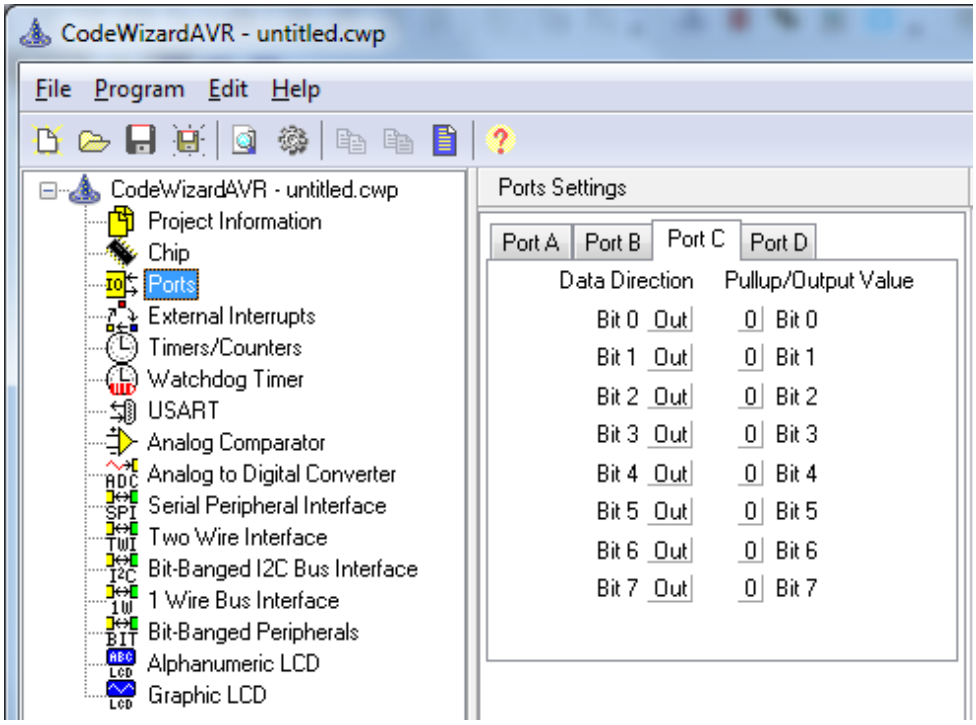


Illustration 22: PORTC en sortie

Pour cela, il suffit de mettre tous les Bits du PORTC par exemple en sortie comme ci-dessus.

Ce qui nous génère un code comme ceci :

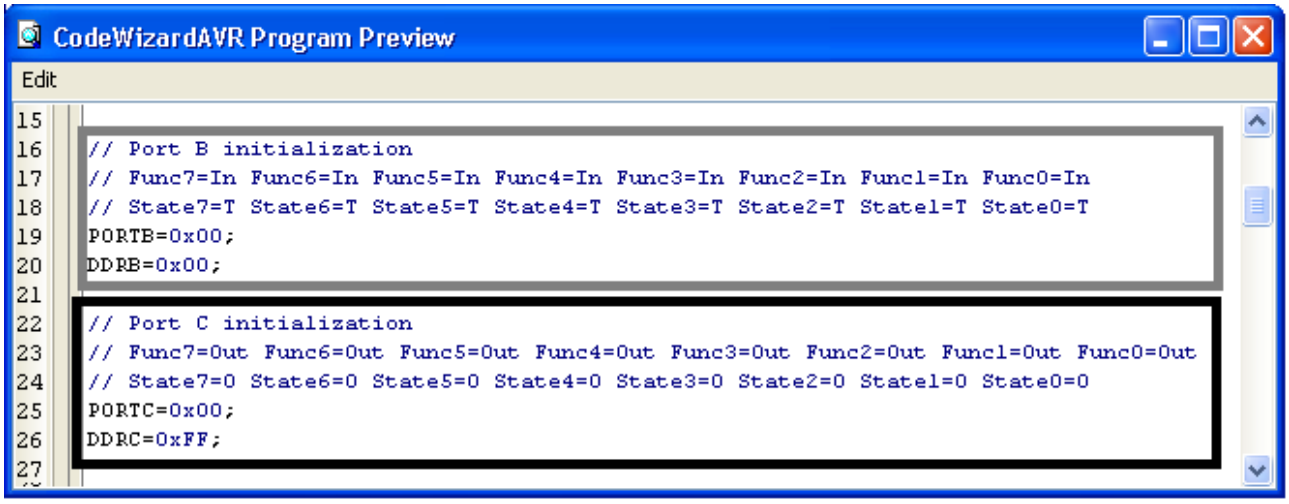


Illustration 23: Visualisation des différences d'un PORT en entrée et l'autre en sortie

Le cadre noir montre le PORT en sortie et le cadre gris montre le PORT en entrée. On peut voir que le DDR[5] du PORTC est à 1 sur tous ces bits car le code hexadécimal du DDRC est de 0xFF, tandis que ceux du PORTB sont à 0 son code hexadécimal est à 0x00.

5

4.2.La fonction principale

Notre fonction principale permettra de contrôler l'allumage des LED et de faire communiquer les 2 ATmega de notre projet. Pour ce qui est du contrôle des LED nous en parlerons dans la sous-partie suivante, ici nous allons expliquer la communication entre 2 ATmega.

Pour faire communiquer les 2 micro-contrôleurs, il existe 2 PINS qui permettent la transmission et la réception d'information. Mais nous avons opté pour un autre procédé, ou nous avons pris 2 PINS quelconques, une en entrée et une en sortie. Afin d'envoyer et recevoir des informations.

Lorsque tous les PINS d'un des micro-contrôleur seront à « 1 », nous enverrons un signal haut sur un PIN de l'autre ATmega qui comprendra ainsi qu'il peut lancer son programme. Comme cela, la deuxième moitié de l'horloge démarrera seulement lorsque la première moitié de l'horloge sera entièrement allumé. Et toutes les LED de l'horloge s'éteindront lorsque les LED de la deuxième partie de l'horloge seront allumées et que l'on enverra un signal haut sur un PIN du premier ATmega.

```
if(minute>30)
{
        PORTA.1=1;
        minute=0;
        heure++;

         if(heure>11)
    {
            heure=0;
    }
        A=B=C=D=0;


}

}
```

Illustration 25: Écriture sur un Bit de la première partie de l'horloge

```
while (1)
    {
     if(PINA.0==1)
     {
    if(minute<=6)
     {
             A=tab[minute+2] & 0xFC;


     }
     if(minute>6 && minute<=14)
     {
             C=tab2[minute-6];


     }
     if(minute>14 && minute<=22)
     {
             D=tab2[minute-14];

     }
     if(minute>22 && minute<=30)
     {
             B=tab2[minute-22];

     }
```

Illustration 24: Lecture sur un bit de la première partie de l'horloge

On écrit sur un PIN d'un PORT avec la fonction PORTA.1, qui écrit sur le PIN 1 du PORTA et on lit sur un PIN d'un PORT avec la fonction PINA.1 qui lit sur le PIN1 du PORTA.

Sur notre premier micro-contrôleur, le PIN 1 sert pour l'écriture et le PIN 0 sert pour la lecture et inversement pour l'autre ATmega. Ainsi le premier côté commencera à compter directement et enverra un signal lorsque l'on atteindra 30 minutes et attendra que l'autre micro-contrôleur ai compté 30 minutes lui aussi et renvoyé un signal pour remettre toutes les LED à 0 et incrémenter les heures, pour qu'une LED rouge se mette à clignoter afin de représenter les heures.

4.2.1.Le tableau

En programmation, un tableau est un ensemble de variables du même type stockée une seule fois que l'on peut récupérer quand on le veut, en les appelant. Pour notre programme, nous avions besoin d'un tableau qui contenait toutes les différentes valeurs que peut prendre le PORT d'un ATmega lorsque l'on alimente toutes les PINS à la suite des autres. De 0 alimentés à 8 alimentés.

```
const unsigned char tab[9]={0x00,0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
const unsigned char tab2[9]={0x00,0x80,0xC0,0xE0,0xF0,0xF8,0xFC,0xFE,0xFF};
```

Illustration 26: Définition des tableaux de variables

On voit donc que l'on allume une LED en plus à chaque fois avec ce programme. Car on met tous les PINS à 1 au fur et à mesure.

Mais pour appeler ce tableau on doit se rendre dans le « While » de la fonction principale et mettre la valeur « minute » dans le tableau qui s'incrémente elle dans la fonction interruption toutes les minutes. Ainsi chaque valeur de minute correspondra à une valeur du tableau.

Exemple : Lorsque « minute » vaut 1, tab[minute]=tab[1] vaut 0x01. Si l'on se réfère à la définition des tableaux de variables. La première valeur du tableau est tab[0].

```
if(minute<=6)
    {
        A=tab[minute];
    }
```

Illustration 27: Appel du tableau

On effectue ce procédé d'incrémentation pour tous les PINS de tous les PORTS du micro-contrôleur pour alimenter à la suite toutes les LED.

Pour notre projet, nous avions besoin d'une fonction qui s' exécute toutes les secondes afin d'incrémenter les secondes pour compter les minutes et les heures. Nous avons donc utilisés la fonction d'interruption fourni par le logiciel CodeVisionAVR.

4.3.La fonction d'interruption

La fonction d'interruption permet d'exécuter une action à une fréquence choisie dans le programme. Pour cela, on crée cette fonction d'interruption dans le programme.

décimal	15625 =	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
binaire		1	1	1	1	0	1	0	0	0	0	1	0	0	0
hexadécimal	3D08 =	3		D				0				8			

Illustration 28: Tableau de conversion Décimal/Binaire/Hexadécimal

Pour choisir la valeur de la vitesse de la fonction d'interruption, on prend la valeur de la fréquence de notre interruption qui est ici de 15 625 Hz, qui est une valeur donnée par le logiciel, puis on la converti en binaire et en hexadécimal. Nous souhaitons avoir une interruption toutes les secondes, pour représenter les secondes qui feront augmenter les minutes de notre programme. Si on rentre la valeur trouvée en hexadécimal dans le logiciel, alors l'interruption se fera toutes les secondes, car toutes les 64µs la fonction d'interruption envoie un échelon de tension si on choisi une fréquence de fonctionnement de 15 625 Hz. Donc lorsque la fonction obtiendra 15625 fois cette échelon, l'interruption se mettra en fonctionnement.

$$15625 \times 0{,}000064 = 1$$

4.3.1.Le masquage

Un masquage permet de faire des modifications d'une valeur ou d'une information mais de façon ciblée sans changer l'état des autres valeurs ou informations. Dans notre cas il agit donc sur un bit d'un octet sans changer la valeur des autres octets.

Pour cela on utilise des masques qui mettent à 1 avec la fonction OU « + » ou à 0 avec la fonction ET « & ».

Entrée		Sortie
a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Illustration 29: Table de vérité de la fonction ET

Entrée		Sortie
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Illustration 30: Table de vérité de la fonction OU

Nous avions besoin de ces fonctions dans notre programme pour faire clignoter les LED des heures sans pour autant interférer avec les LED qui contrôlent les minutes.

```
37
38  // Timer 1 output compare A interrupt service routine
39  interrupt [TIM1_COMPA] void timer1_compa_isr(void)
40  {
41     m=!m;
42
43
44
45              if(m==0)
46              {
47              A=(A&0b10111111);
48              }
49              else
50              {
51              A=(A|0b01000000);
52              }
53
54
```

Illustration 31: Fonction qui permet le clignotement d'une LED

Dans notre interruption nous avons définis une variable m qui prend alternativement la valeur 0 ou 1 à chaque fois que la fonction d'interruption se met en route. Ainsi avec ce procédé, toutes les secondes m change de valeur et lorsque m sera à 0, on mettra le bit 6 du port A à 0 avec la fonction ET agissant sur le PORTA et quand m sera à 1 ou mettra ce même bit à 1 avec la fonction OU agissant sur le PORTA.

On utilisera le même procédé pour chaque LED que l'on souhaite faire clignoter.

Nous n'avons pas eu le temps de programmer la carte électronique en CMS où nous devions mettre en œuvre la transmission de données entre les 2 micro-contrôleurs par les broches TXD et RXD[6]. Ce procédé implique de mettre dans notre programme une fonction qui gère la transmission et la réception d'informations.

6 TXD : PIN qui gère la transmission de donnée, on le relie à un RXD.
RXD : PIN qui gère la réception de donnée, on le relie à un TXD

Conclusion

Ce projet a donc été mené à bien dans sa quasi totalité et nous à été bénéfique car nous avons pu réaliser des cartes électroniques plus complexe que celles des précédents semestres et nous avons appris les bases de la programmation d'un micro-contrôleur avec le logiciel CodeVisionAVR.

Nous avons donc réussis à réaliser une carte électronique en 2 parties de 30 LED chacune avec le système qui les contrôle. Nous avons même conçus une carte électronique de la même horloge avec des composants CMS afin que cela prenne moins de place et que l'on puisse donc apprendre à souder ce type de composant.

Nous avons aussi réussi à programmer la première horloge en grande partie mais nous n'avons pas tout à fait fini, car seulement le côté gauche de l'horloge affiche le clignotement des heures. Le côté droit n'incrémente pas la variable des heures, donc le programme ne rentre pas dans les tests pour les heures et pour faire clignoter les LED. Nous essayerons donc de finir ce programme avant notre présentation orale. Nous avons pas eu le temps de tester le programme de l'horloge sur l'horloge en CMS, mais nous avons tout de même envoyé des petits programmes qui permettaient de savoir si il y avait bien transmission de données entre les ATmega et les LED. Et cela fonctionne. Il ne nous restera donc qu'à essayer de mettre en place le nouveau type de communication entre le TXD et le RXD des micro-contrôleurs.

Lors de ce projet, nous avons appris à travailler en autonomie avec des séances sans enseignant pour nous surveiller ou aider. Et nous avons appris à travailler en équipe réduite et à se répartir le travail car Jamal s'est plus investi dans la partie réalisation de la carte électronique tandis que Sonny lui s'est plus investi sur la programmation de l'horloge.

Résumé

Au cours du 4° semestre de notre DUT GEII, nous avions comme projet au cours des séances d'étude et réalisation, de mettre au point un système fonctionnant à partir d'un micro-contrôleur. Nous avons donc choisis de mettre en œuvre une horloge à LED piloté par Atmega8535.

Notre projet est d'allumer une LED pour chaque minute et de faire clignoter une autre LED pour représenter les heures. Pour cela nous allons créer un programme qui pilotera les ATmega et les fera communiquer et allumera les LED successivement.

Pour commencer nous allons concevoir une carte électronique qui aura la forme d'une horloge. En la découpant en 2 parties 30 LED pour une moitié qui seront contrôlées par un micro-contrôleur et l'autre moitié par l'autre ATmega. Mais nous ajouterons des transistors Darlington qui nous permettrons de pourvoir en courant l'alimentation de toutes les LED. Sans ces transistors Darlington le micro-contrôleur ne pourrait pas faire allumer toutes les LED en même temps. Nous communiquerons avec l'ordinateur par l'intermédiaire d'un connecteur ISP. Nous créerons aussi une carte électronique en CMS.

Nous élaborerons donc un programme qui permettra de piloter les ATmega et d'effectuer la communication entre ces 2 là. Nous utiliserons des tableaux pour stocker les valeurs que prendront les PORTS de l'ATmega8535 pour allumer les LED. Et nous créerons une fonction d'interruption qui incrémentera les secondes pour ensuite incrémenter les minutes et les heures et faire clignoter la LED des heures en fonction de sa valeur.

Index des illustrations

Bibliographie

TL1: Thierry LEQUEU, Documentation ATmega8535, http://www.thierry-lequeu.fr/data/ATMEGA-8535L.pdf
TL2: Thierry LEQUEU, RotaLED, http://www.thierry-lequeu.fr/data/ROTALED1.pdf
TL3: Thierry LEQUEU, Programmation afficheur LCD, http://www.thierry-lequeu.fr/data/DIV517.HTM#02 - Programme de test de l'afficheur LCD

Annexe 1

Programmation de la partie gauche de l'horloge

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 26/09/2012
Author  :
Company :
Comments:


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

#include <mega8535.h>

#include <delay.h>
#include <stdio.h>

// Alphanumeric LCD Module functions
```

```
const unsigned char tab[9]={0x00,0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
const unsigned char tab2[9]={0x00,0x80,0xC0,0xE0,0xF0,0xF8,0xFC,0xFE,0xFF};
unsigned char heure,A,B,C,D,m=0;
int seconde,minute;
bit a;

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
      if(PINA.0==1)
      {
      seconde++;
      if(seconde>=60)
      {
         minute++;
         seconde=0;
      }
      }
      m=!m;

      if(heure==7)
      {
            if(m==0)
            {
            A=(A&0b10111111);
            }
            else
            {
            A=(A|0b01000000);
            }
      }

      if(heure==8)
      {
```

```
        if(m==0)
        {
        C=(C&0b11101111);
        }
        else
        {
        C=(C|0b00010000);
        }
}

if(heure==9)
{
        if(m==0)
        {
        D=(D&0b01111111);
        }
        else
        {
        D=(D|0b10000000);
        }
}

if(heure==10)
{
        if(m==0)
        {
        D=(D&0b11111011);
        }
        else
        {
        D=(D|0b00000100);
        }
}

if(heure==11)
```

```
        {
            if(m==0)
            {
            B=(B&0b11011111);
            }
            else
            {
            B=(B|0b00100000);
            }
        }

        if(heure==0)
        {
            if(m==0)
            {
            B=(B&0b11111110);
            }
            else
            {
            B=(B|0b00000001);
            }
        }

        PORTA=A;
        PORTB=B;
        PORTC=C;
        PORTD=D;

}


// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
```

```
}


// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

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

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

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
```

```
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: CTC top=OCR0
// OC0 output: Disconnected
TCCR0=0x05;
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
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3d;
OCR1AL=0x08;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
```

```
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
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
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization


// Global enable interrupts
#asm("sei")
A=B=C=D=0;

while (1)
      {
      if(PINA.0==1)
```

```
        {
        if(minute<=6)
         {
               A=tab[minute+2] & 0xFC;
         }
         if(minute>6 && minute<=14)
         {
               C=tab2[minute-6];
         }
         if(minute>14 && minute<=22)
         {
               D=tab2[minute-14];
         }
         if(minute>22 && minute<=30)
         {
               B=tab2[minute-22];
         }
         if(minute>30)
         {
               PORTA.1=1;
               minute=0;
               heure++;

                if(heure>11)
                {
                      heure=0;
                 }
               A=B=C=D=0 ;
          }

         }
        }
}
```

Annexe 2

Programmation de la partie droite de l'horloge

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 26/09/2012
Author  :
Company :
Comments:


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

#include <mega8535.h>

#include <delay.h>
#include <stdio.h>

// Alphanumeric LCD Module functions
```

```
const unsigned char tab[9]={0x00,0x01,0x03,0x07,0x0F,0x1F,0x3F,0x7F,0xFF};
const unsigned char tab2[9]={0x00,0x80,0xC0,0xE0,0xF0,0xF8,0xFC,0xFE,0xFF};
unsigned char heure,A,B,C,D,m=0;
int seconde,minute;
bit a;

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
      seconde++;
      if(seconde>=60)
      {
         minute++;
         seconde=0;
      }
      m=!m;

      if(heure==1)
      {
            if(m==0)
            {
            A=(A&0b10111111);
            }
            else
            {
            A=(A|0b01000000);
            }
      }

      if(heure==2)
      {
            if(m==0)
            {
            C=(C&0b11101111);
            }
```

```
        else
        {
        C=(C|0b00010000);
        }
}

if(heure==3)
{
        if(m==0)
        {
        D=(D&0b01111111);
        }
        else
        {
        D=(D|0b10000000);
        }
}

if(heure==4)
{
        if(m==0)
        {
        D=(D&0b11111011);
        }
        else
        {
        D=(D|0b00000100);
        }
}

if(heure==5)
{
        if(m==0)
        {
        B=(B&0b11011111);
```

```
            }
            else
            {
            B=(B|0b00100000);
            }
        }

        if(heure==6)
        {
            if(m==0)
            {
            B=(B&0b11111110);
            }
            else
            {
            B=(B|0b00000001);
            }
        }

        PORTA=A;
        PORTB=B;
        PORTC=C;
        PORTD=D;



}


// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{

}
```

```
// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

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

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

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: CTC top=OCR0
// OC0 output: Disconnected
TCCR0=0x05;
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
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3D;
OCR1AL=0x08;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
```

```
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
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
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization


// Global enable interrupts
#asm("sei")
A=B=C=D=0;
while (1)
      {
       if(minute<=6)
       {
             A=tab[minute+2] & 0xFC;
       }
```

```
        if(minute>6 && minute<=14)
        {
              C=tab2[minute-6];
        }
        if(minute>14 && minute<=22)
        {
              D=tab2[minute-14];
        }
        if(minute>22 && minute<=30)
        {
              B=tab2[minute-22];
        }
        if(minute>30)
        {
              PORTA.0=1;
               if(heure>6)
               {
                     heure=0;
               }
        }
              if(PINA.1==1)
              {
               A=B=C=D=0;
              minute=0;
               heure++;
              }
      }
}
```

Annexe 3

Tableau des positions des Leds

Angle	Pos x	Pos y			rayon
3	4993	262	**D2**	1	**5000**
9	4938	782	**D3**	2	
15	4830	1294	**D4**	3	
21	4668	1792	**D5**	4	
27	4455	2270	**D6**	5	
33	4193	2723	**D7**	6	
39	3886	3147	**D8**	7	
45	3536	3536	**D9**	8	
51	3147	3886	**D26**	9	
57	2723	4193	**D27**	10	
63	2270	4455	**D28**	11	
69	1792	4668	**D29**	12	
75	1294	4830	**D30**	13	
81	782	4938	**D31**	14	
87	262	4993	**D32**	15	
93	-262	4993	**D33**	16	
99	-782	4938	**D18**	17	
105	-1294	4830	**D19**	18	
111	-1792	4668	**D20**	19	
117	-2270	4455	**D21**	20	
123	-2723	4193	**D22**	21	
129	-3147	3886	**D23**	22	
135	-3536	3536	**D24**	23	
141	-3886	3147	**D25**	24	
147	-4193	2723	**D12**	25	
153	-4455	2270	**D13**	26	
159	-4668	1792	**D14**	27	
165	-4830	1294	**D15**	28	
171	-4938	782	**D16**	29	
177	-4993	262	**D17**	30	
183	-4993	-262	**D34**	31	
189	-4938	-782	**D35**	32	
195	-4830	-1294	**D36**	33	
201	-4668	-1792	**D37**	34	
207	-4455	-2270	**D38**	35	
213	-4193	-2723	**D39**	36	
219	-3886	-3147	**D40**	37	
225	-3536	-3536	**D41**	38	
231	-3147	-3886	**D56**	39	
237	-2723	-4193	**D57**	40	
243	-2270	-4455	**D58**	41	
249	-1792	-4668	**D59**	42	
255	-1294	-4830	**D60**	43	
261	-782	-4938	**D61**	44	
267	-262	-4993	**D62**	45	
273	262	-4993	**D63**	46	
279	782	-4938	**D48**	47	
285	1294	-4830	**D49**	48	
291	1792	-4668	**D50**	49	
297	2270	-4455	**D51**	50	
303	2723	-4193	**D52**	51	
309	3147	-3886	**D53**	52	
315	3536	-3536	**D54**	53	
321	3886	-3147	**D55**	54	
327	4193	-2723	**D42**	55	
333	4455	-2270	**D43**	56	
339	4668	-1792	**D44**	57	
345	4830	-1294	**D45**	58	
351	4938	-782	**D46**	59	
357	4993	-262	**D47**	60	

Annexe 4

Nomenclature

Repère	Composants	Valeurs/types	Quantité	Coût unitaire
R2-R32	Résistance	270Ω	30	0,19€
R1	Résistance	1,5kΩ	1	0,19€
C1	Condensateur	100uF 63V	1	0,302 €
C2	Condensateur	470uF 6,3V	1	0,98€
C3,C5	Condensateur	100nF	2	0,36 €
C4	Condensateur	10µF 6,3V	1	0,232€
C6,C7	Condensateur	22pF	2	0,314 €
L1	Inductance	47µH	1	0,80€
D1	Diode	1N4007	1	0,10€
D10	Diode	1N5819	1	0,20 €
D11	Led	2mA	1	0,168 €
Q1	Quartz	16MHz	1	0,57 €
U1	Régulateur	LM2575-5	1	1,95€
U2,3,5,6		ULN2803	1	1,598 €
D2-D34	Led	3mm	30	0,168 €
JP1	Connecteur 2 fil	ALIM	1	0,15€
JP2	Connecteur ATmega8535	CON ISP	1	
U4	µ-contrôleur	ATmega8535	1	3,96 €
			Coût final	12,232€

Dans la nomenclature ci-dessus on retrouve les composants et les prix pour la moitié de la carte, il faut donc multiplier le coût final par 2 pour obtenir le coût réel final.