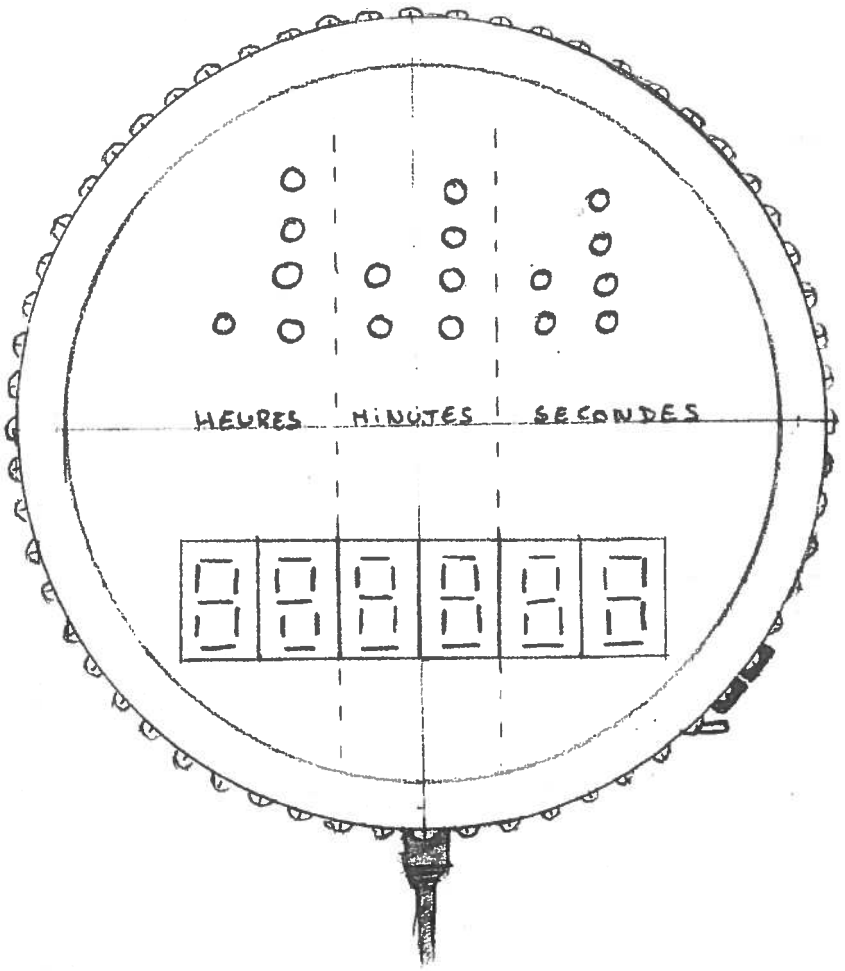
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

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



Rapport de projet tutoré 2ère année

Horloge à LED

Commandée par protocole I2C avec un RTC

TUILARD Benjamin
Groupe P1
Promotion 2010/2012

Professeurs référents:
- *RODIER Sofi*
- *BILLOUÉ Jérôme*

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

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

Rapport de projet tutoré 2ère année

Horloge à LED

Commandée par protocole I2C avec un RTC

TUILARD Benjamin
Groupe P1
Promotion 2010/2012

Professeurs référents:
- *RODIER Sofi*
- *BILLOUÉ Jérôme*

Table des matières

Remerciements

Je souhaite tout d'abord, avant de commencer mon rapport de projet tutoré de 2e année, à adresser des remerciements à certaines personnes qui m'ont fortement aidé durant ce projet.

Dans un premier temps je remercie l'IUT de nous permettre de pouvoir travailler sur un projet, aussi personnel qu'il peut être, dans le cadre de notre formation. Celui-ci nous permet ainsi de mettre en œuvre de manière concrète notre réflexion et nos connaissances sur un projet précis pour en trouver la solution.

Je tiens à remercier ensuite Mr BILLOUE Jérôme qui m'a beaucoup orienté sur les propositions de solutions.

Je remercie également Mr BRUNO Armel et Mr LEQUEU Thierry qui m'ont aidé à comprendre le bus I2C ce qui m'a permis de pouvoir programmer plus facilement.

Pour finir je remercie Mr VAUTIER Richard qui m'a toujours accueilli dans le magasin. Il m'a grandement conseillé durant ce projet.

Introduction

Dans le cadre du semestre 4 en cours d'études et réalisation, j'ai pu réaliser un projet personnel.

Ce projet consiste à fabriquer une horloge à LED[1]. C'est une réalisation qui rôdait dans mes pensées depuis un moment en collaboration avec une amie et je profite de pouvoir concrétiser ce projet dans le cadre de mes études.

Dans ce rapport j'expliquerai tout d'abord en quoi va consister l'horloge à LED. Puis j'exposerai les différentes recherches que j'ai effectué, les problèmes rencontrés ainsi que les solutions envisagées et retenues.

J'approfondirai sur les solutions retenues, comment celles-ci fonctionnent et la manière dont je les ai réalisée.

Pour finir je conclurai en faisant un bilan de mon expérience et de mon projet, mon ressentiment durant sa réalisation et ce que cela m'a apporté.

1 LED: Ligth Emmiting Diode ou Diode ÉlectroLuminescente (DEL)

1. Présentation du sujet

Comme dit dans l'introduction, le but de mon projet est de créer une horloge à LED. Celle-ci doit afficher l'heure de 3 manières différentes:

- Traditionnelle : 60 LED feront le tour de l'horloge. (cf illustration 2).
- Binaire[2] : 24 LED soit pour les heures, minutes et secondes un groupement de plusieurs bits[3] qui vont représenter les dizaines et unités (cf illustration 1). L'heure sera donc en langage DCB (cf Annexe, page 26).
- Afficheur : l'heure sera affiché par des afficheurs 7-segments tel un réveil numérique.

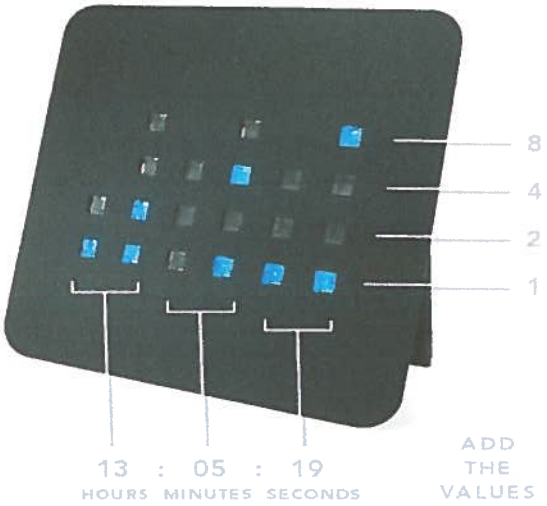


Illustration 1: Horloge binaire de 24 LED[0]

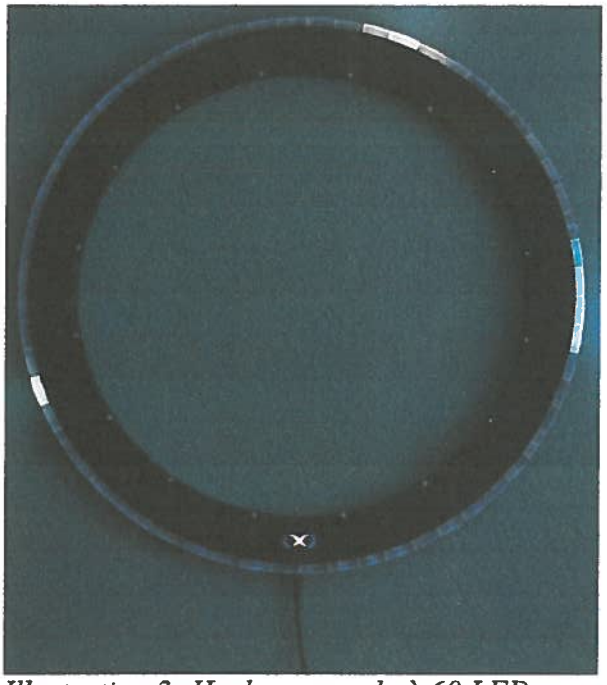


Illustration 2: Horloge murale à 60 LED RVB[0]

2 Langage Binaire : langage informatique composé exclusivement de 0 et de 1. Dans notre cas il sert à convertir des nombres décimales en nombres binaires (cf Annexe, page 25)

3 Bit: chiffre binaire pouvant valoir 0 ou 1. Il décrit ainsi l'état d'une variable, si elle est active ou non. C'est alors une unité de mesure informatique.

Ces 3 affichages seront placés dans un horloge vide avec une façade en plexiglas permettant de laisser toute l'électronique apparente ce qui est un effet désiré. Cette horloge ronde a un diamètre de 30 cm.

Elle devra fonctionner sous une tension de 5V branchée et une fois l'alimentation retirée, elle gardera l'heure en mémoire malgré que les affichages ne fonctionnent plus. Une fois les affichages allumés, l'horloge indiquera de nouveau l'heure comme si il n'y avait jamais eu coupure d'alimentation. L'heure devrait aussi être réglable.

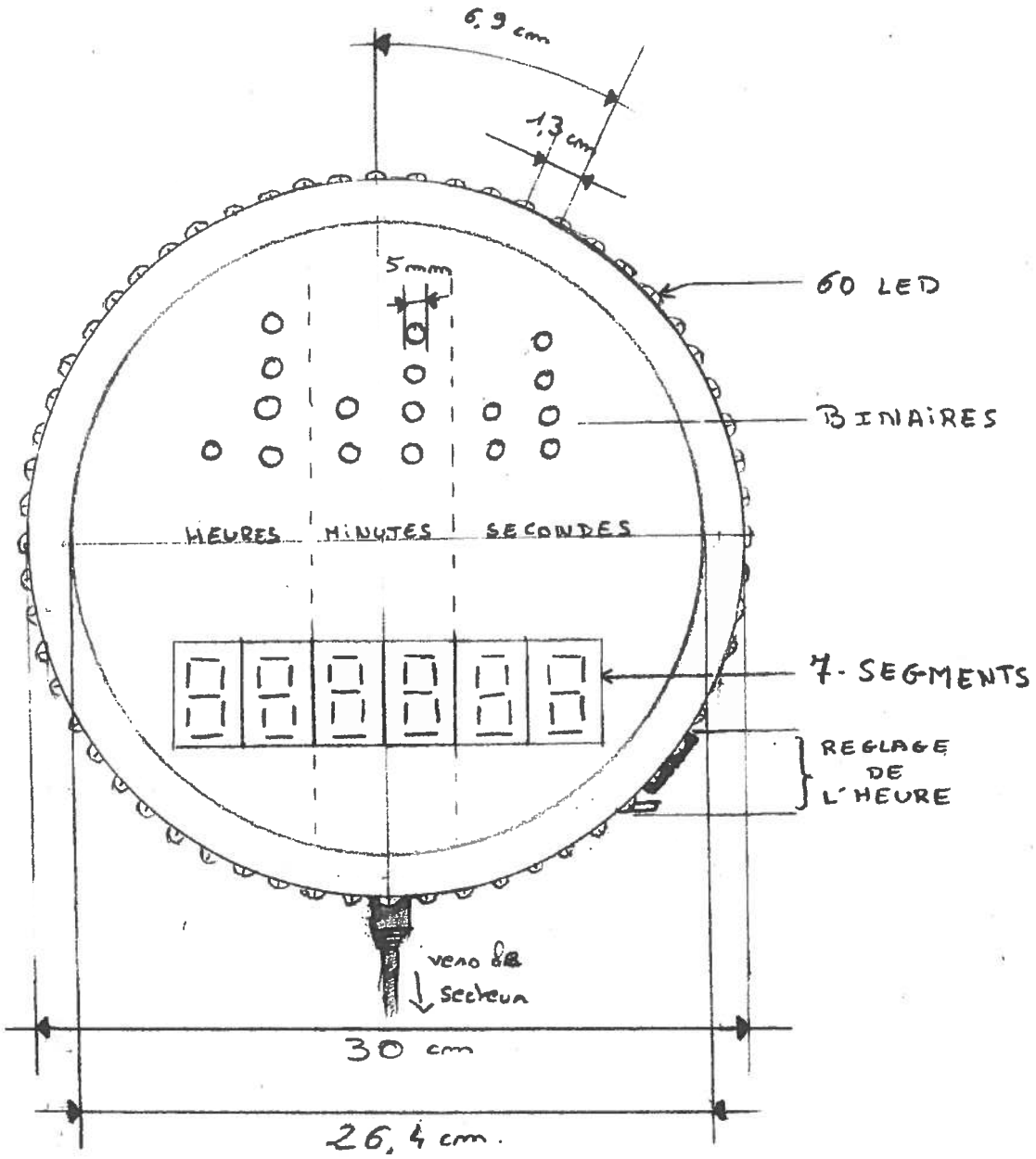


Illustration 3: Schéma d'illustration de l'horloge finale. Croquis fait par l'acteur

2. Analyse du sujet

2.1. Les recherches préliminaires

Avant de pouvoir commencer la réalisation du projet, j'ai recherché les travaux déjà existants sur les différentes horloges, qu'elles soient binaires ou autres.

J'ai alors orienté mes premières recherches sur l'horloge binaire. J'y ai pu voir des solutions simples à réaliser. Tout d'abord, des montages à base de NE555, générant un signal d'horloge[4] de période d'une seconde. Cette horloge est alors reliée à des compteurs mis en cascade qui vont afficher, grâce à des lampes, l'heure en binaire (cf illustration 4).

Ces compteurs aurait pu très bien fonctionner dans mon projet. De plus la récupération de la valeur des compteurs aurait pu servir à l'afficheur 7-segment par le relais de décodeur.

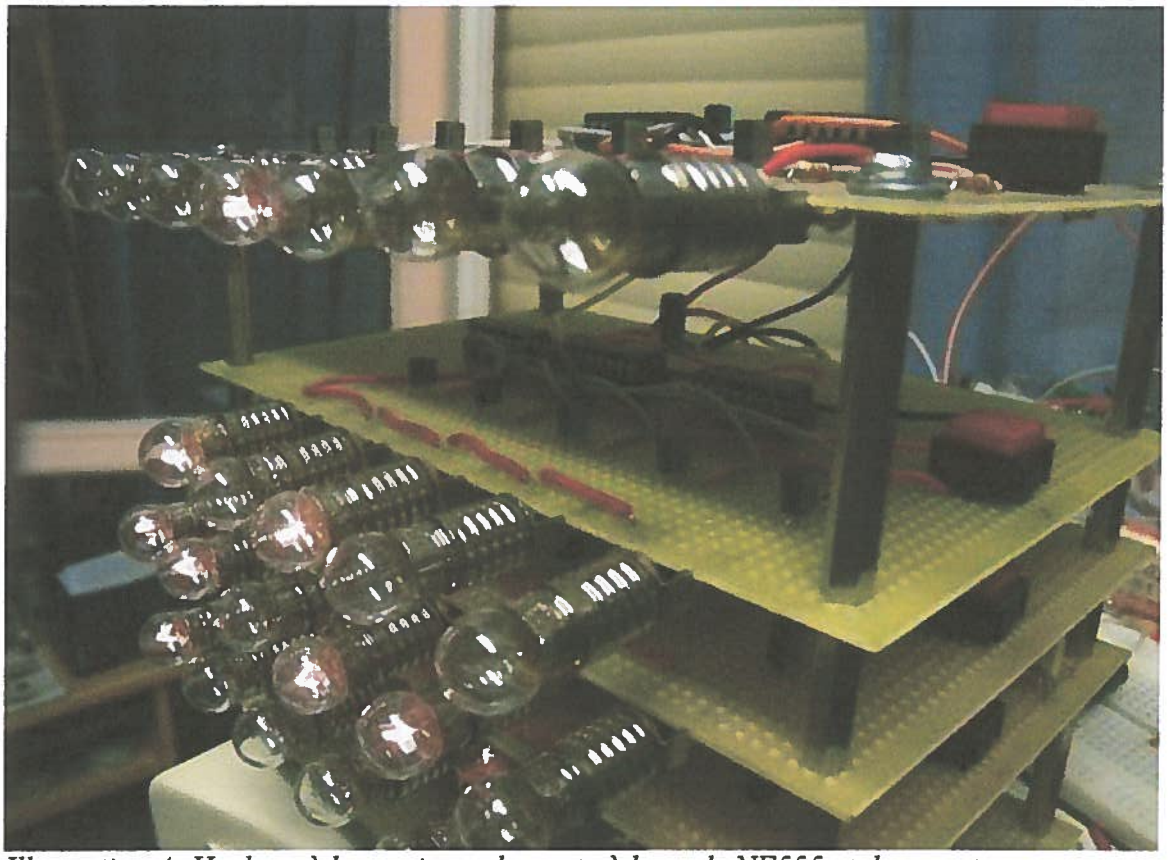


Illustration 4: Horloge à lampe incandescente à base de NE555 et de compteurs CD4024[0]

4 Signal d'horloge: signal électrique en créneaux, c'est-à-dire variant de façon périodique entre 2 états généralement dit état haut et état bas, correspondant à une activation ou non d'un indicateur tel qu'une lampe par exemple.

Ensuite il vient l'horloge à 60 LED. Après des recherches peu fructueuses sur le fonctionnement d'une telle horloge, j'ai vu que cette dernière fonctionne avec des registres à décalages. Chaque LED s'allume les unes après les autres au rythme des secondes, minutes et/ou heures.

Illustration 5: Horloge à 60 LED pouvant fonctionner avec des registres à décalages

Mais toutes ses solutions ne répondent pas au cahier des charges pour la simple et bonne raison qu'une fois les montages plus alimentés, les compteurs ou les registres se remettent à zéro. Ce qui oblige à régler l'heure à chaque coupure d'alimentation. On peut aussi rajouter que le NE555 n'est pas parfait. Même s'il permet de fournir un signal d'horloge de fréquence précise, celle-ci peut au fur et à mesure se décaler rendant l'heure affichée complètement erronée au fils du temps.

De plus l'horloge à 60 LED avec les registres à décalages ne donne pas l'animation souhaitée avec les LED. Je souhaite qu'une simple LED s'allume au rythme des minutes et des heures (cf illustration 2).

C'est alors que je me suis orienté sur une autre solution complètement différente.

2.2. Les solutions envisagées et problèmes rencontrés

Le plus gros problème de ce projet est de garder en mémoire l'heure en cas de rupture d'alimentation et au mieux de pouvoir reprendre l'affichage de l'heure réelle dès qu'on le souhaite.

L'idée est d'avoir un module qui génère une horloge, après un réglage préliminaire, et qui continue de la générer grâce à une alimentation interne à ce module. Celui-ci sera en quelque sorte indépendant à l'alimentation générale de l'horloge et pourra ainsi continuer de compter l'heure malgré tout les affichages éteints.

2.2.1. Le RTC DS1307

Après plusieurs recherches et grâce aux conseils de Mr BILLOUÉ, j'ai trouvé un composant répondant à mes attentes. Il s'agit du DS1307.

C'est un RTC[5], c'est-à-dire une horloge à temps réelle qui va fournit les secondes, minutes et heures, mais aussi les jours, mois et heures grâce à un cristal de quartz. Une pile externe lui permet de toujours être fonctionnelle même en cas de coupure de l'alimentation

L'avantage d'un tel composant est qu'il ne se décale pas dans le temps. Son système reste parfaitement précis. De plus il tient compte des années bissextile jusqu'en 2100.

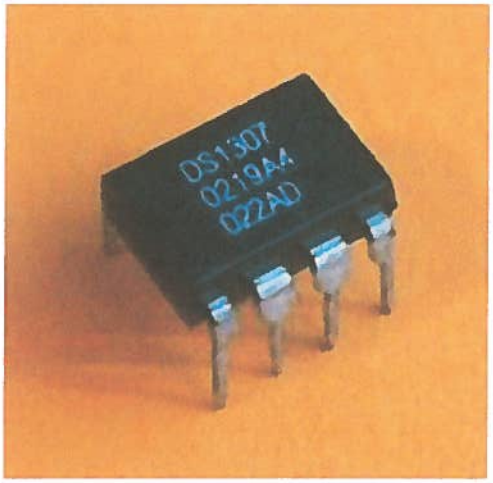


Illustration 7: RTC DS1307[0]

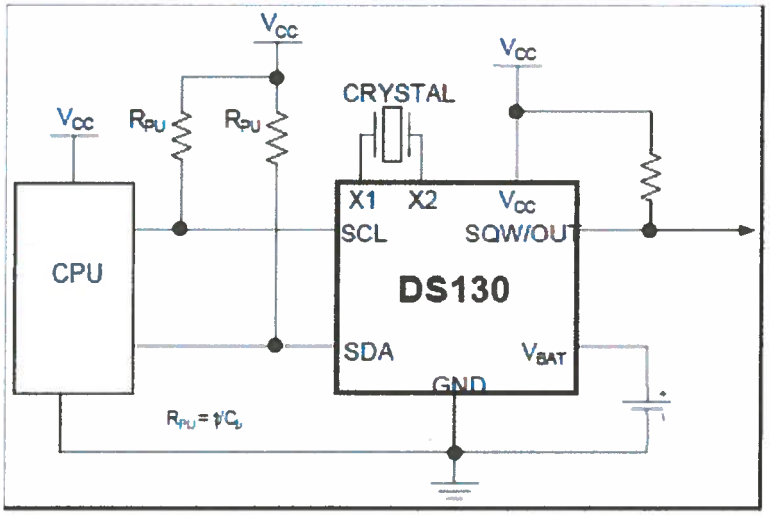


Illustration 6: Schéma de principe du RTC DS1307

Ce composant est piloté par un microcontrôleur. Ce dernier peut lire et écrire sur le RTC via un bus I2C[6](cf partie 3.1, page 15).

5 RTC : Real-Time Clock
6 Bus I2C : Inter Integrated Circuit

2.2.2. L'ATmega8535

Parmi les microcontrôleurs compatibles en bus I2C, j'ai choisi l'ATmega8535. J'ai pu l'étudier durant le module complémentaire « Microprocesseur » avec Mr BESSE Dominique ainsi que durant mon projet du 3e semestre sur la gestion de l'éclairage du kart électrique de l'IUT piloté par un ATmega8535.

L'ATmega8535 est un composant électronique programmable possédant 32 broches d'entrées/sorties sur 4 ports différents (PA, PB, PC et PD) ainsi qu'une horloge interne de 8 MHz. Il possède diverses fonctions tel qu'un comparateur ou un convertisseur. Dans notre cas présent, la fonction qui nous intéresse est le protocole I2C qui se fera sur les broches 22 et 23 qui sont respectivement les bits PC0 (SCL) et PC1 (SDA).

Il sera programmé avec le logiciel « Code Wizard AVR ». Il pilote aussi tous les affichages . C'est alors le cœur de mon projet.

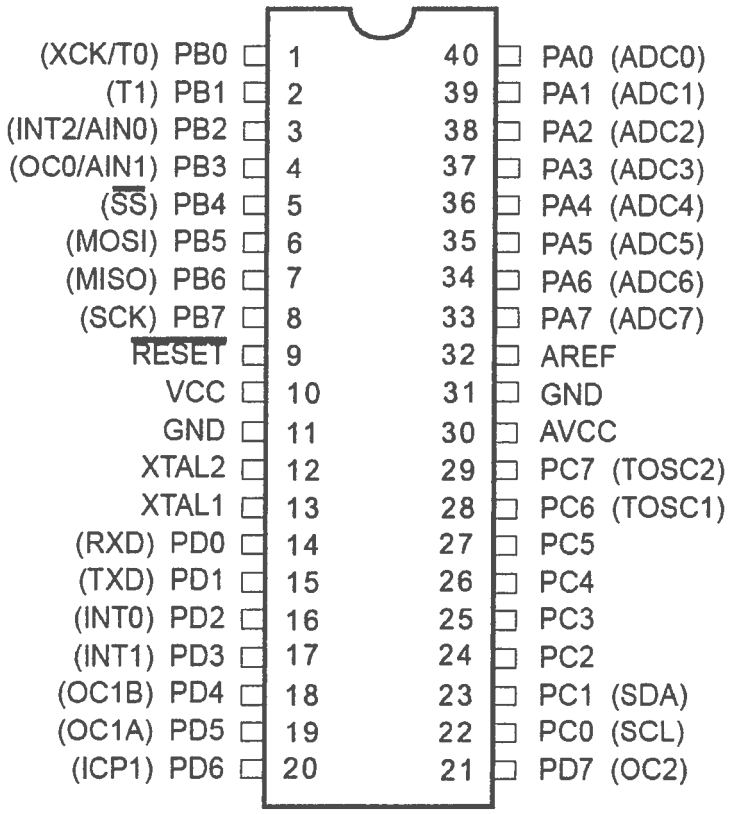


Illustration 8: Schéma de l'ATmega8535 issu de sa datasheet [1]

2.2.3. La CPLD[7]

La commande des 60 LED autour de mon horloge est assez conséquente. Ainsi j'en ai pu en déduire que la mise en cascades de plusieurs registres à décalages est une solution coûteuse et imposante d'un point de vue de la taille.

C'est pourquoi, sur le conseil de Mr BRUNO, je me suis orienté sur une CPLD. Cette dernière désigne un type de circuit complexe programmable. Elle va jouer le rôle des registres à décalages. Il suffit alors de programmer la CPLD en langage Verilog grâce au programme « Quartus ».

Pour afficher l'heure autour de l'horloge, nous avons besoin de 60 LED représentant les minutes et les heures voir même éventuellement les seconds. De plus il faut tenir compte du fait qu'il faut envoyer les données à la CPLD ainsi qu'un signal d'horloge pour qu'elle affiche l'heure.

La CPLD n'est pas compatible en bus I2C. C'est pour cela que l'on doit réservé certaines broches (ou « pin ») de la CPLD pour permettre l'envoi de données entre le microcontrôleur et la CPLD. Chaque broche va représenter un bit.

La valeur maximum a envoyé à la CPLD est 59 (on commence à compter à partir de 0) correspondant au 60 minutes. En binaire c'est une plage de données variant de 0000 0000 (pour 0) à 0011 1011 (pour 59). Cela représente un nombre maximum de bits utilisés de 6. Il faut donc réserver 6 broches de la CPLD.

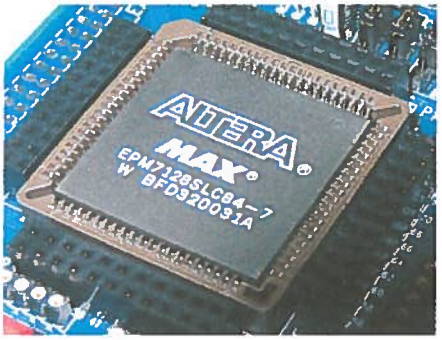


Illustration 9: CPLD EMP7128SLC84 possédant broches[2]

Au total nous avons donc 6 broches pour l'échange de données, une broche pour le signal d'horloge ainsi que 60 broches pour commander toutes les LED pour les minutes et les heures. Cela fait un total de66 broches minimum.

J'ai donc tout naturellement choisi un EPM7128SQC100-15N qui est une CPLD possédant 84 broches d'entrées/sorties et fonctionnant sous une tension de 5V, ce qui est parfait car l'ATmega8535 et le RTC fonctionne avec cette même tension.

---

7 CPLD : Complex Programmable Logic Device

2.2.4. L'afficheur binaire et 7-segments

Il ne restait plus qu'à s'occuper de l'afficheur binaire et 7-segments. Lors des premières recherches, une solution s'est présentée rapidement : un driver 7-segments SAA1064. C'est un composant qui peut communiquer en bus I2C. Il suffit juste d'envoyer la valeur de l'heure et le composant pilote de 2 à 4 afficheurs 7-segments sans avoir à le programmer et de manière automatique.

L'inconvénient est que le composant coûte très cher et qu'il y en faudrait environ 3 pour réaliser l'affichage de l'heure intégralement ce qui représente en budget minimum de 150€.

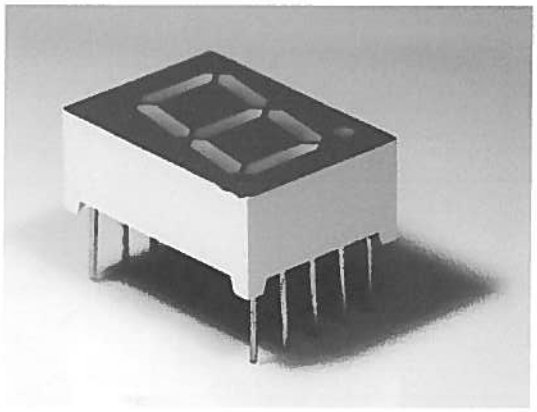


Illustration 10: Afficheur simple 7-segments

J'ai alors mis de côté cette solution. Un autre composant un peu plus complexe que j'ai trouvé en parallèle au précédent va servir d'alternative à ce problème. C'est un MCP23017.

Ce composant communique en bus I2C. C'est un peu plus complexe à programmer pour communiquer avec lui mais il permet de piloter les LED qui serviront à l'horloge binaire.

Le MCP23017 fonctionne également en 5Volts. De plus la récupération des sorties du composants vont être sur 4bits. Ces derniers seront envoyé, en plus des LED, sur des décodeurs SN74LS47. Ce décodeur va être converti la valeur sur 4 bits pour l'afficher sur un afficheur 7-segment (cf Illustration 10).

Alors que tous les composants trouvés et retenus répondent à l'ensemble du cahier des charges, je peux maintenant établir un planning des travaux à réaliser et me concentrer sur le partie la plus complexe et importante du projet : la programmation.

2.3. Le planning

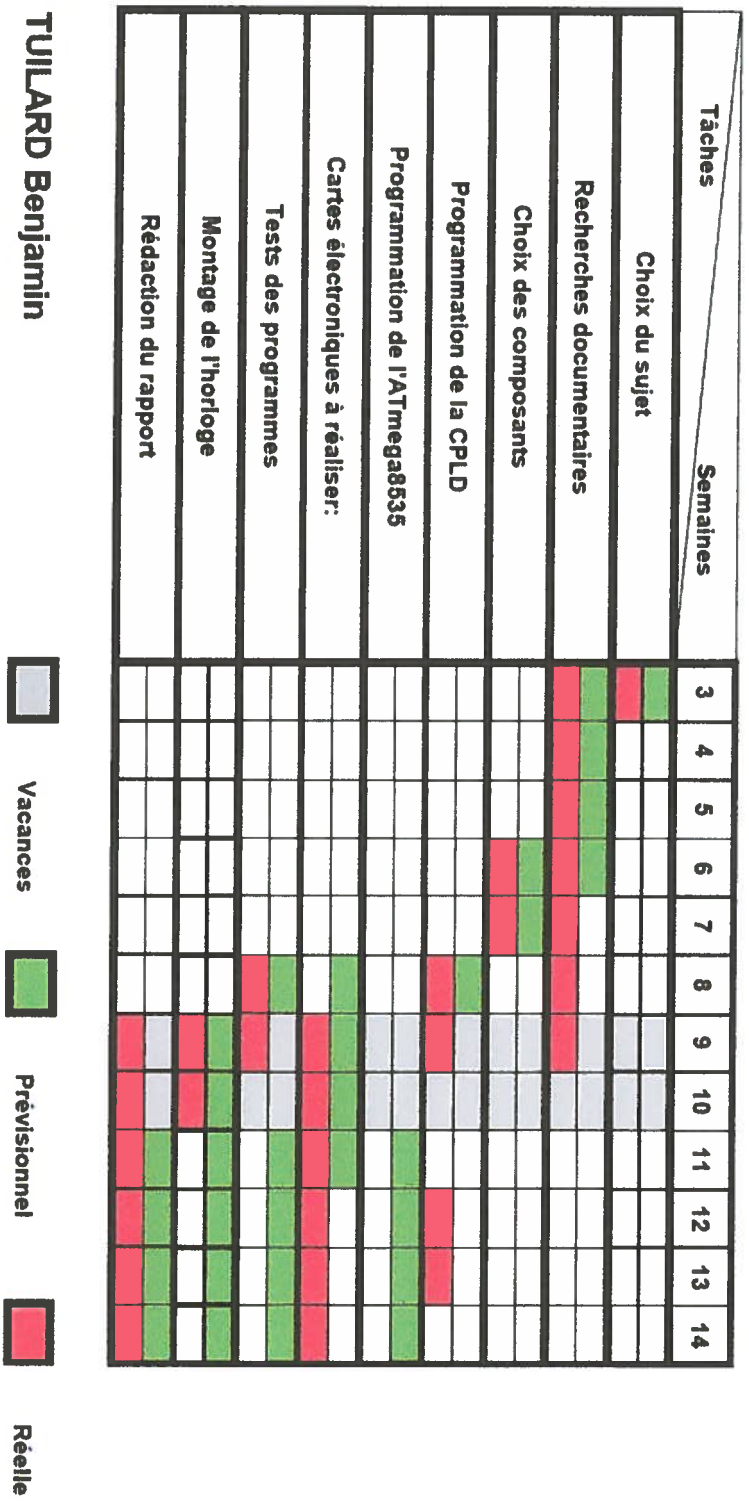


Illustration 11: Planning du projet, fait par les auteurs

3. La programmation

3.1. Le Bus I2C

Le bus[8] I2C a été inventé pour simplifier la communication entre des composants divers. Il fut développé dans les années 1980 par Philips.

Chez certains constructeurs, ce bus est parfois nommé sous le nom de TWI (Two Wire Interface).

Ce bus n'utilise que 3 fils :

- un signal de données (SDA[9])
- un signal d'horloge (SCL[10])
- un signal de référence électrique (masse)

Illustration 12: Logo du bus I2C

Pour les composants voulant communiquer en I2C il suffit de lier ces 3 fils. Le fonctionnement du bus I2C est assez simple malgré que son protocole peut sembler complexe.

La communication I2C repose sur l'échange entre un maître et des esclaves par adressage. C'est-à-dire que chaque composant dit esclave possède une adresse spécifique sur 7bits. C'est le composant maître qui gère la communication. Il envoie les signaux de données et le signal d'horloge. Généralement le maître est un composant dit microcontrôleur.

Le rôle du signal d'horloge SCL est de rythmer l'envoi des données. Chaque bit du signal de données SDA sera envoyé à chaque changement d'état du SCL.

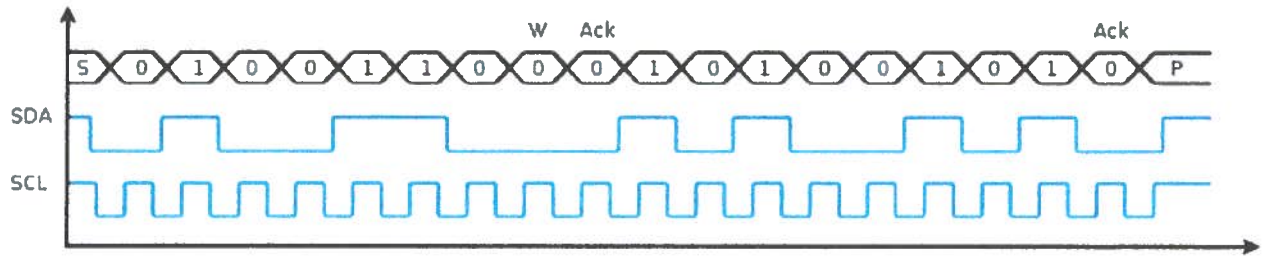


Illustration 13: Exemple de communication I2C[3]

8 Bus : un bus informatique est un système de communication entre des composants. Il désigne l'ensemble des supports de communication (câble, fibre optique...) mais aussi le logiciel et protocole associé (ici le protocole I2C)
9 SDA : Signal DAta
10 SCL : Signal CLock

Au début d'une communication, il faut s'assurer que le SDA et SCL sont en état de repos, soit un niveau logique 1.

Ensuite le maître envoie sur le bus I2C, l'adresse de l'esclave avec qui il veut communiquer. On dit qu'il envoie les bits d'adressage. Suite à cela le maître indique le mode d'action (R/W) c'est-à-dire le maître indique s'il veut écrire ou lire les données de l'esclave.

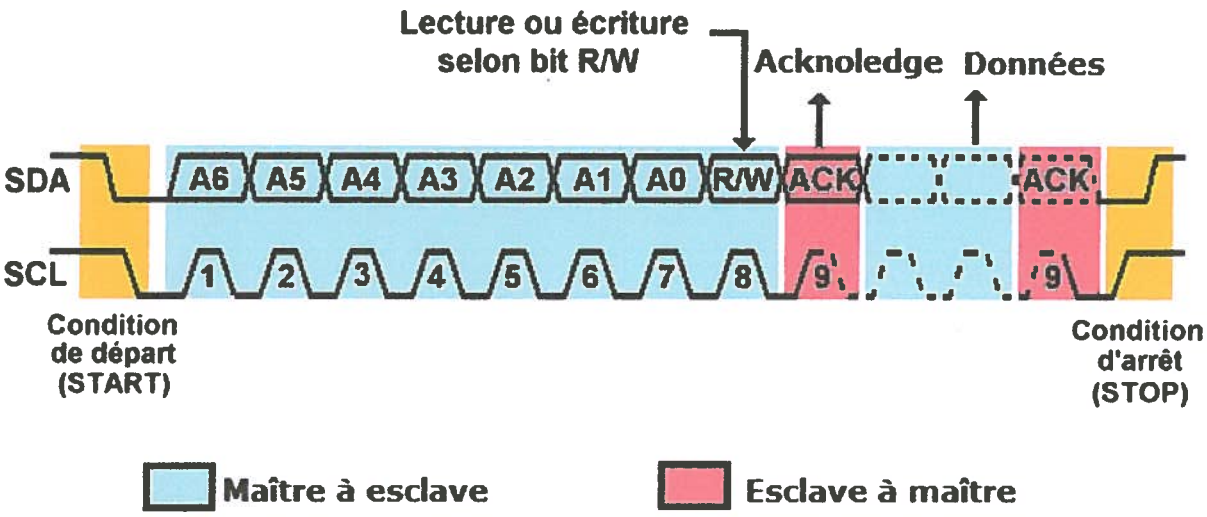


Illustration 14: Schéma de principe d'une trame de communication entre maître et esclave en bus I2C [3]

L'esclave possédant l'adresse indiqué répond par un « Acknowledge » (ACK). C'est un bit dit d'acquittement pour indiquer qu'il a bien reçu les instructions du maître. Ce bit est symbolisé par un forçage de l'esclave à mettre la ligne SDA à un niveau logique 0.

L'acquittement reçu par le maître, celui-ci envoyé alors les données à l'esclave qui acquittera à nouveau pour confirmer la réception. Le maître peut alors envoyé autant de donnée qu'il veut sur un même esclave qui acquittera à chaque opération.

Pour finir, un maître peut communiquer avec plusieurs esclaves à condition de paramétrer les adresses de tous les esclaves. Il devra alors indiquer, à chaque changement d'esclave, l'adresse du nouveau esclave avec qui il veut communiquer.

3.2. Analyse des programmes

3.2.1. La CPLD

La CPLD n'est pas un composant qui peut communiquer en bus I2C. Il faut donc envoyer les données manuellement. De plus l'heure sera transmise en langage binaire. Sachant que ces données sont les heures et les minutes, elles vont variés de 0 à 59.

```
module horloge (clock, data, affichage, mode);
```
Déclaration du nom du programme avec toutes les variables utilisées

Le nombres de bits nécessaire dépend de la valeur maximum à transmettre, ici 59. On sait déjà qu'il nous faut 6 bits pour l'envoie des données , ce qui correspond à transmettre un nombre maximum égale à 2^6-1 soit 63.

C'est pourquoi j'ai choisi de réserver 6 ports de la CPLD pour l'envoi de données de l'ATmega8535. Il correspondra à une variable DATA du programme de la CPLD qui sera alors tout simplement un nombre entier codé sur 6 bits.

```
input clock ;
input [5:0] ;
input mode;
```
Déclaration des entrées ([5:0] nombre de bit soit ici 6)

Le principe de transmission est simple. Tout d'abord on doit synchronisé l'envoi des données. On va déclarer alors une variable CLOCK dans le programme. L'ATmega8535 enverra alors sur une broche de la CPLD un signal horloge qui va rythmer l'envoi des données. Dans le programme, nous détecterons alors un front montant[11] de l'horloge, il sera l'évènement déclencheur.

```
always@ ( posedge clock)
begin
        [Traitement]...
end
```
Détection de chaque front montant de clock, on applique le traitement

Ensuite un bit correspondant à une variable « MODE » sera transmis entre l'ATmega8535 et la CPLD. Ce bit a pour rôle de savoir si le nombre transmis à la CPLD indique les minutes, les heures ou les secondes. Ainsi si MODE est à 0, le nombre transmis à travers DATA sera les minutes, si MODE est à 1 on envoie les heures, et si MODE est à 2 DATA recevra les secondes.

11 Front montant: passage du niveau bas à un niveau haut d'un signal.

Le rôle de la CPLD est alors de piloter 60 LED pour les minutes, pour les heures et à la fois pour les secondes. On va donc déclarer une variable qui sera un nombre de 60 bits.

```
output [59:0] affichage;
```
Déclaration de la sortie

En langage binaire, chaque bit est symbolisé par un rang. Par exemple un nombre de 3 bits est composé d'un bit de rang 0, un bit de rang 1 et un bit de rang 2 soit au total de 3 rangs.

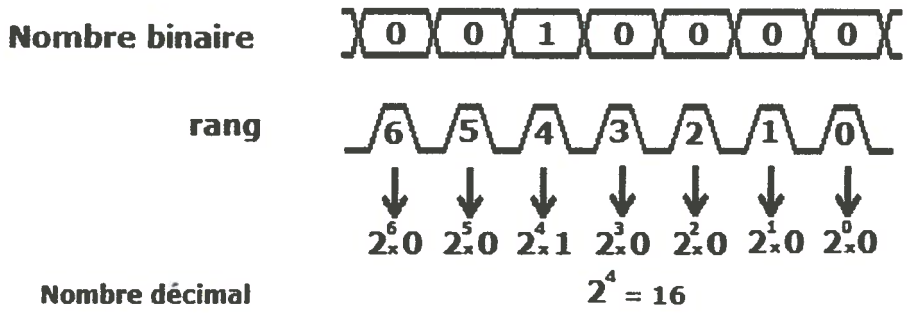


Illustration 15: Illustration d'un nombre binaire de 7 bit, faite par l'auteur

Sur l'horloge, je souhaite qu'une seule LED soit allumée et qu'elle corresponde à la minute ou l'heure désirée. Pour cela je vais exploiter la composition d'un nombre binaire. Par exemple pour les minutes, la variable affichage est composé de 60bits, soit 60 rangs variant de 0 à 59. Pour activer une LED il suffit de mettre un 1 logique sur le bit au rang correspondant.

Si l'on prend en exemple 6minutes, il faut allumer la 6e LED, soit mettre un 1 logique sur le bit de rang 6. C'est pour cela que l'on va écrire sur le variable la valeur 2^6, qui correspond à mettre uniquement un 1 logique sur le bit de rang 6 et de laisser tous les zéros à 0 (cf illustration 15).

Donc pour n'importe quelle valeur des minutes, il suffira d'assimiler le rang d'un bit à la valeur minute et d'écrire sur la variable la valeur 2^{minute}. Cette opération est la même que pour les heures mais sur 12 bits. C'est deux opérations se font alors à chaque front montant de l'horloge et suivant la valeur du bit MODE. On allumera ainsi seulement les LED que l'on souhaite pour obtenir l'heure et l'affichage désirée.

```
if (mode == 0)
        minute = 2**data;
if (mode ==1)
        heure = 2**(data*5);
if (mode ==2)
        seconde = 2**data;
```
On regarde la valeur de MODE et on affecte à la variable la valeur 2^{DATA}

3.2.2. L'ATmega8535

C'est la partie de programmation la plus complexe du projet. Malheureusement par le manque de temps je n'ai pas pu aboutir sur ce point.

La programmation aurait été faite grâce au logiciel « Code Wizard AVR ». Celui-ci aurait permit de fournir directement des fonctions liées au bus I2C, rendant la programmation plus facile et plus rapide.

4. La conception matérielle

4.1. Les cartes électroniques

Les cartes électroniques ont été faite grâce au logiciel ORCAD. Le schéma a été fini mais les typons non réalisé donc par conséquent les cartes électroniques non plus. Sur le schéma électrique j'ai dû penser à assigner toutes les broches pour le bon fonctionnement de l'horloge mais aussi prévoir les prises J-TAG pour pouvoir programmer l'ATmega8535 et la CPLD.

4.2. L'horloge

Le support de l'horloge est une ancien horloge personnel dont j'ai enlevé tout le mécanisme des aiguilles.

Le fond de l'horloge étant démontable, il facilite l'accès à l'intérieur de l'horloge. J'ai percé 60 trous de 5mm sur la face de l'horloge pour y insérer les LED ainsi que 3 trous sur la plaque du fond pour pouvoir mettre un interrupteur et des boutons poussoirs pour changer l'heure.

Conclusion

Mon projet était de créer une horloge à LED basé sur 3 types d'affichages: un affichage traditionnelle, binaire et numérique. Malgré ma persévérance et mon implication dans ce projet qui me tenu à cœur, j'ai vu un projet trop grand. C'est-à-dire que je ne pensais pas que le projet allé devenir si complexe et long à réaliser.

Je n'ai pas pu aboutir ce projet pour plusieurs raison. D'une part j'ai cruellement manqué de temps. Les séances d'études et réalisations sont trop peu nombreux durant le semestre 4 pour permettre la réalisation d'un projet de cette taille. Je pense que si j'avais travaillé ce projet durant le semestre 3 j'aurai pu le mener à terme.

La seconde raison est dû à mes recherches. En débutant mon projet, je n'ai aucune idée de comment le réaliser, et je suis parti sans aucune base de travail sur quoi m'appuyer. Cela m'a alors pris beaucoup de temps avant de me diriger sur des pistes de réflexion et de travaille assez solide pour démarrer concrètement le projet.

La dernière raison de mon retard est le RTC. Effectivement malgré l'aspect très avantageux de ce composant celui-ci communique avec un microcontrôleur en bus I2C. Cette communication m'a posé problème pour l'unique raison que je ne le connaissais absolument pas. J'ai dû me renseignement, l'étudier, le comprendre.

Pour finir, je dirais que ce projet m'a apporté que du positif malgré son non-aboutissement. Tout d'abord, il m'a permis d'acquérir de nouvelles connaissances, notamment sur le bus I2C. Il m'a aussi apporter une réelle motivation à réaliser un projet seul et par mes propres moyens. J'ai pu aussi voir tout les implications qu'un projet engendre, de part sa recherches jusqu'à sa réalisation.

Malgré l'amertume face à l'échec de mon projet je garde tout fois espoir de pouvoir le finir un jour.

Résumé

Mon projet fut de créer une horloge à LED complètement où l'heure doit s'afficher autour du cadran, en binaire à l'aide de LED et en manière numérique grâce à des afficheurs 7-segments. Celle-ci doit laisser toute l'électronique apparente en plus de garder l'heure en mémoire en cas de coupure de courant.

Pour cela j'ai effectué énormément de recherches, trouvé des solutions mais ces dernières ne répondaient pas au cahier des charges . Grâce à l'aide de certains professeurs et des recherches plus approfondies, j'ai pu trouver la meilleure solution pour mon projet: un RTC. Je suis donc parti de cet unique composant pour faire toutes les autres recherches.

Tout d'abord, le pilotage du RTC par un microcontrôleur fut rapide à trouver. J'ai utilisé un microcontrôleur dont je connaissais parfaitement, un Atmega8535. Il vient ensuite le problème de la communication. Ne connaissant pas le protocole I2C j'ai du l'étudier pour le comprendre. Ensuite les solutions pour les divers affichages sont venues rapidement et naturellement comme par exemple la CPLD.

Mais la compréhension de ces composants m'ont fait perdre un temps considérable sur mon projet, me permettant ainsi de savoir que je n'aurai pas le temps nécessaire pour le concrétiser jusqu'au bout. Je me suis alors fixé en objectif de faire toute la partie théorique. C'est-à-dire de comprendre tous les composants et leur fonctionnement respectif et de faire le schéma électrique de l'horloge.

J'ai pu aussi élaborer le programme de la CPLD, servant à l'affichage autour du cadran. Ce programme fut très rapidement établi, et a pu être simulé grâce au logiciel QUARTUS.

Malheureusement il me reste beaucoup à faire pour finir le projet entièrement. Dans un premier temps, il reste les cartes électroniques à mettre en place. Puis le programme principal du microcontrôleur reste à être établir, représentant le plus gros du travail.

Nombre de mots : 302

Index des illustrations

Bibliographie

[0] **Corolab**. *Horloge BCD*, 2009, [En ligne]. (Page consultée le 02/12) <http://corolab.unblog.fr/2009/05/04/horloge-bcd/>
[0] **Bram Knaapen**. *Equinox Clock*, 2010, [En ligne]. (Page consultée le 02-12) <http://www.bramknaapen.com/?p=549>
[0] **Jean-Louis Raynal**. *Horloge binaire à ampoules incandescentes*, 2011, [En ligne]. (Page consultée le 02/12) <http://www.jlr-blog.com/?menu=afficher_projet&id_projet=42>
[0] **samuel goutenoir**. *DS1307*, 2010, [En ligne]. (Page consultée le) <http://samuel.goutenoir.com/archives/134>
[1] . *Datasheet ATmega8535*, , [En ligne]. (Page consultée le 02/12) <http://www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA8535.shtml>
[2] **Wikipédia**. *Complex programmable logic device*, , [En ligne]. (Page consultée le 02/12) <http://en.wikipedia.org/wiki/Complex_programmable_logic_device>
[3] **Aurelien Jarno**. *le bus I²C*, , [En ligne]. (Page consultée le 02/12) <http://www.aurel32.net/elec/i2c.php>

Annexes

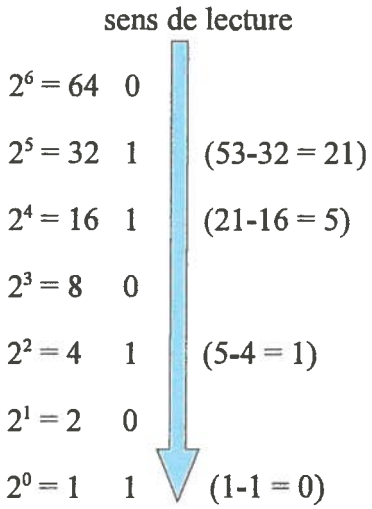
Le langage binaire

Le langage binaire utilisé dans ce projet va transcrire un nombre entier dit en base de 10 en nombre binaire à l'aide de 0 et de 1 dit en base de 2. Ce nombre sera composé de **N** bits.

Chaque bit est situé dans ce nombre suivant un rang **R**. Pour transcrire un nombre décimal **D** en nombre binaire **B**, on soustrait le nombre D le nombre le plus proche en puissant de 2. A chaque puissant utilisé celui correspond à un 1 dans le nombre B. Et on répète l'opération jusqu'à avoir 0.

Pour transcrire le nombre B en nombre D, on additionne toutes les puissances de 2 au rang R où il y a un 1.

Pour le nombre décimal $(53)_{10}$:



Donc : $(53)_{10} = (0110101)_2$

Pour le nombre binaire $(101010)_2$:

B	1	0	1	0	1	0
rang	5	4	3	2	1	0

D $2^5x1 + 2^4x0 + 2^3x1 + 2^2x0 + 2^1x1 + 2^0x0$

$32 + 0 + 8 + 0 + 2 + 0$

42

Donc : $(101010)_2 = (42)_{10}$

Le langage DCB

Le langage DCB est une « variante » du langage binaire, Pour un nombre entier, celui ci sera composé d'autant de groupements de 4bits en binaire qu'il y a de chiffres dans ce nombre. On dit que l'on a un nombre en Binaire Naturel (DCB)

Par exemple: $(23)_{10}$ = $(2)_{10}$ et $(3)_{10}$

= $(0010)_2$ et $(0011)_2$

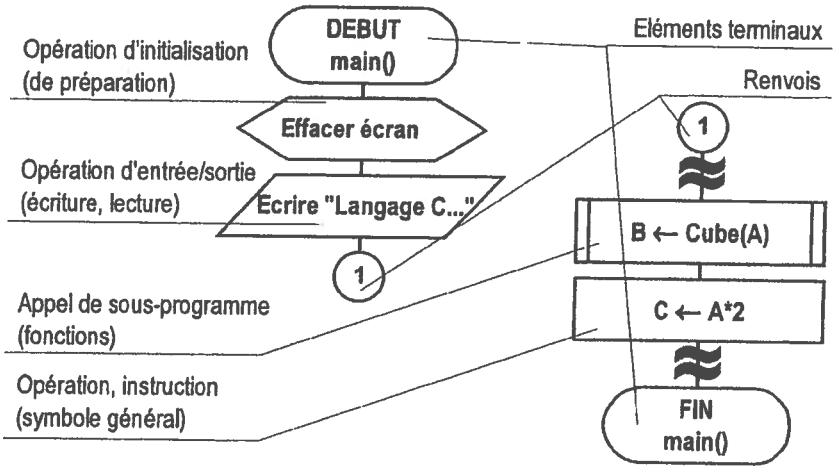
$(23)_{10}$ = $(0010\ 0011)_2$

Nombre en DCB				**Chiffre en décimal**
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

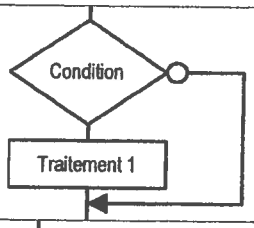
ORGANIGRAMMES DE PROGRAMMATION ET LANGAGE C

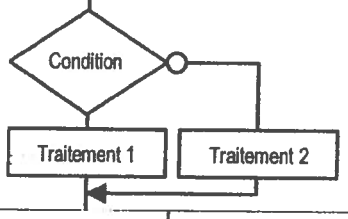
D'après NF Z67-010 Août 1975

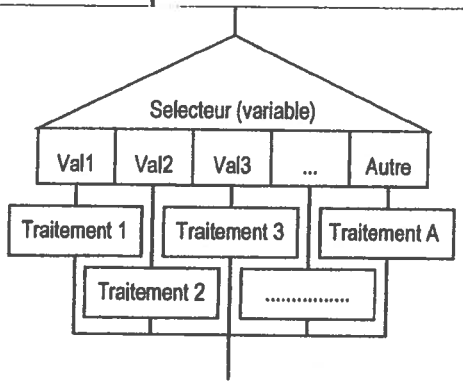
Éléments terminaux et opérations



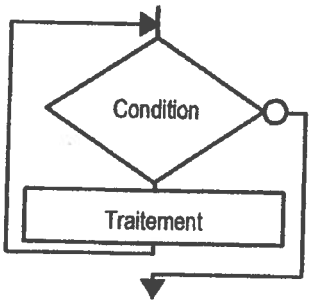
Structures alternatives

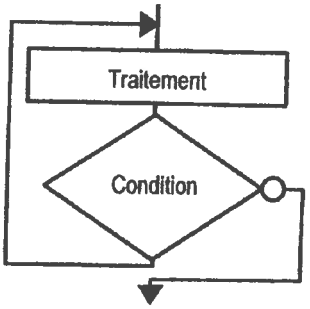
Alternative simple		`if ( Condition )`
		`{`
		`  // Traitement1 ;`
		`}`

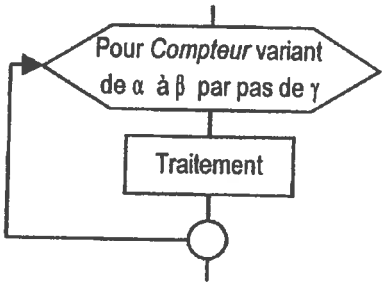
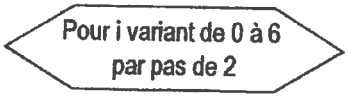
Alternative double		`if ( Condition )`
		`{`
		`  // Traitement1 ;`
		`}`
		`else`
		`{`
		`  // Traitement2 ;`
		`}`

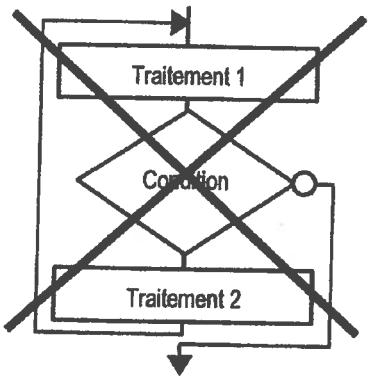
Sélecteur de choix		`switch ( Selecteur )`
		`{`
		`  case Val1 :`
		`  // Traitement1 ;`
		`  break ;`
		`  case Val2 :`
		`  // Traitement2 ;`
		`  break ;`
		`  case Val3 :`
		`  // Traitement3 ;`
		`  break ;`
		`  /* … … … … … … */`
		`  default :`
		`  // TraitementA ;`
		`}`

Structures répétitives

Tant que...
(itération à condition d'entrée)
		`while ( Condition )`
		`{`
		`  // Traitement ;`
		`}`

Faire...
tant que
(itération à condition de sortie)
		`do`
		`{`
		`  // Traitement ;`
		`}`
		`while ( Condition ) ;`

Pour...
(itération avec compteur)
	
	Exemple :
	 
		`for ( Init. ; Cond. ; Pas )`
		`{`
		`  // Traitement ;`
		`}`
		Exemple :
		`for (i=0; i<=6; i=i+2)`
		`{`
		`  // Traitement ;`
		`}`

Structure non programmable (forme générale d'une structure d'itération)		Cette structure n'existe ni en C, ni en Pascal !

Remarques

Toutes les extrémités d'un symbole doivent posséder un chemin.
Chaque structure de base n'a qu'une seule entrée et qu'une seule sortie.
Les structures peuvent être imbriquées les unes dans les autres (une opération de traitement peut être composée d'une ou de plusieurs structures).
Il n'existe qu'un seul point d'entrée et qu'un seul point de sortie pour l'ordinogramme. Pour le programme principale, il peut ne pas exister de point de sortie (boucle infinie).

Programme de la CPLD

```
module horloge (clock, data, affichage,envoie);    // Déclaration du nom du programme avec
                                                   // toutes les variables utilisées
    input clock;
    input [5:0] data;                              // Déclaration des entrées ( [5:0] nombre de
    input [1:0]mode;                                      // bit soit ici 6)

    output [59:0] affichage;                              // Déclaration des sorties

    reg [59:0] minute;                             // Déclaration des registres pour garder en
    reg [11:0] heure;                              // mémoire la valeur des valeurs

    always@ ( posedge clock )                      // A chaque front montant de clock
        begin
            if (mode == 0)                         // on regarde si MODE est à 0
                begin
                    minute = 2**data;  // on affecte à minute la valeur 2^DATA
                end

            if (mode ==1)                          // on regarde si MODE est à 1
                begin
                    heure = 2**(data*5);  // On affecte à heure la valeur 2^DATA
                end
            if  (mode==3)
                begin
                    seconde=2**data;
                end

            affichage = 0;                         // on remet à zéro l'affichage
            affichage[minute] = 1;                 // on affecte un 1 logique au bit minute
            affichage[heure] = 1;                  // on affecte un 1 logique au bit heure
            affichage[seconde] = 1;                // on affecte un 1 logique au bit seconde
        end
endmodule                                          // Fin du programme
```