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

Projet Tutoré – Étude et Réalisation

Éclairage d'ambiance à LEDs

SAPIENS Corentin – PETITEAU Ivan
Année 2011 – Groupe Q1
Promotion 2009 - 2011

Enseignants : M. LEQUEU
M. GLIKSOHN

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

Projet Tutoré – Étude et Réalisation

Éclairage d'ambiance à LEDs

SAPIENS Corentin – PETITEAU Ivan
Année 2011 – Groupe Q1
Promotion 2009 - 2011

Enseignants : M. LEQUEU
M. GLIKSOHN

Sommaire

Introduction....4

1.Présentation du sujet....5

1.1.Cahier des charges....5

1.2.Généralités sur la MLI....6

1.3.Planning....7

2.Études du système....9

2.1.Solutions envisagées....9

2.2.Nomenclatures....12

2.3.Les principaux composants....13

2.4.La Modulation de Largeur d'Impulsion logicielle....15

2.5.Initialisation de CodeVision AVR....16

3.Réalisation....19

3.1.Routage sur Eagle....19

3.2.Programmation....26

3.3.Problèmes rencontrés....38

3.4.Améliorations possibles....38

Conclusion....39

Résumé....40

Annexes....43

Introduction

Dans le cadre de notre 4e semestre à l'IUT, nous devons réaliser un projet en Étude et Réalisations basé en priorité sur la programmation informatique. Nous avons choisi le projet d'éclairage d'ambiance à LED, déjà débuté par Corentin SAPIENS quelques mois auparavant. Ce projet est basé sur le principe de la MLI[1] afin de créer des effets de lumière.

Le projet est divisé en 2 parties: la partie électronique, et la partie programmation. Nous allons utilisés le principe de la MLI afin de faire varier, de façon agréable, les couleurs et l'effet que cela produira dans la pièce. Le projet ayant déjà été commencé un petit peu par Corentin, la plus grande partie de notre travail sera basé sur la programmation, à l'aide du logiciel CodeVision AVR. D'autre part nous avons récupéré les premiers typons et avons pu ainsi, améliorer son principe et son coté esthétique.

Ce rapport commence par la présentation du sujet, suivi par l'étude technologique du système et enfin par la méthode de réalisation électronique et informatique du projet.

1 Modulation de Largeur d'Impulsion

1. Présentation du sujet

Les lampes et autres appareils lumineux étaient, autrefois, avant tout utilisés pour pouvoir continuer à lire ou à travailler lorsque la nuit était tombée. Pourtant, il y a de plus en plus de gens qui possède de petite lampe ayant pour but de créer une ambiance agréable grâce aux couleurs qu'elle dégage. On retrouve également ces effets dans les boîtes de nuit ou autre salle des fêtes où les personnes viennent pour s'amuser ou danser, éclairées par de puissants spots avec des filtres de couleurs afin de créer une ambiance appropriée à la soirée, et ainsi avoir une sensation de bien être.

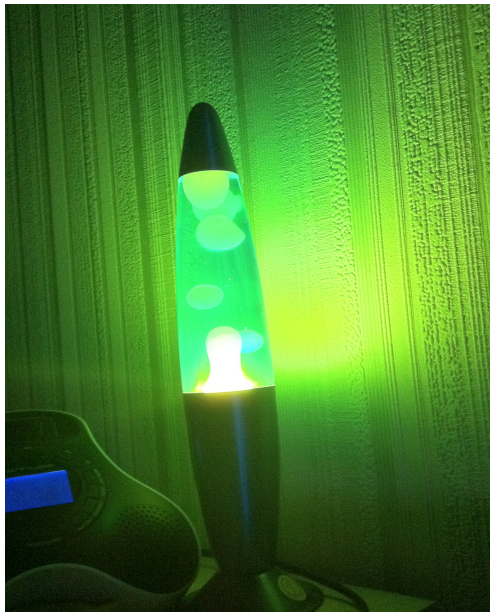

Illustration 1: Exemple de lampe d'ambiance [1]

1.1. Cahier des charges

Ce sont des dispositif qui ne se trouvent pas encore beaucoup dans le commerce à l'exception des magasins spécialisé et sont donc très chers. Corentin a donc préféré en fabriquer une lui même pour en comprendre le fonctionnement. Ce projet a un but personnel et, une fois fini, servira à une utilisation domestique, dans une chambre.

Tous les schémas ont été réalisés par Corentin pendant son temps libre. Et nous avons décidé de continuer sa réalisation comme projet tutoré de notre semestre 4.

Ce système peut être branché sur n'importe quelle prise secteur 230V, via une alimentation à découpage 230V/12V. Il sera placé dans un boitier, le plus compact possible afin d'économiser de la place (dimension 190x125x45 mm), et respectera les normes de CEM[2].

Illustration 2: Boitier [1]

Une interface de navigation est prévue grâce à un écran LCD 4 lignes, et 4 boutons poussoirs, afin de pouvoir régler les signaux.

Pour nos séances de réalisation nous mettrons les LEDs sur une carte à part, pour plus tard être placées dans un endroit plus adapté. S'agissant d'un projet personnel, nous allons faire en sorte qu'il coûte le moins possible.

2 Compatibilité Electro-Magnétique

1.2. Généralités sur la MLI

La MLI ou Modulation de Largeur d'Impulsions est une technique permettant de simuler une tension variable en agissant sur le rapport cyclique[3] du signal.

La MLI la plus classique consiste à comparer, à l'aide d'un AOp[4], un signal continu à un signal triangulaire ou en dent de scie. Nous obtiendrons alors un signal en créneau dont nous pourrons facilement modifier le rapport cyclique en agissant sur la composante continue.

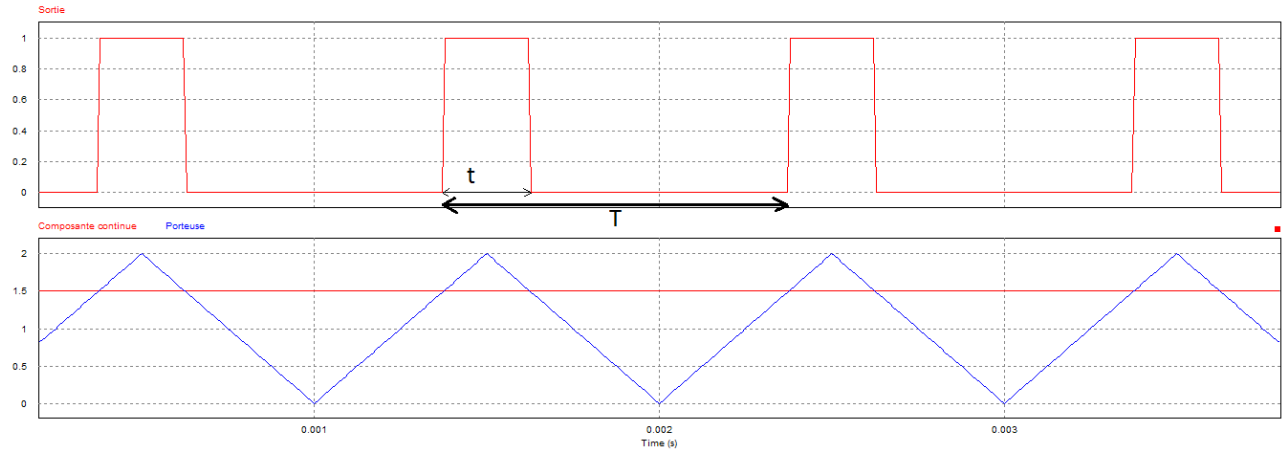

Illustration 3: Oscillogramme d'une MLI [1]

Rapport cyclique: Alpha=t/T

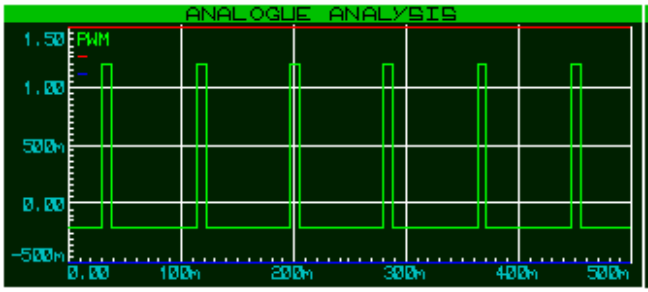

Illustration 4: MLI à 10% [2]

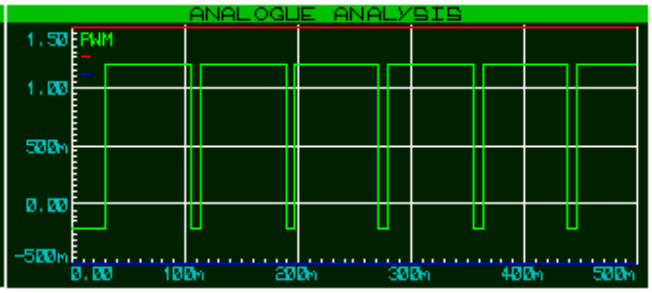

Illustration 5: MLI à 90% [2]

On peut alors utiliser ce signal afin d'alimenter nos appareils, et faire varier la tension qui les traverse (variations de lumières). Même si un AOp délivre un faible courant, il est possible d'utiliser des transistors qui reproduiront le signal mais permettront d'alimenter des appareils de fort courant (variateur de vitesse sur un moteur).

3 Largeur des impulsions à 1 par rapport à 0
4 Amplificateur Opérationnel

Il est alors possible de rajouter un filtre juste avant la charge afin d'obtenir des signaux continus, égaux à la valeur moyenne du signal de sortie, afin de ne pas endommager les appareils sensibles.

Il est également plus facile d'utiliser un simple amplificateur de puissance, en revanche il coûte cher. Utiliser un amplificateur de puissance coûtera entre 2 et 3 fois plus cher qu'un AOp à faible courant réglé en comparateur et des transistors pour des puissances équivalentes.

1.3. Planning

Pour répondre au cahier des charges, nous avons bénéficié de huit séances de projet et de quatre séances libres. Nos typons ont été tracés à l'aide du logiciel Eagle. Le programme a été réalisé sous CodeVision AVR. Le planning prévisionnel a été bien respecté, nous avons seulement pris un peu plus de temps que prévu sur la création de la carte, du fait des nombreuses modifications que nous avons apporté. Le rapport détaillé des semaines figure en annexe 4 :

Semaines	5	6	7	8	9	10	11	12	13	14
Tâches				Vacances						
Découverte et compréhension du projet										
Réalisation du cahier des charges										
Réalisation de la carte										
Test de la partie électronique de la carte										
Programmation du µC										
Test de la partie programmation de la carte										
Réalisation de la lampe										
Réalisation du rapport										
					Prévisionnel			Réel		

Illustration 6: Planning prévisionnel et réel

2. Études du système

2.1. Solutions envisagées

2.1.1. Partie électronique

Au départ du projet, il était prévu de l'alimenter via une alimentation à découpage d'ordinateur, afin d'obtenir directement du 5V sur la carte. Sur le premier prototype effectué, les boutons poussoirs et l'écran LCD se trouvaient sur la même carte que le micro-contrôleur.

Nous avons décidé de remplacer l'alimentation d'ordinateur par des régulateur de tensions 5V, ce qui permet ainsi d'alimenter les cartes grâce à une alimentation à découpage 12V que l'on peut trouver dans n'importe quelle grande surface. Nous avons aussi décidé de séparer les boutons poussoirs et l'écran de la carte principale contenant le micro-contrôleur, afin de pouvoir les faire tenir dans un boitier adéquat.

Voici le schéma fonctionnel de notre projet:

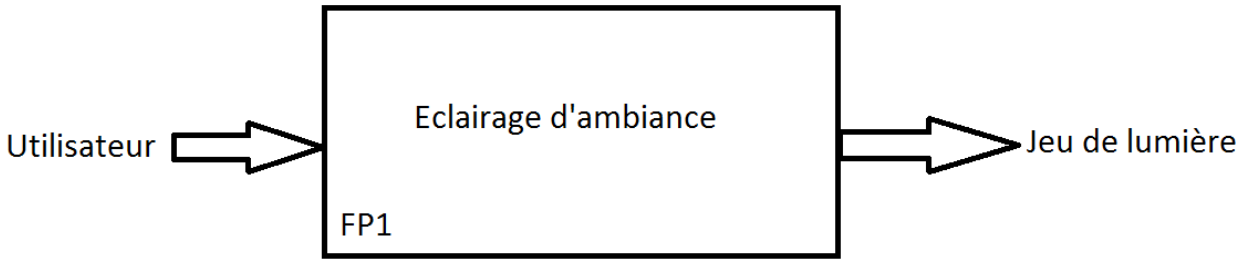

Illustration 7: Schéma fonctionnel de niveau 1 [1]

Un utilisateur pourra contrôler les effets de lumière.

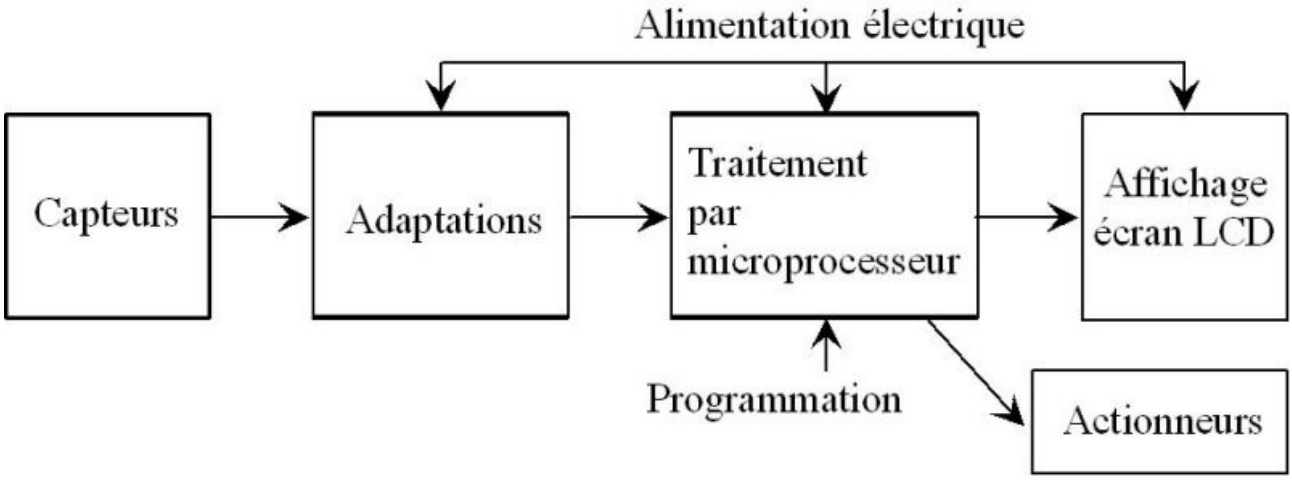

Illustration 8: Schéma fonctionnel de niveau 2 [3]

Les capteurs seront réalisés par des boutons poussoirs, au nombre de quatre, afin de permettre la navigation dans un menu.

C'est le micro-contrôleur qui gérera le traitement des entrées, et les interprétera pour permettre au microprocesseur de commander l'état des sortie via un programme que nous avons écrit. Il enverra aussi les informations à afficher sur l'écran LCD.

Le système sera alimenté par deux régulateurs de tensions, afin d'avoir une alimentation stable.

2.1.2. Partie informatique

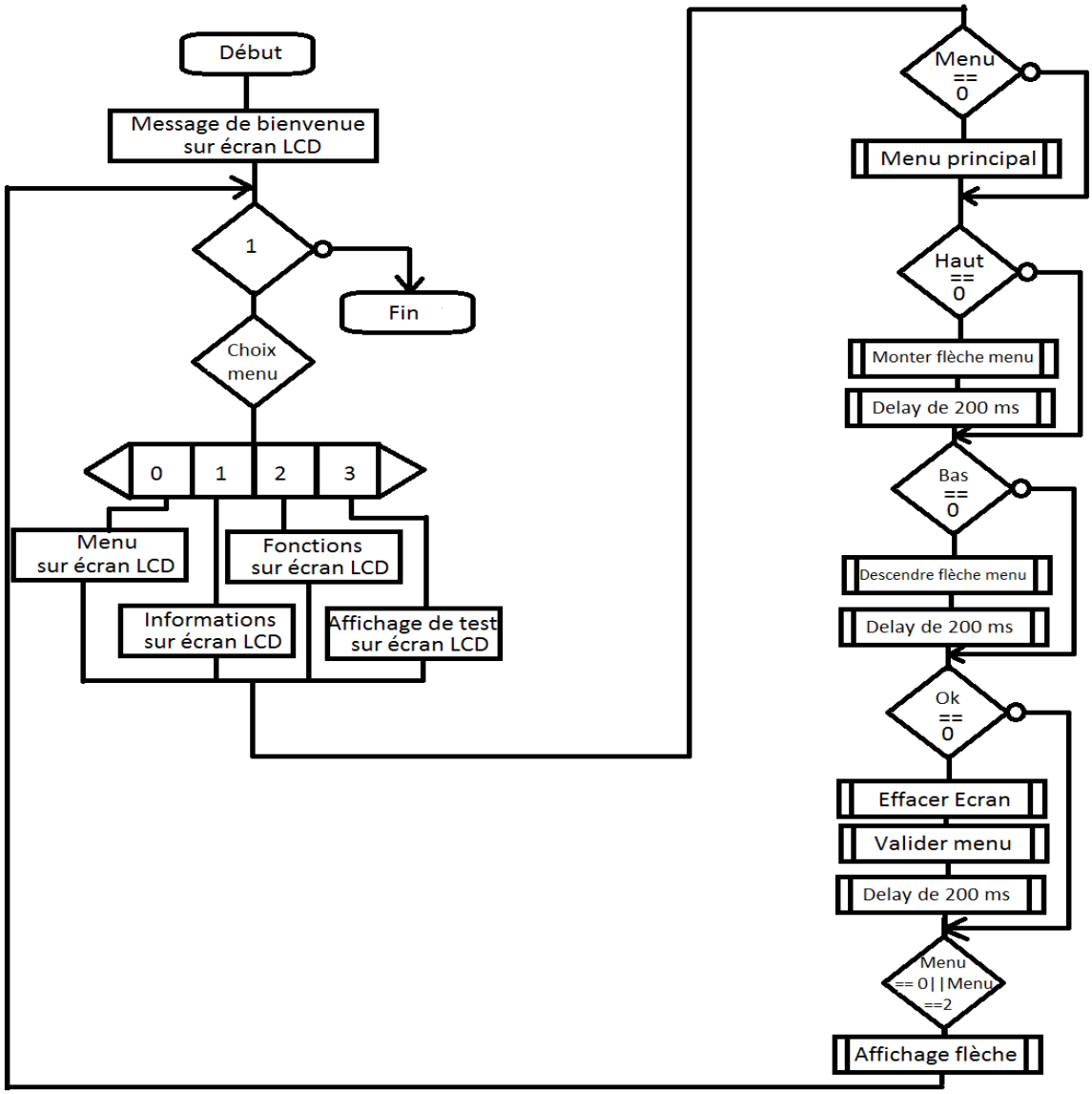

Illustration 9: Ordinogramme du "main()" [1]

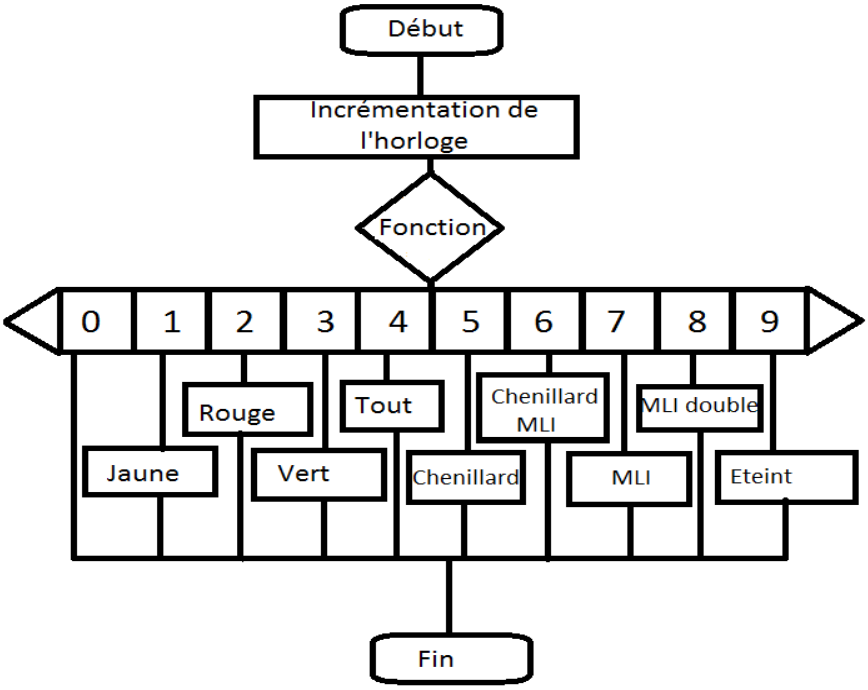

Illustration 10: Ordinogramme des interruptions[1]

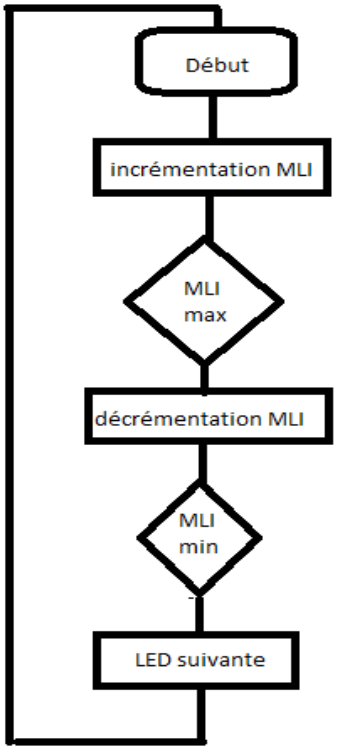

Illustration 11: Orgamigramme de la MLI[1]

2.2. Nomenclatures

Pour respecter le budget, les composants les moins chers ont été choisi. Voici leur nomenclature et le prix de chacun :

Fournisseur	Nature du composant	Référence	Prix unitaire	Quantité	Prix total
Gotronic	Diode1N5822	44017	0,50 €	2	1,00 €
Gotronic	Diode 1N5819	44016	0,15 €	2	0,30 €
Gotronic	Bornier à vis double	08090	0,30 €	1	0,30 €
Gotronic	Condensateur 100 µF	04935	0,15 €	1	0,15 €
Gotronic	Condensateur 470 µF	04947	0,20 €	1	0,20 €
Gotronic	Condensateur 1000 µF	04952	0,35 €	1	0,35 €
Gotronic	Condensateur 10 µF	04909	0,10 €	1	0,10 €
Gotronic	Condensateur 100 nF	04817	0,15 €	1	0,15 €
Gotronic	Condensateur 22 pF	15816	0,08 €	2	0,16 €
Gotronic	Inductance 47µH	15474	0,85 €	1	0,85 €
Gotronic	Inductance 100µH	15441	1,40 €	1	1,40 €
Gotronic	Connecteur 10 broches droit	09780	0,50 €	3	1,50 €
Gotronic	Quartz 16MHz	05270	1,50 €	1	1,50 €
Gotronic	LM2574	42432	1,65 €	1	1,65 €
Gotronic	tulipe LM2574	08161	0,08 €	1	0,08 €
Gotronic	LM2576	42636	2,40 €	1	2,40 €
Gotronic	LED verte Ø 3mm	03002	0,15 €	3	0,45 €
Gotronic	Atméga 8535	16006	5,70 €	1	5,70 €
Gotronic	tullpe Atméga	08169	0,37 €	1	0,37 €
Gotronic	Transistors BC337-40	02174	0,15 €	15	2,25 €
Gotronic	Port parallèle femelle		1,90 €	1	1,90 €
				Total	22,76 €

Tableau 1: Nomenclature de la carte principale

Fournisseur	Nature du composant	Référence	Prix unitaire	Quantité	Prix total
Gotronic	Condensateur 10 µF	04909	0,10 €	2	0,20 €
Gotronic	Condensateur 100 nF	04817	0,15 €	2	0,30 €
Gotronic	Connecteur 10 broches coudé	09790	0,50 €	1	0,50 €
Gotronic	écran LCD + tulipe		7,50 €	1	7,50 €
Gotronic	Potentiomètre 10kΩ		0,10 €	1	0,10 €
Gotronic	Résistance 10kΩ		0,10 €	1	0,10 €
Magasin IUT	Boutons poussoirs		0,75 €	4	3,00 €
Gotronic	Connecteur 10 broches coudé	09790	0,50 €	1	0,50 €
Gotronic	Boitier 190x125x45	11028	4,90 €	1	4,90 €
Maitre LEQUEU	LED verte Ø 5mm		0,20 €	36	7,20 €
Maitre LEQUEU	LED jaune Ø 5mm		0,20 €	36	7,20 €
Maitre LEQUEU	LED rouge Ø 5mm		0,20 €	36	7,20 €
Gotronic	Port parallèle mâle		0,85 €	1	0,85 €
Gotronic	protection port parallele		0,60 €	1	0,60 €
Gotronic	Résistances 1,5kΩ		0,10 €	4	0,40 €
Gotronic	Résistances 1,8kΩ		0,10 €	12	1,20 €
Magasin IUT	Gaine spiralé			1	
IKEA	Support de lampe		15,00 €	1	15,00 €
CASTORAMA	Tube de PVC		1,50 €	1	1,50 €
				Prix total	56,75 €

Tableau 2: Nomenclatude des autres carte et de la lampe

Le prix d'une lampe est donc d'environ 80€ TTC. Il faut savoir que plus les composants sont commandés en quantité, plus le coût total diminue.

2.3. Les principaux composants

2.3.1. Le micro-contrôleur ATMEGA 8535

L'ATMEGA 8535 est un micro-contrôleur très utilisé à l'IUT. C'est pourquoi il a été choisi, car il est facilement disponible au magasin de l'IUT. Il n'y avait donc pas eu de besoin de créer de commande.

Il dispose de nombreuses fonctionnalités sur les entrées/sorties, comme des entrées analogiques, des interruptions externes, des timers permettant de réaliser de la MLI,...

Ce composant permet après programmation de gérer des entrées/sorties en autonomie, il écrit aussi sur l'afficheur. Pour fonctionner, il a besoin d'un montage annexe composé d'un quartz et de condensateurs. Celui-ci va permettre de donner une vitesse de calcul comme dans le cas d'un ordinateur.

Afin de programmer l'ATMEGA 8535, on utilise le logiciel CodeVision AVR. La programmation se fait en C, en utilisant les fonctions basiques de ce langage.

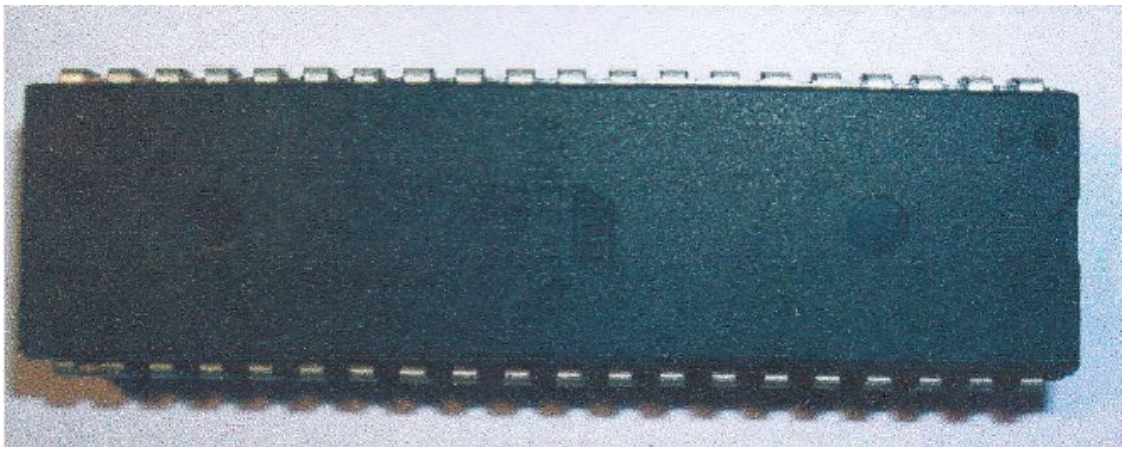

Illustration 12: ATMEGA 8535[1]

2.3.2. Le régulateur de tension LM2574N-5

Le LM2574N-5 est un régulateur de tension, capable de supporter une source de tension comprise entre 7 et 40 volts continu. Il va faire en sorte d'obtenir 5 volts en sortie en toutes circonstances, et délivre en sortie un courant maximal de 500 mA. C'est pourquoi lors des essais, le montage a été alimenté avec une alimentation continue de 15 volts. Le bon fonctionnement du régulateur est indiqué par une LED verte positionnée à coté de l'afficheur.

Nous utilisons aussi un régulateur LM2576, qui délivre un courant maximal de 3 A .

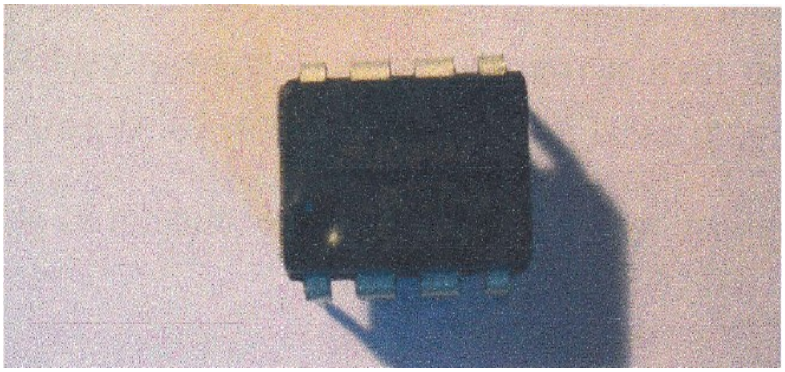

Illustration 14: Régulateur de tension LM2574N-5 [1]

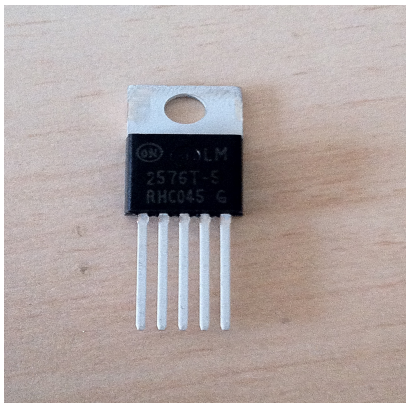

Illustration 13: Régulateur de tension LM2576 [1]

2.3.3. L'afficheur 16x4 caractères

L'afficheur LCD choisi, dispose de 16 caractères sur 4 lignes. L'écran est choisi rétro-éclairé ce qui permet un meilleur confort lorsqu'il fait sombre ou même nuit, avec la possibilité de l'éteindre lorsque l'utilisateur le souhaite. Il affiche grâce aux données transmises par l'ATméga, des informations pratiques pour l'utilisateur, la navigation dans un menu, et les différents réglages possibles.

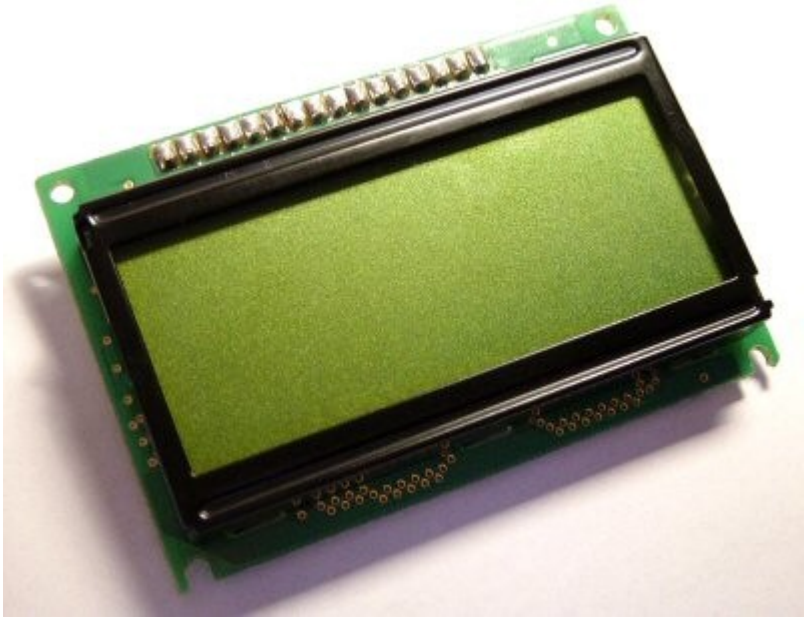

Illustration 15: Ecran LCD [2]

2.4. La Modulation de Largeur d'Impulsion logicielle

Nous devions contrôler par MLI douze transistors de façon indépendante, mais comme nous pouvons le voir dans la datasheet de l'ATméga [4], il ne disposait que de quatre sorties basées sur des timers, ce qui était bien insuffisant pour réaliser le projet initial. Donc nous nous sommes tournés vers une autre façon de la faire, et nous avons décidé de la réaliser de façon logicielle (grâce au logiciel CodeVision AVR). Bien que plus complexe, cela nous a permis de contrôler chaque transistor indépendamment.

2.5. Initialisation de CodeVision AVR

CodeVision AVR est un logiciel de programmation en C qui permet d'utiliser toutes les fonctions d'un micro-contrôleur, c'est pourquoi nous l'avons choisi. Lors de la création du projet sur CodeVision, il nous suffit d'insérer les différentes données (écran LCD, interruptions...)

Une fois CodeVision AVR ouvert, il faut cliquer sur « Fichier » → « New », puis sélectionner « Poject ». Une page s'affichera alors pour demander si on veut utiliser CodeWizartAVR, il faut répondre « Yes ».

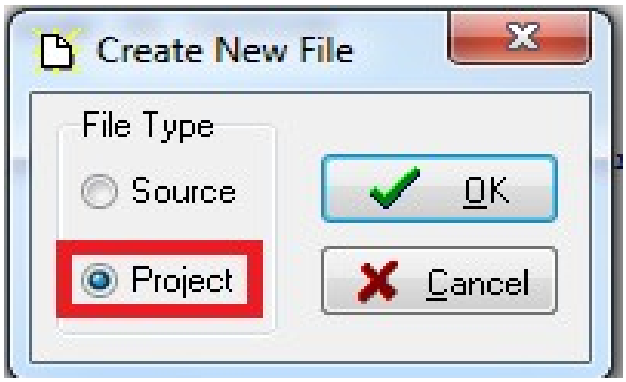

Illustration 16: CodeVision Initialisation 1 [1]

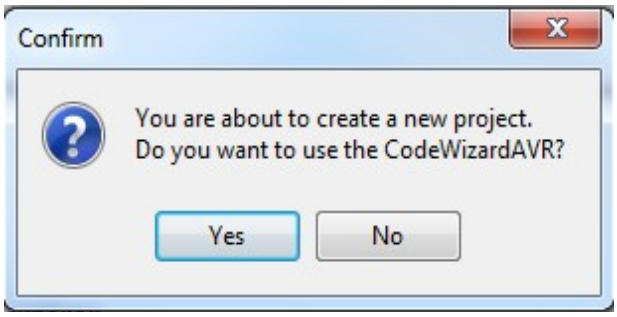

Illustration 17: CodeVision Initialisation 2 [1]

Une nouvelle fenêtre va s'ouvrir et va demander de choisir tous les composants qui seront connectés au micro-contrôleur.

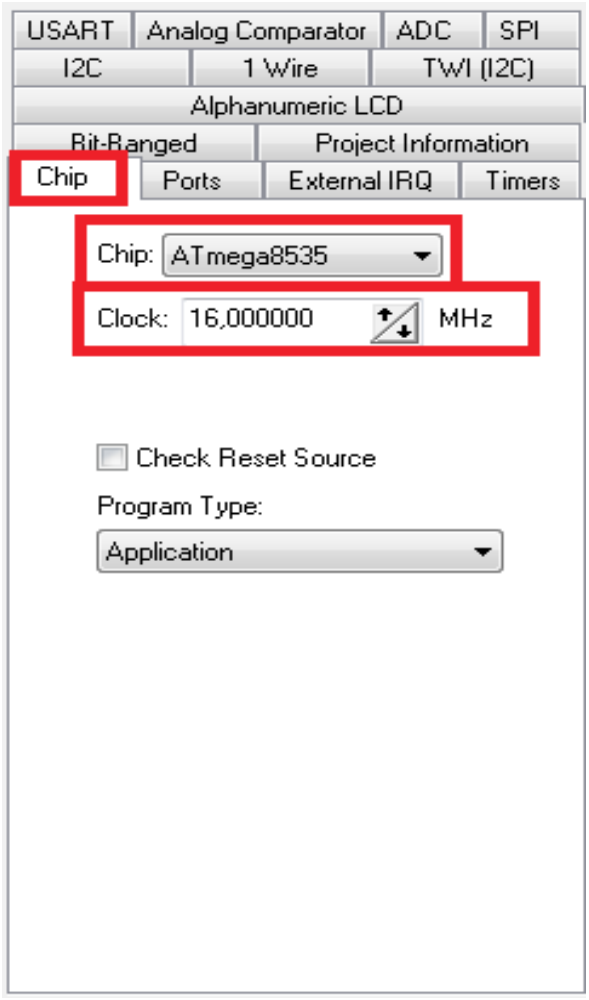

Illustration 18: CodeVision Initialisation 3 [1]

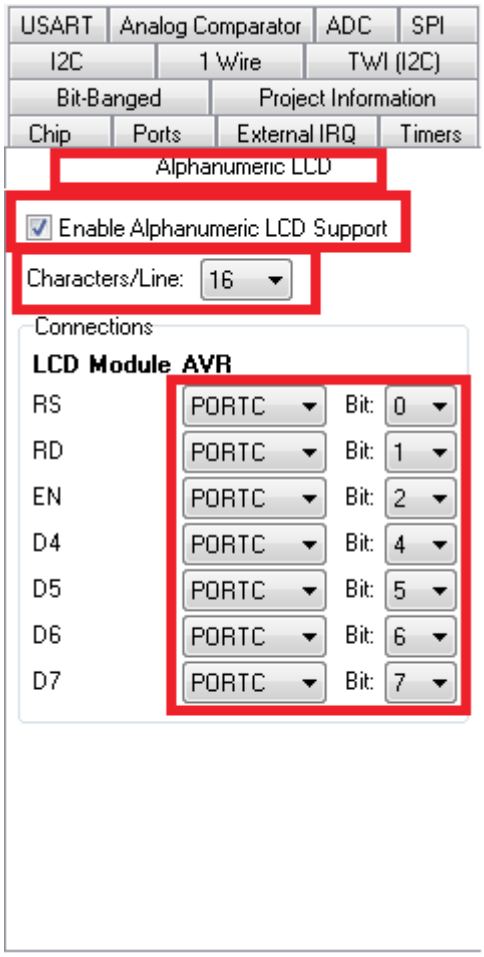

Illustration 19: CodeVision Initialisation 4 [1]

Nous avons choisi un ATméga 8535 avec une horloge 16 MHz, un écran LCD que nous avons connecté sur le PORTC.

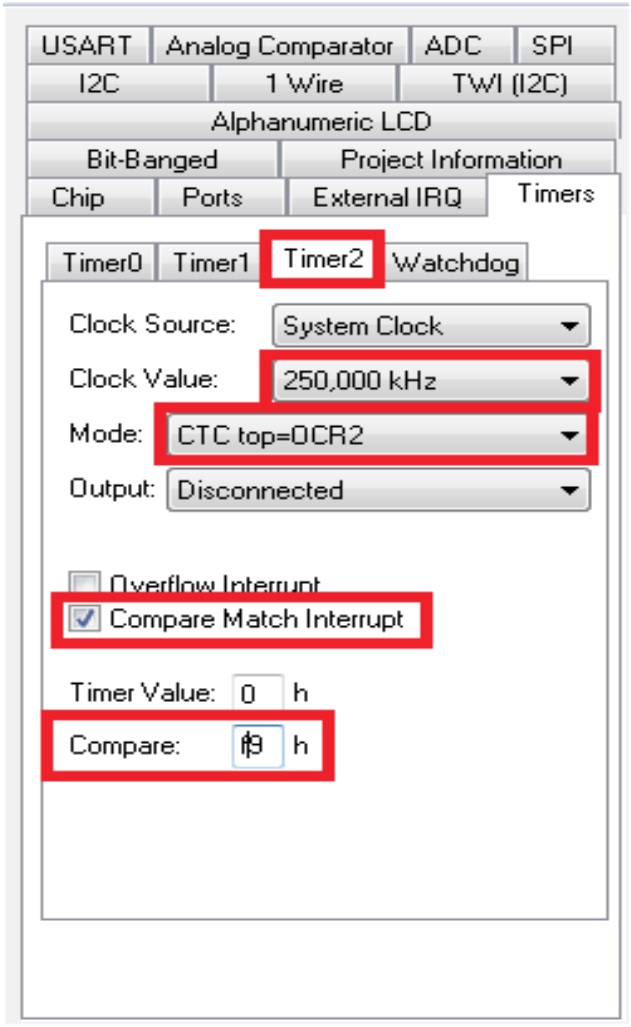

Illustration 20: CodeVision Initialisation 5 [1]

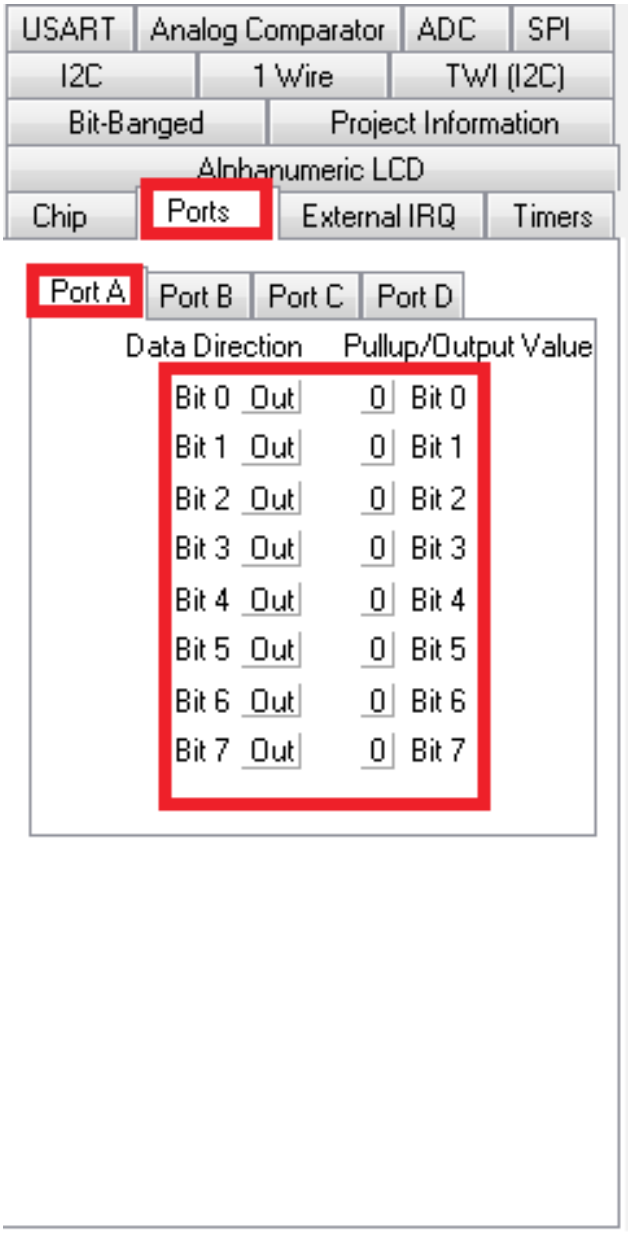

Illustration 21: CodeVision Initialisation 6 [1]

Il faut demander à CodeVision de générer le code, pour cela « File » → « Generate, Save and Exit ».

Il y a alors 3 fichiers à sauvegarder, le .c, le .prj et le .cwp. Afin d'éviter tout problème de compilation avec CodeVision, les 3 fichiers doivent être sauvegardés sous des noms différents.

On peut également sélectionner « Program Preview » qui permet une prévisualisation du code d'initialisation sans effacer notre code actuel, si on rajoute un composant par exemple.

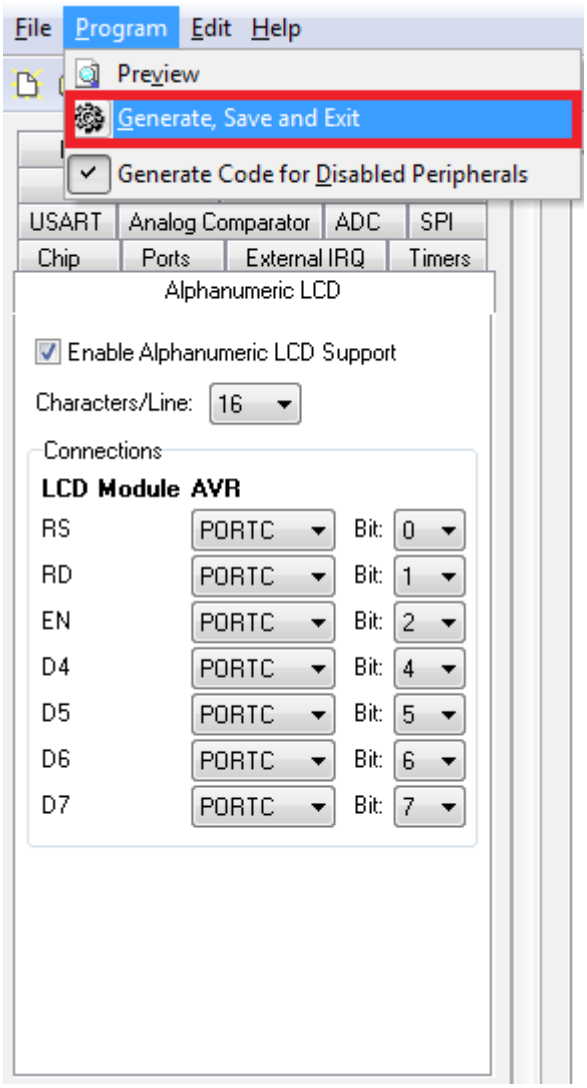

Illustration 22: CodeVision Initialisation 7 [1]

3. Réalisation

La programmation de l'ATméga 8535 est faite à partir de CodeVision AVR, un logiciel de programmation en C payant.

3.1. Routage sur Eagle

Les typons créés pendant son temps libre par Corentin ont été réalisés avec le logiciel Eagle. Les prototypes réalisés pendant nos séances d'études et réalisations nous ont permis de tester nos premiers programmes pour l'ATméga et nous a permis de nous rendre compte de certains problèmes liés à la réalisation des cartes.

3.1.1. Schéma d'alimentation

Voici donc le schéma électrique de l'alimentation de la partie puissance et de la partie commande, séparées pour des raisons de sécurité, dont les régulateurs permettent d'obtenir les tensions (5 volts) nécessaires aux cartes. Deux diodes Schottky[5] ont été installées afin de protéger les cartes contre les inversions de tensions.

Le régulateur de la partie commande délivre un courant maximal de 500 mA ce qui est suffisant pour alimenter l'ATméga, l'écran LCD et les boutons poussoirs.

Le régulateur de la partie puissance délivre un courant maximal de 3 A et alimentera toutes les LEDs.

Nous aurions pu grouper la partie puissance et la partie commande mais cela nous aurait fait acheté un régulateur plus puissant, donc plus cher, ce qui n'était pas nécessaire.

Une diode verte pour chaque régulateur montre le bon fonctionnement de ceux-ci.

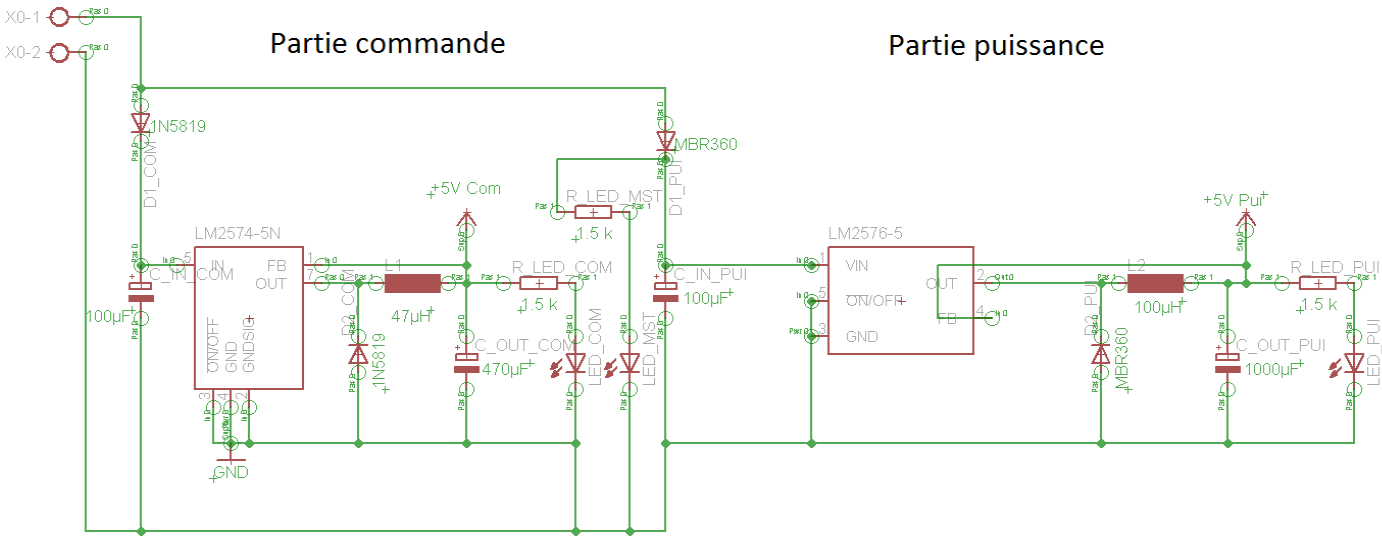

Illustration 23: Schéma électrique de la partie alimentation[1]

3.1.2. La partie ATMEGA 8535

Le micro-contrôleur possède le schéma électrique le plus complexe. En effet, il est le centre de la carte et gère tous les composants.

5 Diode à commutation très rapide

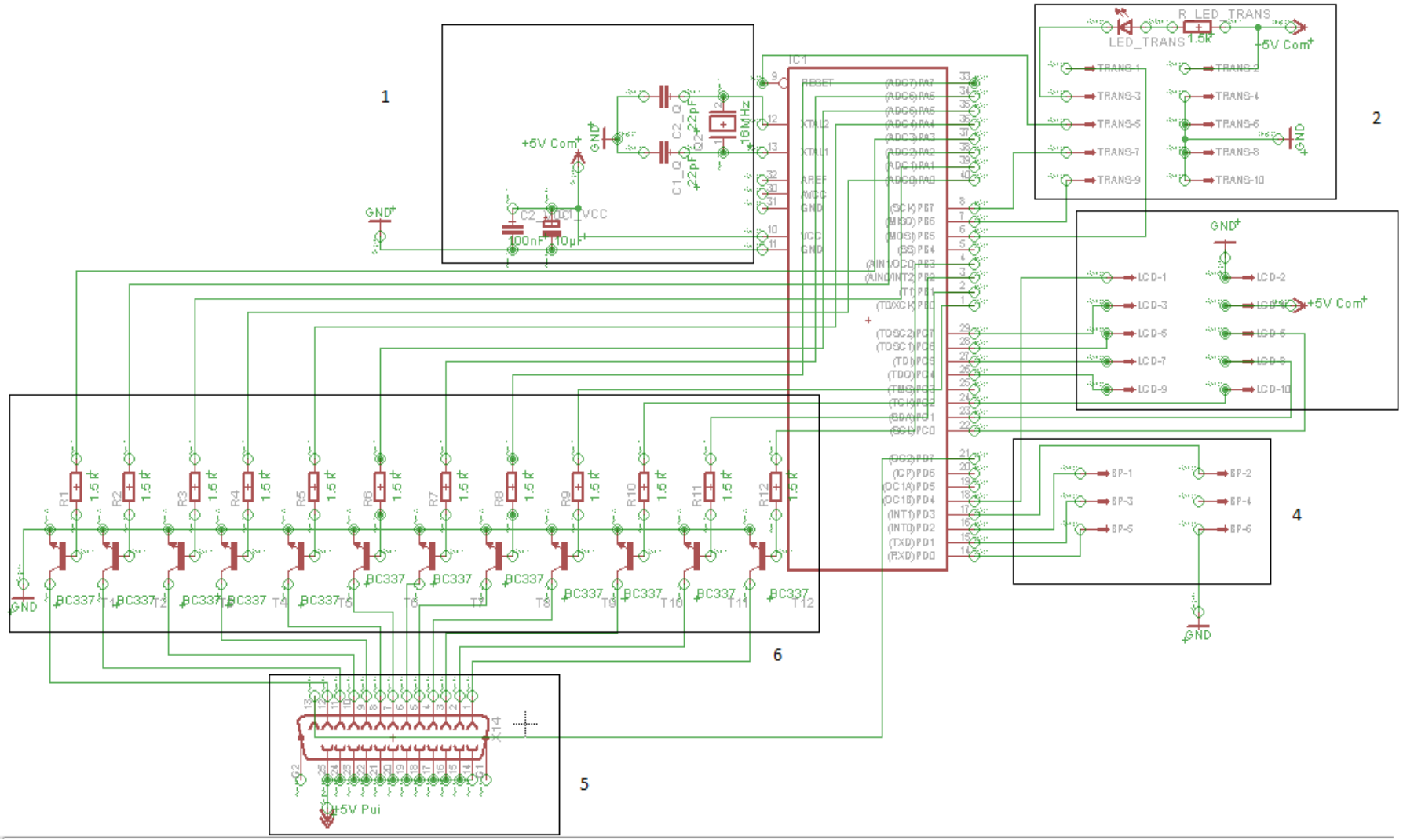

Illustration 24: Schéma électrique de la carte principale [1]

Sur le schéma (page précédente), la première partie représente l'alimentation de l'ATméga (au centre), et le circuit de son horloge. Les quatre parties suivantes servent respectivement à la connexion avec la carte de programmation, la carte le l'écran LCD, la carte des boutons poussoirs, et enfin la lampe. La sixième partie est composée de douze transistors afin de commander les LEDs.

3.1.3. La partie écran LCD

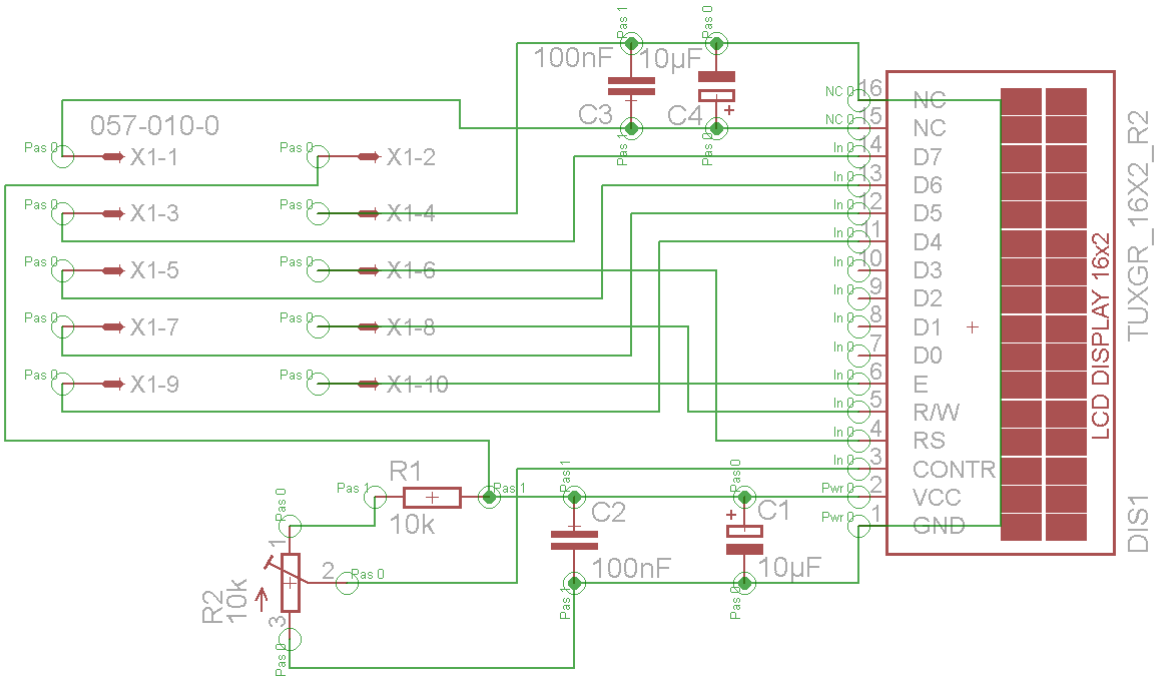

Illustration 25: Schéma électrique de l'écran LCD[1]

Le connecteur 10 broches sert à faire la liaison avec la carte principale.

Nous utilisons un montage diviseur de tension à l'aide d'un potentiomètre et d'une résistance fixe permettant ainsi le réglage du contraste de l'écran LCD.

Les condensateurs sont là pour filtrer la tension, car il est possible d'avoir des interférences magnétiques à cause de la longueur des pistes et de la nappe.

Nous avons choisi de séparer l'alimentation de l'écran LCD et l'alimentation du rétro-éclairage afin de pouvoir éteindre ce dernier si on le souhaite.

3.1.4. Les typons

Typon de la carte de l'écran LCD

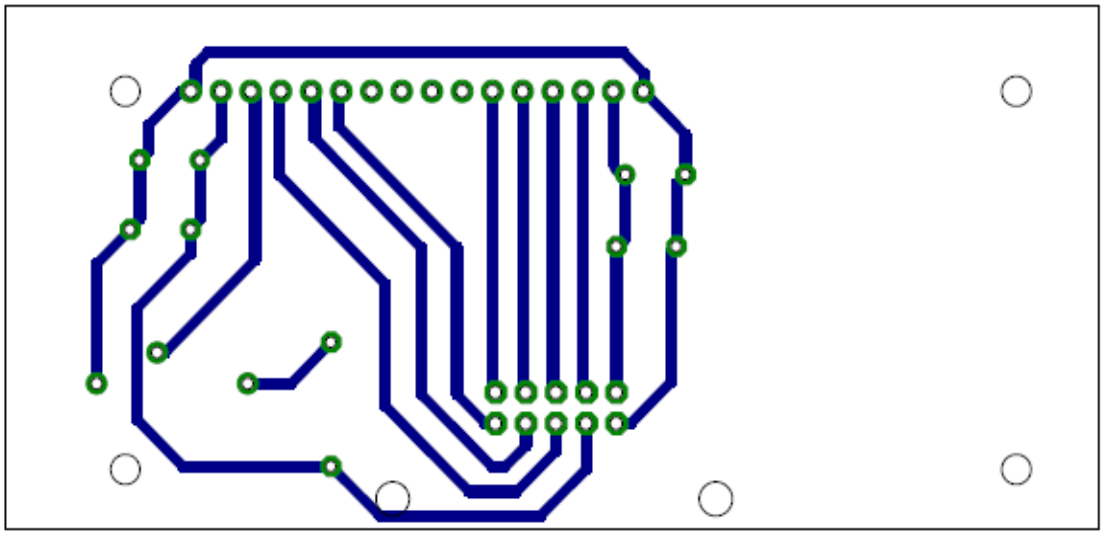

Illustration 26: Typon de la carte écran LCD coté pistes[1]

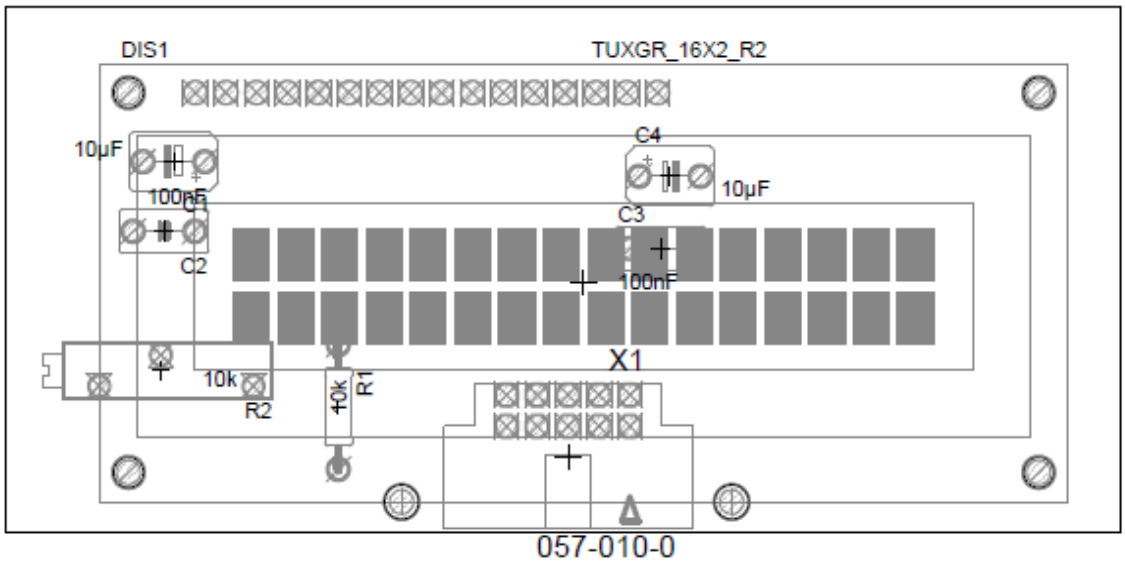

Illustration 27: Typon de la carte écran LCD coté composants[1]

Typon de la carte des boutons poussoirs

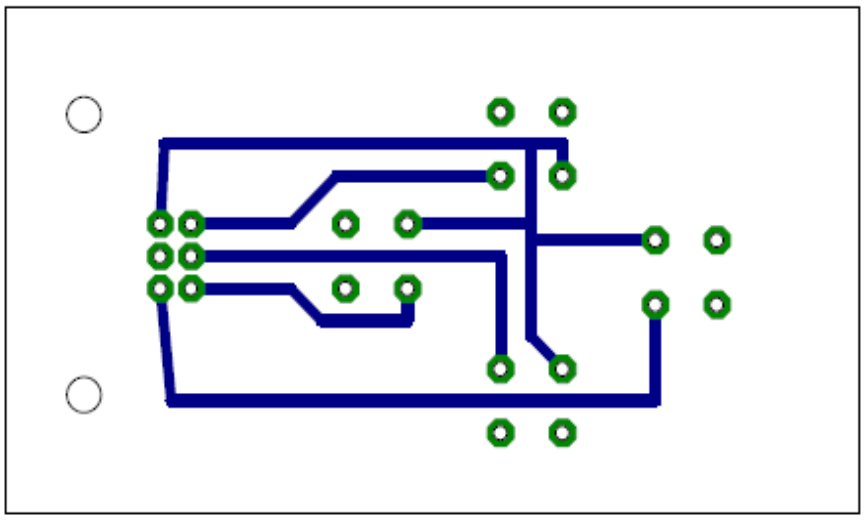

Illustration 28: Typon de la carte boutons poussoirs coté pistes[1]

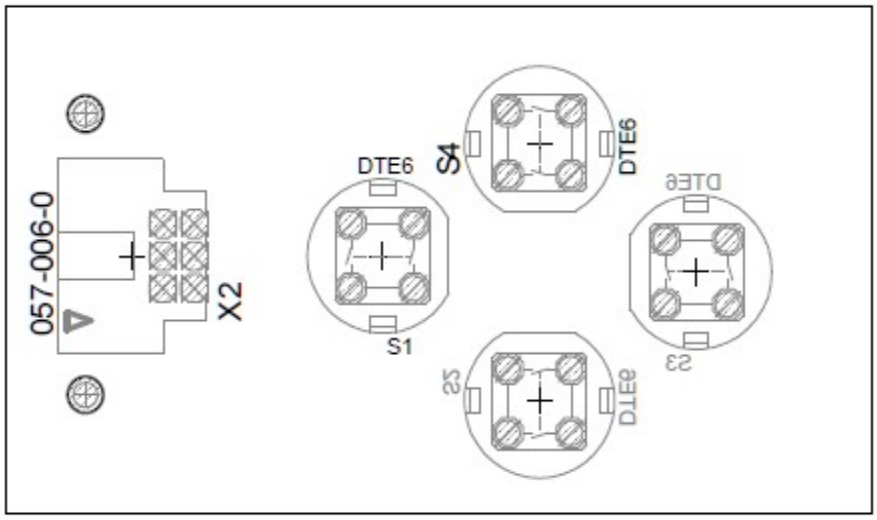

Illustration 29: Typon de la carte boutons poussoirs coté composants[1]

Typons de la carte principale vue du dessous

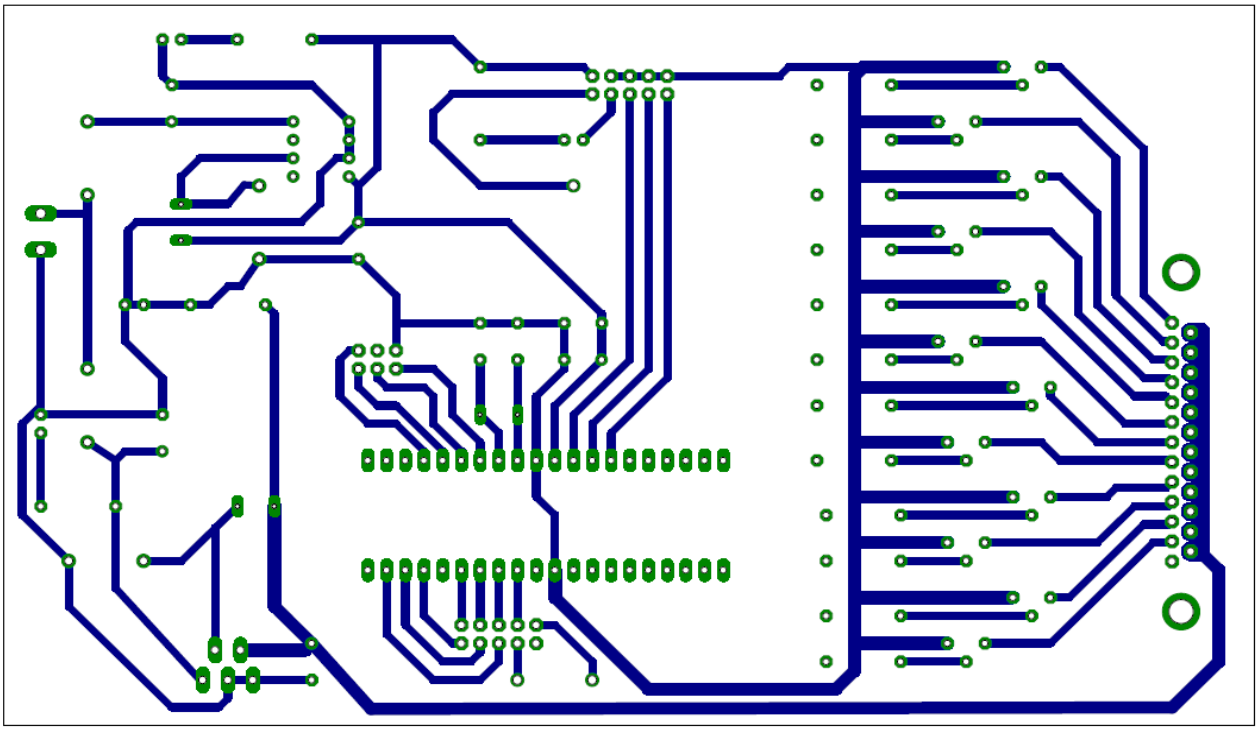

Illustration 31: Typons de la carte principale vue du dessous [1]

Typons de la carte principale vue du dessus

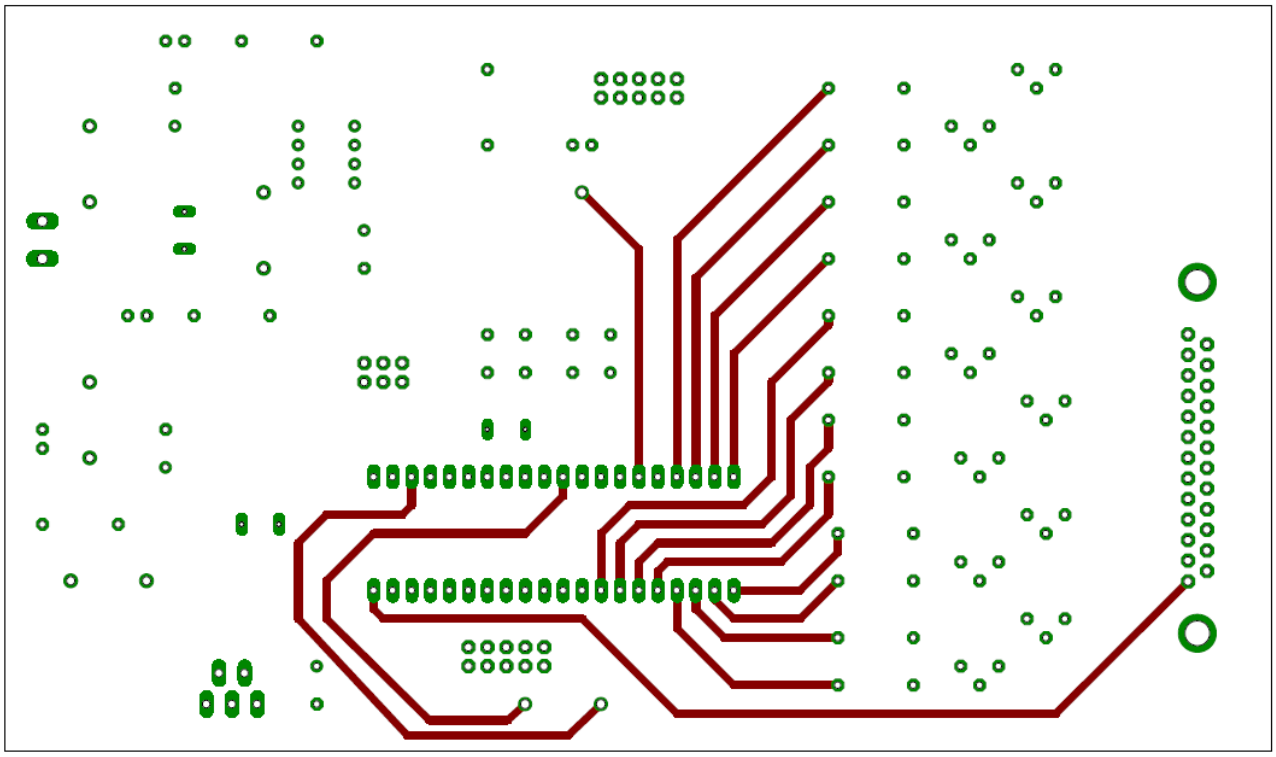

Illustration 30: Typons de la carte principale vue du dessus [1]

Typons de la carte principale coté composants

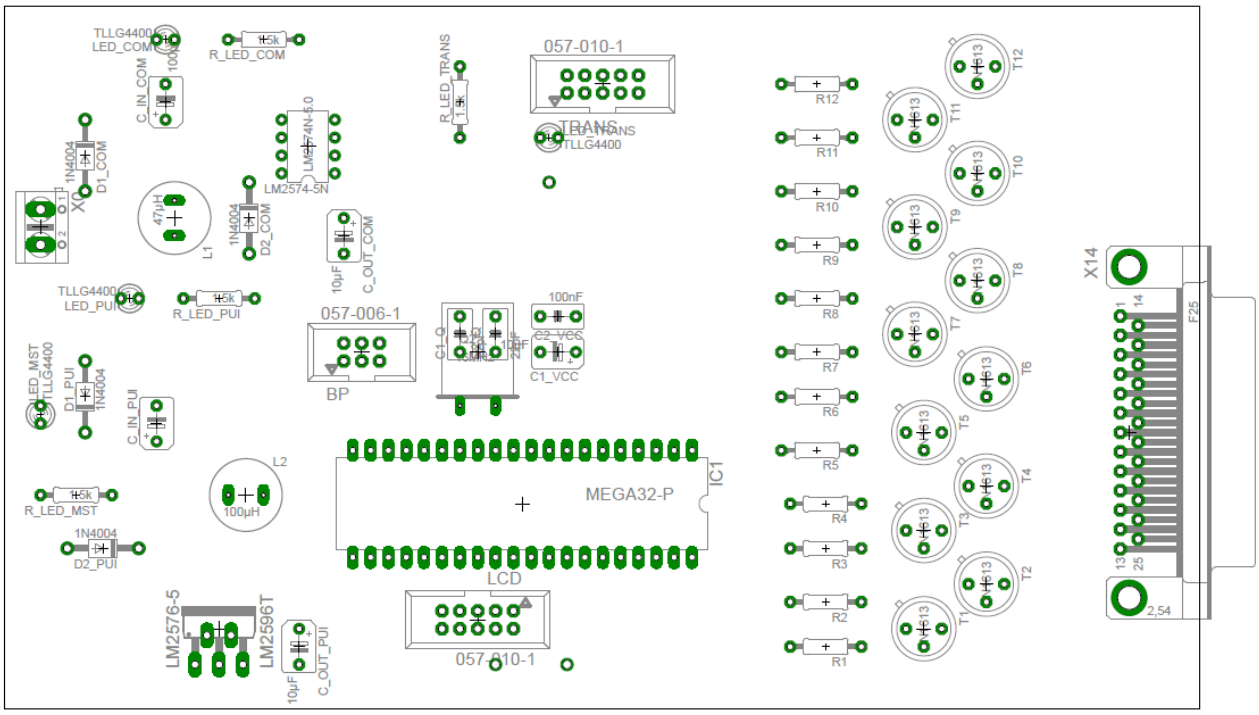

Illustration 32: Typons de la carte principale coté composant [1]

3.2. Programmation

L'intégralité du programme est disponible en annexe 1

3.2.1. Initialisation de l'horloge et des ports d'entrées et de sorties

Afin d'aérer notre programme, nous avons mis les paramètres d'initialisation dans une bibliothèque « Header.c » que nous avons ensuite ajoutée à notre projet. Nous avons pu ainsi créer une fonction « *Init()* » que nous appelons simplement en début de programme.

```
void Init(void)

{

// 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=0xFF;

// 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=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=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In

// State7=0 State6=0 State5=0 State4=0 State3=P State2=P State1=P State0=P

PORTD=0x0F;

DDRD=0xF0;


// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: Timer 0 Stopped

TCCR0=0x00;

TCNT0=0x00;
```

```
// Mode: Normal top=FFh
// OC0 output: Disconnected
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: CTC top=OCR2
```

```
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x0C;
TCNT2=0x00;
OCR2=0xF9;                  //Interruption réalisée toutes les 1 ms

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

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;

// 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")


    //Déclaration de l'écran LCD
      #asm
        .equ __lcd_port = 0x15;
      #endasm
}
```

Nous déclarons ici toutes les entrées/sorties, ainsi que l'interruption interne pour qu'elle s'active toute les 1ms.

3.2.2. Programmation de l'horloge

En utilisant l'interruption interne du timer, nous avons décidé de réaliser une horloge qui indiquera à l'utilisateur depuis combien de temps la fonction actuelle est active.

```
  //Horloge en seconde
    MSec++;
    if(MSec >= 1000)
    {
        Seconde++;
        MSec = 0;
        if(Seconde>=60)
        {
            Minute++;
            Seconde = 0;
        }
    }
```

3.2.3. Programmation de la M.L.I.

La MLI était la partie la plus complexe du programme.

Toute les fonctions de MLI étant basées sur le même principe, nous en expliquerons juste une.

```
if(Variable >= 100)
{
        if(FlagUpDown == 0)         //MLI s'incrémente ou se décrémente
                FlagMLI++;
        else
                FlagMLI--;

        if(FlagMLI >= 10)           //Quand MLI est au max, on décrémente
                        FlagUpDown = 1;

                if(FlagMLI < 0)     //Quand MLI au minimum, on passe à la LED suivante
                {
                        FlagUpDown = 0;
                        if(FlagSens == 0)     //Choix de la led en fonction du sens
                                FlagSwch++;
                        else
                                FlagSwch--;
                }
                Variable = 0;
        }
        switch(FlagSwch)      //Choix des  LEDs activées
        {
                case 0: if(Temps <=FlagMLI)
                        {
                                Jaune1 = 1;
                                Jaune2 = 1;
                                Jaune3 = 1;
                                Jaune4 = 1;
```

```
        }
        else
        {
                Jaune1 = 0;
                Jaune2 = 0;
                Jaune3 = 0;
                Jaune4 = 0;
        }
        break;
case 1: if(Temps <=FlagMLI)
        {
                Vert1 = 1;
                Vert2 = 1;
                Vert3 = 1;
                Vert4 = 1;
        }
        else
        {
                Vert1 = 0;
                Vert2 = 0;
                Vert3 = 0;
                Vert4 = 0;
        }
        break;
case 2: if(Temps <=FlagMLI)
        {
                Rouge1 = 1;
                Rouge2 = 1;
                Rouge3 = 1;
```

```
                Rouge4 = 1;
            }
            else
            {
                Rouge1 = 0;
                Rouge2 = 0;
                Rouge3 = 0;
                Rouge4 = 0;
            }
            break;
        }
        if(FlagSwch >= 2)
            FlagSens = 1;

        if(FlagSwch <= 0)
            FlagSens = 0;
```

3.2.4. Programmation du menu

Nous avons réalisé un menu qui permet ainsi une navigation simple et agréable parmi les différents réglages et choix de fonctions. En revanche, étant composé de beaucoup d'affichage, c'est également la partie qui demande le plus de mémoire dans le micro-contrôleur.

```
while(Menu != 0)
  {
     lcd_gotoxy(2,0);
    lcd_putsf("Bienvenue");
    lcd_gotoxy(2,1);
    lcd_putsf("Jeux de LED");
    lcd_gotoxy(0,2);
    lcd_putsf("Appuyer sur MENU");
```

```
    lcd_gotoxy(1,3);
    lcd_putsf("pour continuer");
  }
  while(1)
  {
     if(FlagMenu1 == 0)                    //Menu Principal
     {
        lcd_gotoxy(1,0);
        lcd_putsf("Informations    ");
        lcd_gotoxy(1,1);
        lcd_putsf("Fonctions       ");
        lcd_gotoxy(1,2);
        lcd_putsf("Test des LED    ");
        lcd_gotoxy(1,3);
        lcd_putsf("Retro-Eclairage ");
     }
     else if(FlagMenu1 == 1)               //Menu Informations
     {
        lcd_gotoxy(2,0);
        lcd_putsf("Jeux de LEDs");
        sprintf(tampon, "Fonction %d", Fonction);
        lcd_gotoxy(2,1);
        lcd_puts(tampon);
        sprintf(tampon, "T: %2d min ", Minute);
        lcd_gotoxy(1,3);
        lcd_puts(tampon);
        sprintf(tampon, "%2d s", Seconde);
        lcd_gotoxy(11,3);
        lcd_puts(tampon);
```

```
    }
    else if(FlagMenu1 == 2)              //Menu des fonctions
    {
       for(i=(FlagMenu*4)+1;i<(FlagMenu*4)+5;i++)
          {
          sprintf(tampon, "%s", MenuDep[i]);
          lcd_gotoxy(1,(i-(FlagMenu*4)-1));
          lcd_puts(tampon);
          }
    }
    else if(FlagMenu1 == 3)              //Menu de test des leds
    {
       lcd_gotoxy(1,2);
       lcd_putsf("Test en cours");
    }
  if(Menu == 0)                          //Appui sur le bouton Menu
     FlagMenu1 = 0;
  if(Haut == 0)                          //Appui sur le bouton Haut
  {
              j--;
              if(j<0)
              {
                        j=3;
                        if(FlagMenu1 == 2)
                                  FlagMenu--;
              }
              if(FlagMenu<0)
                        FlagMenu = NBR_MENU;
              delay_ms(200);
```

```
}
 if(Bas == 0)                          //Appui sur le bouton Bas
 {
            j++;
            if(j>3)
            {
                    j=0;
                    if(FlagMenu1 == 2)
                            FlagMenu++;
            }
            if(FlagMenu > NBR_MENU)
                    FlagMenu = 0;
            delay_ms(200);
 }
 if((FlagMenu1 == 0)||(FlagMenu1 == 2))      //Affichage de la flèche de choix de menu
 {
   for(i=0;i<4;i++)
   {
        if(i != j)
        {
                lcd_gotoxy(0,i);
                lcd_putchar(' ');
        }
        else
        {
                lcd_gotoxy(0,i);
                lcd_putchar('>');
        }
   }
```

```
    }
     if(Ok == 0)                    //Appui sur le bouton Ok
     {
        lcd_clear();
        if(FlagMenu1 == 0)
           FlagMenu1 = j + 1;
        if(FlagMenu1 == 2)
        {
           Fonction = j + (FlagMenu*4) + 1;
           Variable = 0;
           FlagSwch = 0;
           FlagFirst = 0;
           Minute = 0;
           Seconde = 0;
        }
        if(FlagMenu1 == 3)              //Activation du test des Leds
        {
           Fonction = -1;
           Variable = 0;
        }
        if(FlagMenu1 == 4)
        {
           FlagMenu1 = 0;
           RELCD = ~RELCD;              //Rétro-Eclairage de l'écran LCD
        }
     }
  }
```

L'affichage du temps dans le menu informations est séparé en deux, car le micro-contrôleur n'acceptait pas d'avoir à afficher les minutes et les secondes dans la même ligne, lorsqu'on la faisait, le micro-contrôleur redémarré.

3.3. Problèmes rencontrés

Lors de notre projet, nous avons rencontré quelques problèmes mineurs qui nous ont pris plus ou moins de temps à résoudre, notamment un problème de composants demandés, à cause de mauvais choix par le magasinier.

Nous avons également rencontrés un problème lors de la programmation: dans la fonction interruption du timer, dans le « switch », à cause de la présence du « default », qui faisait buggé l'affichage du menu. L'interruption est activé toutes les 1 ms, un affichage sur l'écran LCD dure plus longtemps d'1 ms et avait donc pour effet de provoquer une erreur dans le programme.

Nous avons perdu du temps lors des premières séances avec la programmation de la MLI, car nous avons décider de l'effectuer de manière logicielle, et non matérielle.

3.4. Améliorations possibles

S'agissant d'un éclairage lumineux, il y a énormément de possibilité d'améliorations. Il nous est possible par exemple de rajouter un potentiomètre à pied ce qui ferait en sorte que les led ne dépassent pas une certaines intensité lumineuse.

Ou tout simplement de rajouter de nouvelle fonction en combinant adroitement les couleurs et leurs effets.

Conclusion

Lors de ce projet, nous avons mis en application toutes les connaissances acquises durant nos 2 ans à l'IUT, pendant nos cours d'informatique industrielle. Il nous a également fallu bien comprendre le principe de la MLI afin de pouvoir la mettre en pratique dans notre projet.

Le projet a été débuté par Corentin SAPIENS en décembre 2010 qui souhaitait mettre un peu d'ambiance chez lui lors de soirée. C'est lui qui a effectué les calculs nécessaires aux premiers choix des composants, ainsi que le premier typon de la carte principale.

Ce projet nécessite une compréhension sur le fonctionnement de chaque composant afin de pouvoir les « câbler » et programmer. L'étude des schémas électriques, la réalisation des typons et des cartes et enfin, la partie programmation de la MLI en utilisant la logiciel Code Vision AVR. Le logiciel ne nous a pas posé beaucoup de problème car un membre du binôme avait déjà réalisé son projet de semestre 3 à l'aide de ce logiciel.

Ce projet nous a permis de mettre en application de façon concrète et utile ce que nous avons appris au cours de nos 2 ans. C'est un sujet qu'il nous est possible de devoir réaliser quelque soit l'entreprise dans laquelle nous travaillerons plus tard. Ce projet représente donc pour nous, une véritable expérience professionnelle.

Résumé

La modulation de largeur d'impulsions est une technique qui permet de donner l'illusion de variation de tension sur un composant. Dans notre cas, elle permet de faire varier l'intensité lumineuse de plusieurs LED, et nous permettra ainsi, de créer des effets lumineux

Le but de notre projet a été de créer une lampe d'ambiance qui pourrait produire des effets de couleurs, rythmiques et lumineux. Pour cela, nous avons dans un premier temps, récupérés le travail de Corentin SAPIENS ainsi que son premier prototype pour analyse. Nos séances ont alors eu pour but: de faire fonctionner une horloge, réaliser de la MLI sur des LEDs et naviguer à travers un menu.

Tous les composants électroniques ont alors été répartis sur 3 cartes afin de pouvoir rentrer le tout dans un boitier et que cela soit esthétique et sûr pour les utilisateurs.

Une fois toute les cartes réalisées et placées dans le boitier, nous avons pu écrire nos propre programmes mettant en œuvre la MLI. A ce jour, la carte finale est réalisée et placée dans son boitier, et les programmes presque terminés. La lampe pourra alors être placé dans une chambre ou un appartement afin de créer une ambiance lumineuse agréable en fonction de l'utilisateur.

203 mots

Index des illustrations

Illustration 1: Exemple de lampe d'ambiance [1].....5
Illustration 2: Boitier [1].....5
Illustration 3: Oscillogramme d'une MLI [1].....6
Illustration 4: MLI à 10% [2].....6
Illustration 5: MLI à 90% [2].....6
Illustration 6: Planning prévisionnel et réel.....8
Illustration 7: Schéma fonctionnel de niveau 1 [1].....9
Illustration 8: Schéma fonctionnel de niveau 2 [3].....9
Illustration 9: Ordinogramme du "main()" [1].....10
Illustration 10: Ordinogramme des interruptions[1].....11
Illustration 11: Orgamigramme de la MLI[1].....11
Illustration 12: ATMEGA 8535[1].....14
Illustration 13: Régulateur de tension LM2576 [1].....14
Illustration 14: Régulateur de tension LM2574N-5 [1].....14
Illustration 15: Ecran LCD [2].....15
Illustration 16: CodeVision Initialisation 1 [1].....16
Illustration 17: CodeVision Initialisation 2 [1].....16
Illustration 18: CodeVision Initialisation 3 [1].....17
Illustration 19: CodeVision Initialisation 4 [1].....17
Illustration 20: CodeVision Initialisation 5 [1].....18
Illustration 21: CodeVision Initialisation 6 [1].....18
Illustration 22: CodeVision Initialisation 7 [1].....19
Illustration 23: Schéma électrique de la partie alimentation[1].....20
Illustration 24: Schéma électrique de la carte principale [1].....21
Illustration 25: Schéma électrique de l'écran LCD[1].....22
Illustration 26: Typon de la carte écran LCD coté pistes[1].....23
Illustration 27: Typon de la carte écran LCD coté composants[1].....23
Illustration 28: Typon de la carte boutons poussoirs coté pistes[1].....24
Illustration 29: Typon de la carte boutons poussoirs coté composants[1].....24
Illustration 30: Typons de la carte principale vue du dessus [1].....25
Illustration 31: Typons de la carte principale vue du dessous [1].....25
Illustration 32: Typons de la carte principale coté composant [1].....26
Illustration 33: Schéma du LM2574 69
Illustration 34: Schéma du LM2576.....69

Bibliographie

[1] **SAPIENS/PETITEAU**, "*Photos personnelles*", , 2011.
[2] **sonelec-musique.com**. , , [En ligne]. (Page consultée le 31 Mars 2011) <http://www.sonelec-musique.com/electronique_bases_modulation_largeur_impulsion.html>
[3] **LEQUEU Thierry**. *Schéma fonctionnel*, , [En ligne]. (Page consultée le 24 mars 2011) <http://www.thierry-lequeu.fr/data/DATA450.HTM>
[4] **ATMEL**. , , [En ligne]. (Page consultée le 31 mars 2011) <http://www.atmel.com/dyn/resources/prod_documents/2502s.pdf>

Annexes

Annexe 1 : Programme de l'ATméga8535 main.c

Annexe 2 : Programme de l'ATméga8535 Header.c

Annexe 3 : Schéma du LM2574 et LM2576

Annexe 4 : Fiche de suivi de projet

Annexe 1: Programme de l'ATméga 8535 main.c

```
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#include <lcd.h>
#include "Header.c"

#define Bas PIND.0
#define Haut PIND.1
#define Menu PIND.2
#define Ok PIND.3
#define RELCD PIND.4
#define NBR_MENU 1
#define Jaune1 PORTA.0
#define Jaune2 PORTA.1
#define Jaune3 PORTA.2
#define Jaune4 PORTA.3
#define Vert1 PORTA.4
#define Vert2 PORTA.5
#define Vert3 PORTA.6
#define Vert4 PORTA.7
#define Rouge1 PORTB.3
#define Rouge2 PORTB.2
#define Rouge3 PORTB.1
#define Rouge4 PORTB.0

void FonctionMLI(void);
void FonctionMLIChen(void);
void FonctionMLIDouble(void);
```

```
unsigned char Temps=0, MSec=0, Seconde=0, Minute=0;
int Variable=0, FlagMLI=1,FlagMenu = 0;
char FlagUpDown=0, FlagSens=0, FlagSwch=0, Fonction=0, FlagFirst, FlagChenille=0;
char FlagMenu1=0;
unsigned char tampon[15];
unsigned char MenuDep[9][16];
int i, j=0;

// Timer 0 output compare interrupt service routine toute les 1ms
interrupt [TIM2_COMP] void timer0_comp_isr(void)
{
  Variable++;
  Temps++;
  if(Temps >= 10)
     Temps = 0;
  //Horloge en minutes/seconde
        MSec++;
        if(MSec >= 1000)
        {
              Seconde++;
              MSec = 0;
              if(Seconde>=60)
              {
                    Minute++;
                    Seconde = 0;
              }
        }
  //Choix de la fonction
  switch(Fonction)
```

```
{
        case -1:if(Variable<=3000)          //Test
        {
           PORTA = 0x2F;
                PORTB = 0x0F;
           PORTD = 0x7F;
        }
        else
        {
           PORTA = 0x00;
                PORTB = 0x00;
                PORTD = 0x0F;
           FlagMenu1 = 0;
        }
                break;
    case 1:   PORTA = 0x0F;             //Jaune
                PORTB = 0x00;
        PORTD = 0x0F;
        break;
    case 2: PORTA = 0x00;               //Rouge
                PORTB = 0x0F;
        PORTD = 0x0F;
        break;
    case 3:   PORTA = 0x20;             //Vert
                PORTB = 0x00;
        PORTD = 0x7F;
        break;
    case 4:   PORTA = 0x2F;             //Tout
                PORTB = 0x0F;
```

```
        PORTD = 0x7F;
        break;
    case 5: if(Variable >= 1000)          //Chenillard
        {
            FlagChenille++;
            Variable = 0;
        }
        if(FlagChenille >= 3)
            FlagChenille = 0;
        switch(FlagChenille)
        {
            case 0:     PORTA = 0x0F;
                        PORTB = 0x00;
                    PORTD = 0x0F;
                    break;
            case 1:     PORTA = 0x20;
                        PORTB = 0x00;
                    PORTD = 0x7F;
                    break;
            case 2:     PORTA = 0x00;
                        PORTB = 0x0F;
                    PORTD = 0x0F;
                    break;
        }
        break;
    case 6:  FonctionMLIChen();           //Chenillard MLI
                break;
    case 7: FonctionMLI();              //MLI
                break;
```

```
        case 8: FonctionMLIDouble();          //MLI Double
            break;
            }
}

void main(void)
{
   Init();          //Initialisation des ports d'entrées/sorties, des interruptions et de l'écran LCD

   sprintf(MenuDep[0], "Menu :     ");
   sprintf(MenuDep[1], "Jaune      ");
   sprintf(MenuDep[2], "Rouge      ");
   sprintf(MenuDep[3], "Vert       ");
   sprintf(MenuDep[4], "Complet    ");
   sprintf(MenuDep[5], "Chenillard ");
   sprintf(MenuDep[6], "Chen. MLI  ");
   sprintf(MenuDep[7], "MLI        ");
   sprintf(MenuDep[8], "MLI Double ");

   while(Menu != 0)
   {
        lcd_gotoxy(2,0);
      lcd_putsf("Bienvenue");
      lcd_gotoxy(2,1);
      lcd_putsf("Jeux de LED");
      lcd_gotoxy(0,2);
      lcd_putsf("Appuyer sur MENU");
      lcd_gotoxy(1,3);
      lcd_putsf("pour continuer");
```

```
}

while(1)
{
    if(FlagMenu1 == 0)
    {
        lcd_gotoxy(1,0);
        lcd_putsf("Informations    ");
        lcd_gotoxy(1,1);
        lcd_putsf("Fonctions       ");
        lcd_gotoxy(1,2);
        lcd_putsf("Test des LED    ");
        lcd_gotoxy(1,3);
        lcd_putsf("Retro-Eclairage ");

    }
    else if(FlagMenu1 == 1)
    {
        lcd_gotoxy(2,0);
        lcd_putsf("Jeux de LEDs");
        sprintf(tampon, "Fonction %d", Fonction);
        lcd_gotoxy(2,1);
        lcd_puts(tampon);
        sprintf(tampon, "T: %2d min ", Minute);
        lcd_gotoxy(1,3);
        lcd_puts(tampon);
        sprintf(tampon, "%2d s", Seconde);
        lcd_gotoxy(11,3);
        lcd_puts(tampon);
```

```
    }
    else if(FlagMenu1 == 2)
    {
        for(i=(FlagMenu*4)+1;i<(FlagMenu*4)+5;i++)
            {
            sprintf(tampon, "%s", MenuDep[i]);
            lcd_gotoxy(1,(i-(FlagMenu*4)-1));
            lcd_puts(tampon);
            }
    }
    else if(FlagMenu1 == 3)
    {
        lcd_gotoxy(1,2);
        lcd_putsf("Test en cours");
    }
  if(Menu == 0)
     FlagMenu1 = 0;

  if(Haut == 0)
  {
            j--;
     if(j<0)
     {
        j=3;
        if(FlagMenu1 == 2)
            FlagMenu--;
     }
     if(FlagMenu<0)
        FlagMenu = NBR_MENU;
```

```
            delay_ms(200);
        }
        if(Bas == 0)
        {
            j++;
           if(j>3)
           {
              j=0;
               if(FlagMenu1 == 2)
                  FlagMenu++;
           }
           if(FlagMenu > NBR_MENU)
              FlagMenu = 0;
           delay_ms(200);
        }
        if((FlagMenu1 == 0)||(FlagMenu1 == 2))
        {
           for(i=0;i<4;i++)
           {
                if(i != j)
                {
                        lcd_gotoxy(0,i);
                        lcd_putchar(' ');
                }
                else
                {
                        lcd_gotoxy(0,i);
                        lcd_putchar('>');
                }
```

```
        }
      }
      if(Ok == 0)
      {
        lcd_clear();
        if(FlagMenu1 == 0)
          FlagMenu1 = j + 1;
        if(FlagMenu1 == 2)
        {
          Fonction = j + (FlagMenu*4) + 1;
          Variable = 0;
          FlagSwch = 0;
          FlagFirst = 0;
          Minute = 0;
          Seconde = 0;
        }
        if(FlagMenu1 == 3)
        {
          Fonction = -1;
          Variable = 0;
        }
        if(FlagMenu1 == 4)
        {
          FlagMenu1 = 0;
          RELCD = ~RELCD;     //Retro-Eclairage de l'écran LCD
        }
      }
    }
}
```

```
void FonctionMLI(void)
{
        if(FlagFirst == 0)
   {
      PORTA = 0x00;
           PORTB = 0x00;
           PORTD = 0x0F;
   }
   FlagFirst++;
        if(Variable >= 100)
        {
                if(FlagUpDown == 0)
                        FlagMLI++;
        else
              FlagMLI--;
                if(FlagMLI >= 10)
                        FlagUpDown = 1;
                if(FlagMLI < 0)
                {
                        FlagUpDown = 0;
                        if(FlagSens == 0)
                                FlagSwch++;
                        else
                                FlagSwch--;
                }
                Variable = 0;
         }
        switch(FlagSwch)
```

```
{
   case 0:       if(Temps <=FlagMLI)
            Jaune1 = 1;
   else
            Jaune1 = 0;
   break;
case 1: if(Temps <=FlagMLI)
            Jaune2 = 1;
   else
            Jaune2 = 0;
   break;
case 2: if(Temps <=FlagMLI)
            Jaune3 = 1;
else
            Jaune3 = 0;
break;
case 3: if(Temps <=FlagMLI)
            Jaune4 = 1;
else
            Jaune4 = 0;
break;
case 4: if(Temps <=FlagMLI)
            Vert1 = 1;
else
            Vert1 = 0;
break;
case 5: if(Temps <=FlagMLI)
            Vert2 = 1;
            else
```

```
        Vert2 = 0;
break;
case 6: if(Temps <=FlagMLI)
        Vert3 = 1;
else
        Vert3 = 0;
break;
case 7: if(Temps <=FlagMLI)
        Vert4 = 1;
        else
        Vert4 = 0;
break;
case 8: if(Temps <=FlagMLI)
        Rouge1 = 1;
else
        Rouge1 = 0;
break;
case 9: if(Temps <=FlagMLI)
        Rouge2 = 1;
else
        Rouge2 = 0;
break;
case 10:if(Temps <=FlagMLI)
        Rouge3 = 1;
else
        Rouge3 = 0;
break;
case 11:if(Temps <=FlagMLI)
        Rouge4 = 1;
```

```
                else
                        Rouge4 = 0;
                break;
        }
        if(FlagSwch >= 11)
           FlagSens = 1;
        if(FlagSwch <= 0)
           FlagSens = 0;
}

void FonctionMLIChen(void)
{
   if(FlagFirst == 0)
   {
      PORTA = 0x00;
           PORTB = 0x00;
           PORTD = 0x0F;
   }
   FlagFirst++;

        if(Variable >= 100)
        {
                if(FlagUpDown == 0)
                        FlagMLI++;
        else
              FlagMLI--;

                if(FlagMLI >= 10)
                        FlagUpDown = 1;
```

```
            if(FlagMLI < 0)
            {
                  FlagUpDown = 0;
                  if(FlagSens == 0)
                        FlagSwch++;
                  else
                        FlagSwch--;
            }
            Variable = 0;
      }
      switch(FlagSwch)
{
            case 0:     if(Temps <=FlagMLI)
            {
                  Jaune1 = 1;
            Jaune2 = 1;
            Jaune3 = 1;
            Jaune4 = 1;
            }
            else
            {
                  Jaune1 = 0;
            Jaune2 = 0;
            Jaune3 = 0;
            Jaune4 = 0;
            }
            break;
            case 1: if(Temps <=FlagMLI)
```

```
{
        Vert1 = 1;
   Vert2 = 1;
   Vert3 = 1;
   Vert4 = 1;
}
  else
{
        Vert1 = 0;
   Vert2 = 0;
   Vert3 = 0;
   Vert4 = 0;
}
break;
case 2: if(Temps <=FlagMLI)
{
        Rouge1 = 1;
   Rouge2 = 1;
   Rouge3 = 1;
   Rouge4 = 1;
}
  else
{
        Rouge1 = 0;
   Rouge2 = 0;
   Rouge3 = 0;
   Rouge4 = 0;
}
break;
```

```
   }
        if(FlagSwch >= 2)
           FlagSens = 1;
        if(FlagSwch <= 0)
           FlagSens = 0;
}

void FonctionMLIDouble(void)
{
   if(FlagFirst == 0)
   {
           PORTA = 0x00;
           PORTB = 0x00;
           PORTD = 0x0F;
   }
   FlagFirst++;
        if(Variable >= 50)
        {
                if(FlagUpDown == 0)
                        FlagMLI++;
                else
                        FlagMLI--;
                if(FlagMLI >= 10)
                        FlagUpDown = 1;
                if(FlagMLI < 0)
                        FlagUpDown = 0;
        Variable = 0;
        }
        switch(FlagSwch)
```

```
{
   case 0:      if(Temps <=FlagMLI)
                    Jaune1 = 1;
            else
                    Jaune1 = 0;
            break;
   case 1:      if(Temps <=FlagMLI)
                    Rouge4 = 1;
            else
                    Rouge4 = 0;
            break;
   case 2:      if(Temps <=FlagMLI)
                    Jaune2 = 1;
            else
                    Jaune2 = 0;
            break;
        case 3: if(Temps <=FlagMLI)
                    Jaune1 = 1;
        else
                    Jaune1 = 0;
        break;
        case 4: if(Temps <=FlagMLI)
                    Jaune3 = 1;
        else
                    Jaune3 = 0;
        break;
        case 5: if(Temps <=FlagMLI)
                    Jaune2 = 1;
                    else
```

```
        Jaune2 = 0;
break;
case 6: if(Temps <=FlagMLI)
        Jaune4 = 1;
else
        Jaune4 = 0;
break;
case 7: if(Temps <=FlagMLI)
        Jaune3 = 1;
        else
        Jaune3 = 0;
break;
case 8: if(Temps <=FlagMLI)
        Vert1 = 1;
else
        Vert1 = 0;
break;
case 9: if(Temps <=FlagMLI)
        Jaune4 = 1;
else
        Rouge4 = 0;
break;
case 10:if(Temps <=FlagMLI)
        Vert2 = 1;
else
        Vert2 = 0;
break;
case 11:if(Temps <=FlagMLI)
        Vert1 = 1;
```

```
        else
                Vert1 = 0;
        break;
    case 12:if(Temps <=FlagMLI)
                Vert3 = 1;
          else
                Vert3 = 0;
          break;
    case 13:if(Temps <=FlagMLI)
                Vert2 = 1;
          else
                Vert2 = 0;
          break;
    case 14:if(Temps <=FlagMLI)
                Vert4 = 1;
          else
                Vert4 = 0;
          break;
        case 15:if(Temps <=FlagMLI)
                Vert3 = 1;
        else
                Vert3 = 0;
        break;
        //Jaune2 = 1;
        case 16:if(Temps <=FlagMLI)
                Rouge1 = 1;
        else
                Rouge1 = 0;
        break;
```

```
case 17:if(Temps <=FlagMLI)
        Vert4 = 1;
        else
        Vert4 = 0;
break;
case 18:if(Temps <=FlagMLI)
        Rouge2 = 1;
else
        Rouge2 = 0;
break;
case 19:if(Temps <=FlagMLI)
        Rouge1 = 1;
        else
        Rouge1 = 0;
break;
case 20:if(Temps <=FlagMLI)
   Rouge3 = 1;
else
        Rouge3 = 0;
break;
case 21:if(Temps <=FlagMLI)
        Rouge2 = 1;
else
        Rouge2 = 0;
break;
case 22:if(Temps <=FlagMLI)
        Rouge4 = 1;
else
        Rouge4 = 0;
```

```
                break;
                case 23:if(Temps <=FlagMLI)
                        Rouge3 = 1;
                else
                        Rouge3 = 0;
                break;
        }
        if(FlagSwch >= 23)
           FlagSwch = 0;
}
```

Annexe 1: Programme de l'ATméga 8535 Header.c

```
void Init(void)
{
// 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=0xFF;

// 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=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=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=P State2=P State1=P State0=P
PORTD=0x0F;
DDRD=0xF0;
```

```
// 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: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
```

```
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: CTC top=OCR2
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x0C;
TCNT2=0x00;
OCR2=0xF9;

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

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;

// 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")


//Déclaration de l'écran LCD
   #asm
      .equ __lcd_port = 0x15;
   #endasm



}
```

Annexe 3 : Schéma du LM2574 et LM2576

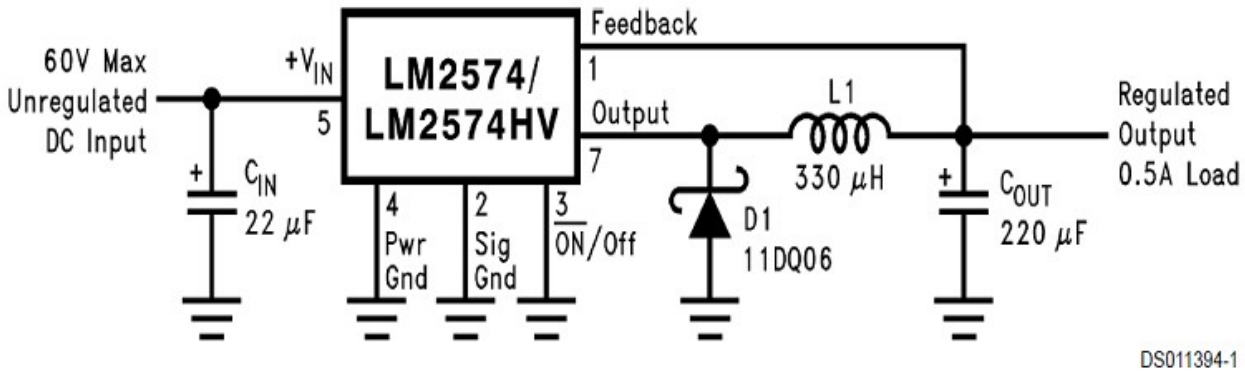

Illustration 33: Schéma du LM2574

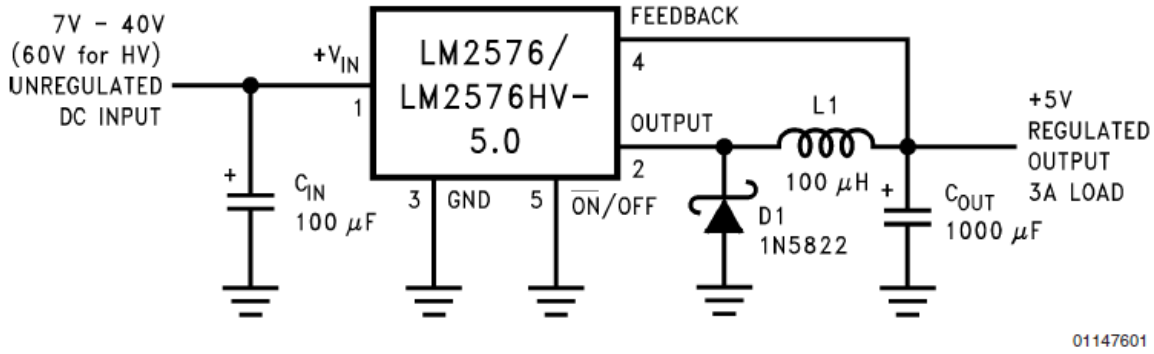

FIGURE 1.

Illustration 34: Schéma du LM2576

Annexe 4 : Fiche de suivi de projet

Semaine 5 (04/02/11) :

- découverte du sujet,
- réalisation du cahier des charges,
- réflexion d'amélioration .

Semaine 6 (11/02/11) :

- réalisation des cartes,
- récupération des composants,
- début de soudure.

Semaine 7 (18/02/11) :

- fin des cartes,
- fin de soudure, réalisation d'une carte LCD fiable,
- test de l'alimentation général de la carte et du micro-contrôleur,
- début des test par programmation.

Semaine 8 et 9 (vacances) :

- placement de la carte dans le boitier,
- continue des tests de programmation.

Semaine 10 (11/03/11) :

- réglage problème de « default » dans le switch du programme,
- changement de l'inductance d'alimentation de la partie puissance,
- programmation du menu,

◆ début du rapport.

Semaine 11 (15/03/11) :

◆ création de la lampe,

◆ rédaction du rapport,

◆ second changement de l'inductance d'alimentation de la partie puissance (le courant supporté par la première inductance était trop faible),

◆ essai d'ajout d'une autre page au menu.

Semaine 12 (25/03/11) :

◆ rédaction du rapport,

◆ réalisation de la carte finale.

◆ fin de la programmation

Semaine 13 (28/03/11) :

◆ réalisation de la lampe,

◆ remise du rapport écrit.

Semaine 14 (04/04/11) :

◆ présentation du rapport de projet à l'oral.