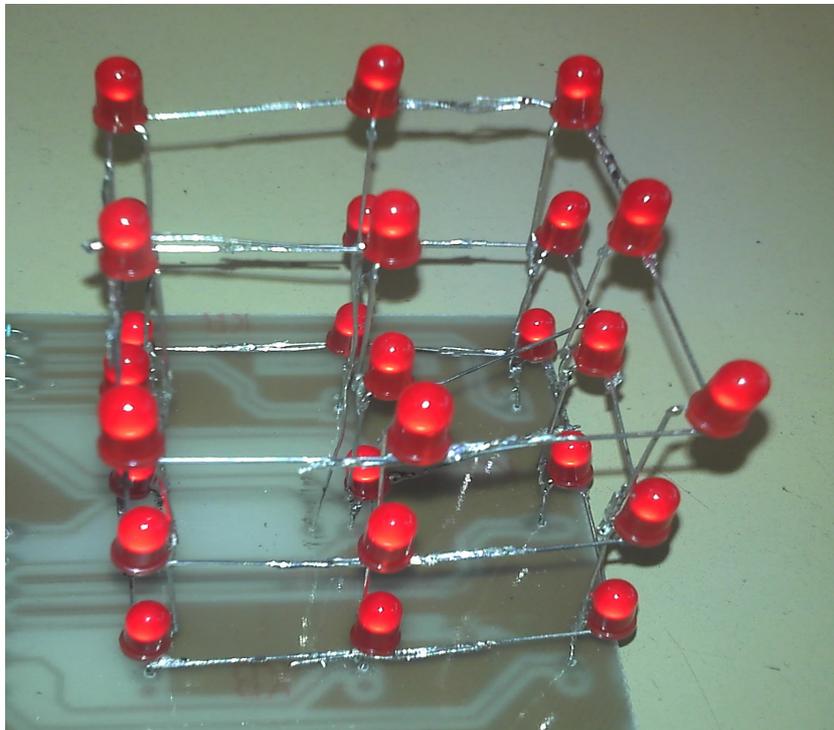


## CUBE DE LEDS 3x3x3





**CUBE DE LEDS 3x3x3**

# Sommaire

Introduction .....	5
1.Présentation du projet.....	6
1.1.Cahier des charges .....	6
1.2.Schéma fonctionnel.....	7
1.3.Planning prévisionnel .....	8
2.Étude théorique.....	8
2.1.Les LEDs.....	8
2.2.Atmega .....	9
2.3.L'alimentation 12/5V.....	9
3.Réalisation.....	10
3.1.La maquette.....	10
3.2.Le logiciel Orcad .....	10
3.3.Schéma capture.....	11
3.4.Layout Plus.....	14
4.Programmation.....	16
4.1.Logiciel.....	16
4.2.Programme de test .....	18
4.3.Programmation.....	19
4.4.Améliorations.....	25
5.Liste des composants.....	26
6.Planning final.....	27
Conclusion.....	28
Résumé.....	29
Index des illustrations.....	30
Bibliographie.....	31
Annexes.....	32

## Introduction

Au cours du quatrième semestre nous devons réaliser un projet d'étude et réalisation. Nous avons choisi notre propre sujet. Ce projet a pour but de créer un cube de LEDs en trois dimensions de dimension 3x3x3 qui nous permettra, selon la programmation, de faire plusieurs effets visuels.

Dans un premier temps nous étudierons des projets similaires existants puis nous essayerons d'améliorer le projet. Nous créerons une carte électronique afin de commander les différentes LEDs et nous réaliserons un programme permettant le fonctionnement voulu. Par la suite nous essayerons d'améliorer le projet en créant un programme nous permettant de programmer le cube plus facilement.

Dans ce document, nous vous présenterons le projet en entier. Nous vous parlerons ensuite de l'étude théorique que nous avons effectuée avant de passer sur la partie réalisation. Nous terminerons par une partie destinée à expliquer la programmation que nous avons faite.

# 1. Présentation du projet

Le projet consiste à créer une carte électronique permettant le contrôle de différentes LEDs placées en forme de cube afin d'obtenir plusieurs effets visuels désirés. Nous examinerons dans un premier temps les modèles de cube déjà existants puis nous créerons une carte électronique afin de commander les différentes LEDs avec le microcontrôleur choisi (Atmega 8535), puis nous apporterons des modifications afin de le rendre plus performant. Nous programmerons la carte avec le logiciel CodeVisionAVR afin de choisir l'effet visuel que nous désirons. Nous établirons en premier lieu un planning prévisionnel, qui nous permettra d'organiser notre travail et de voir l'avancement de celui-ci par rapport au temps fourni, et un cahier des charges afin de définir les objectifs et les contraintes du projet.

## 1.1. Cahier des charges

### ◆ Enjeu :

- Création d'un cube de LEDs 3x3x3.

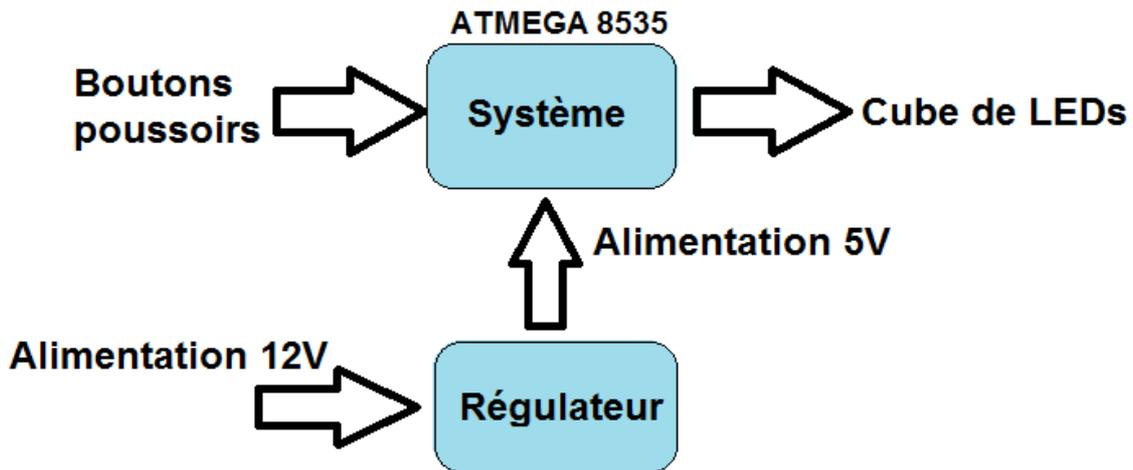
### ◆ Objectifs :

- Création de la carte électronique
- Programmation de l'Atmega 8535

### ◆ Contraintes :

- Alimentation de la carte 0/12V
- Alimentation du microcontrôleur 0/5V
- Réalisation de la carte électronique
- Programmation du microcontrôleur pour la gestion des LEDs

## 1.2. Schéma fonctionnel



*Illustration 1: Schéma fonctionnel [5]*

### 1.3. Planning prévisionnel

N° Semaine	5	6	7	8	9	10	11	12	13	14
Choix du sujet	Prévisionnel		Vacance	Vacance						
Cahier des charges	Prévisionnel		Vacance	Vacance						
Recherche de solutions		Réal	Vacance	Vacance	Prévisionnel					
Realisation du typon		Prévisionnel	Vacance	Vacance	Prévisionnel					
Programmation			Vacance	Vacance	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel		
Test			Vacance	Vacance			Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel
Prototype			Vacance	Vacance			Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel
Redaction synthese	Prévisionnel	Prévisionnel	Vacance	Vacance	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel
Remise du dossier			Vacance	Vacance					Prévisionnel	
Oral			Vacance	Vacance						Prévisionnel

	Prévisionnel		Vacance		Réal
--	--------------	--	---------	--	------

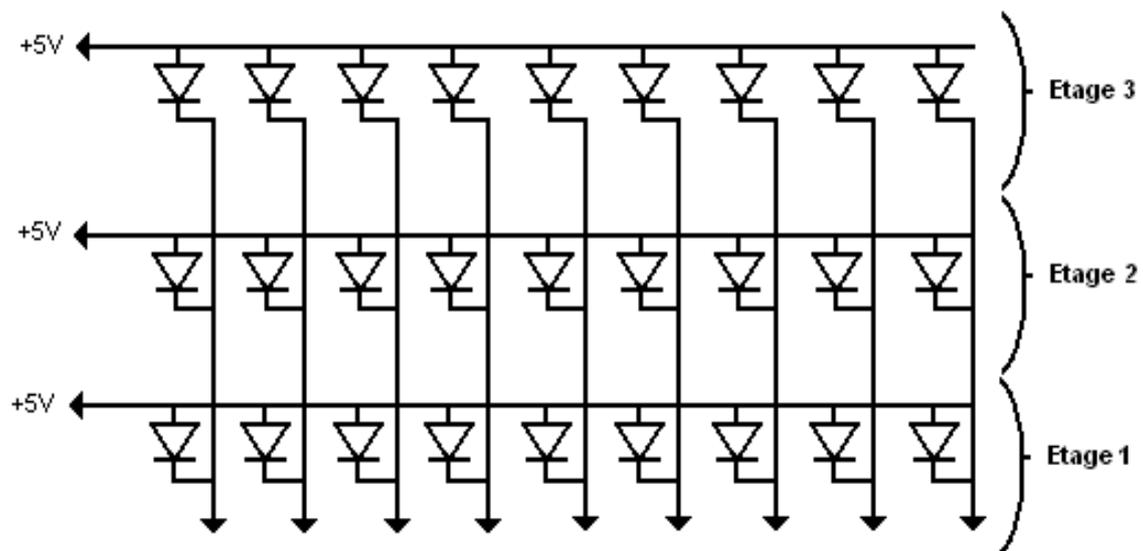
Illustration 2: Planning prévisionnel [5]

## 2. Étude théorique

### 2.1. Les LEDs

Nous avons choisi, pour créer notre cube de LEDs, des LEDs de 5mm de diamètre, de couleur rouge et de courant 20mA. Nous avons choisi, pour utiliser le moins possible de sortie de l'Atmega 8535, de relier les LEDs par leurs anodes<sup>1</sup> (positif) pour créer 3 étages et de les relier par leurs cathodes pour créer 9 colonnes (Voir schéma suivant).

<sup>1</sup> Une LED émet de la lumière lorsque le potentiel à l'anode est supérieur au potentiel de la cathode



*Illustration 3: Schéma électrique du cube [2][5]*

Ce câblage nous permet d'allumer la LED voulue en mettant l'étage de la LED à 5v et la colonne à 0v.

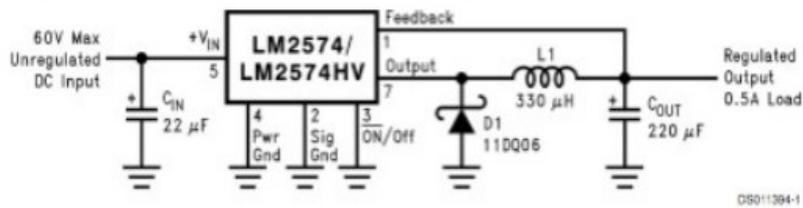
## 2.2. Atmega

Afin de commander les LEDs et le reste de la carte électronique nous utiliserons un microcontrôleur Atmega 8535. Ce microcontrôleur offre quatre ports de huit broches paramétrables soit en entrée, soit en sortie, selon notre souhait. Ce qui nous permet de commander notre carte facilement. Dans ce projet nous utiliserons neuf sorties pour les colonnes de LEDs, trois sorties pour les étages du cube et trois entrées pour des interrupteurs.

## 2.3. L'alimentation 12/5V

La carte électronique sera alimentée par un câble d'alimentation qui fournit une tension continue de 12V. Mais le microcontrôleur Atmega 8535 doit être alimenté en +5V ainsi que d'autres composants de la carte. C'est pourquoi nous réaliserons une alimentation à découpage basée sur un régulateur LM2574. Nous avons vu en cours de MC- ET2 comment réaliser cette alimentation à découpage.

## Typical Application (Fixed Output Voltage Versions)



Note: Pin numbers are for 8-pin DIP package.

Illustration 4: Schéma du régulateur LM2574 [3]

Ce montage permet d'obtenir une tension continue choisie (ici 5V) à partir d'une tension continue supérieure (ici 12V).

## 3. Réalisation

### 3.1. La maquette

Afin de commander les différentes LEDs nous avons à notre disposition notre carte fini permettant la commande des différentes entrées. Cette carte comporte :

- Un BP<sup>2</sup> d'alimentation générale ON/OFF
- Trois BPs permettant de choisir jusqu'à 7 effets visuels différent.

### 3.2. Le logiciel Orcad

#### 3.2.1. Capture

Le logiciel Orcad capture nous a permis de construire le schéma électrique à partir de l'étude théorique que nous avons étudiée précédemment. En utilisant ce schéma électrique, nous le transformons en un fichier qui s'appelle Netlist qui fait une liaison avec tous les composants que nous devons utiliser.

### 3.3. Schéma capture

#### 3.3.1. L'alimentation à découpage

Nous avons étudié dans le cours MC-ET2 que l'alimentation découpage de type Buck permet d'abaisser la tension d'entrée. Comme la commande de l'Atmega a besoin d'une tension continue de 5 Volts à partir de la tension continue de 12 Volts qui nous est fournie par un adaptateur, nous avons décidé d'utiliser ce type d'alimentation.

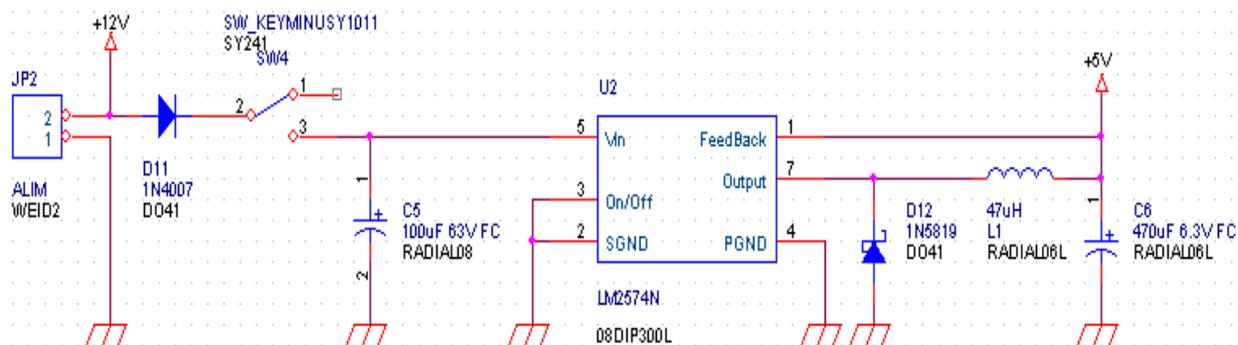


Illustration 5: Schématique de l'alimentation à découpage[3][5]

### 3.3.2. Atmega



Illustration 6: Photo de l'Atmega[1]

### 3.3.3. Assignation des pattes.

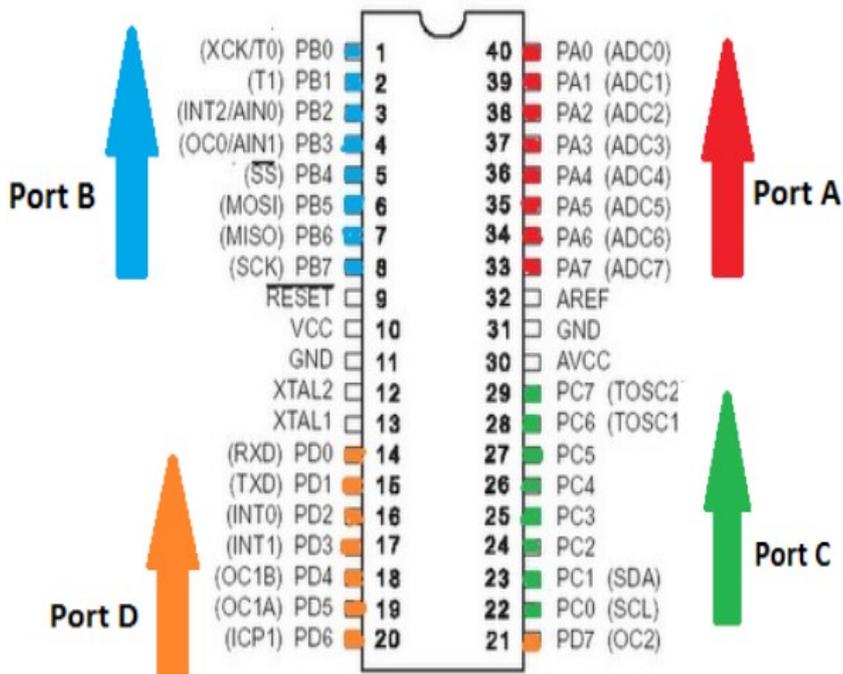


Illustration 7: Assignation des broches de l'Atmega[1]

On placera les sorties sur le port A et C et les entrées sur le port D.



- ◆ Trois Bps<sup>3</sup> à deux positions.

Ces trois Boutons poussoirs seront directement intégrés à la carte, nous n'avons donc pas besoin de connecteur.

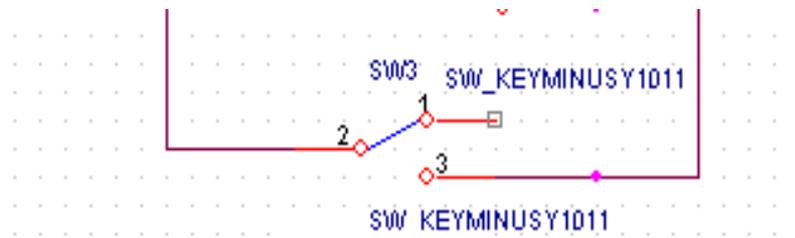


Illustration 9: Schématique d'un bouton poussoir [5]

## 3.4. Layout Plus

### 3.4.1. Typon

Après avoir terminé le schéma électrique sur Capture, nous avons créé la « Netlist » et nous l'avons envoyé vers *Orcad Layout Plus* pour réaliser le typon .

### 3.4.2. Contraintes

Pour réaliser le typon, nous avons pris le schéma électrique des précédents projets pour recréer l'alimentation à découpage ainsi que la partie horloge du microcontrôleur.

Dans cette étape, il faut bien prendre en compte la taille que l'on veut donner au cube qui sera placé directement sur la carte électronique et limiter la taille du reste de la carte pour que nous puissions éventuellement la mettre dans la boîte. Nous avons regroupé aussi chaque partie du projet pour pouvoir limiter la longueur de la piste afin d'éviter les problèmes de compatibilité électromagnétique.

De plus nous avons remarqué qu'il fallait placer le quartz plus près du microcontrôleur afin d'assurer le bon fonctionnement de l'Atmega 8535. Nous avons également étudié en cours de Physique (CEM) que le plan de masse est une bonne solution pour éviter le CEM<sup>4</sup>, cela explique l'utilisation d'un plan de masse dans notre carte.

---

<sup>3</sup> Bps : Boutons poussoirs

<sup>4</sup> CEM: Compatibilité électromagnétique.

### 3.4.3. Résultat final

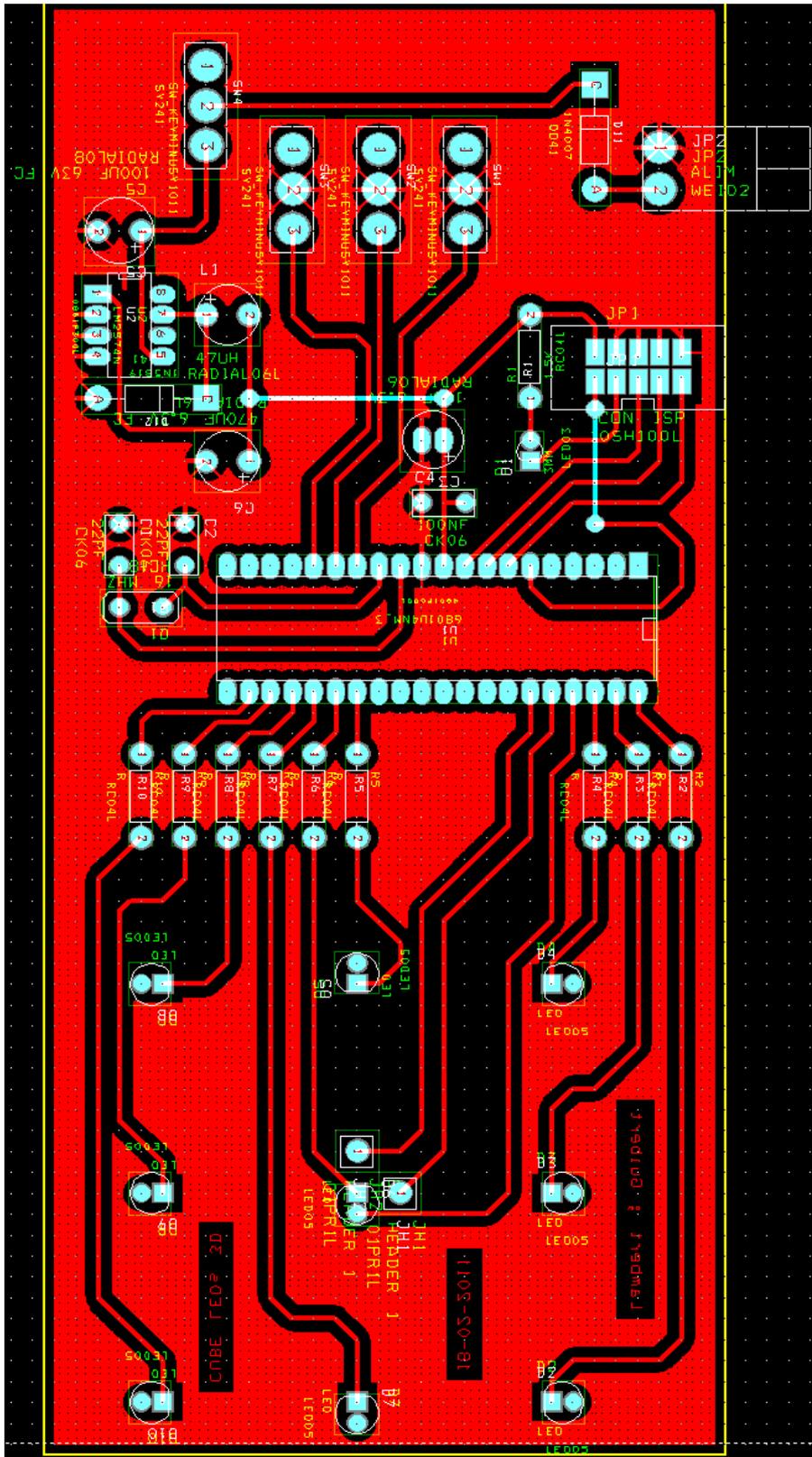
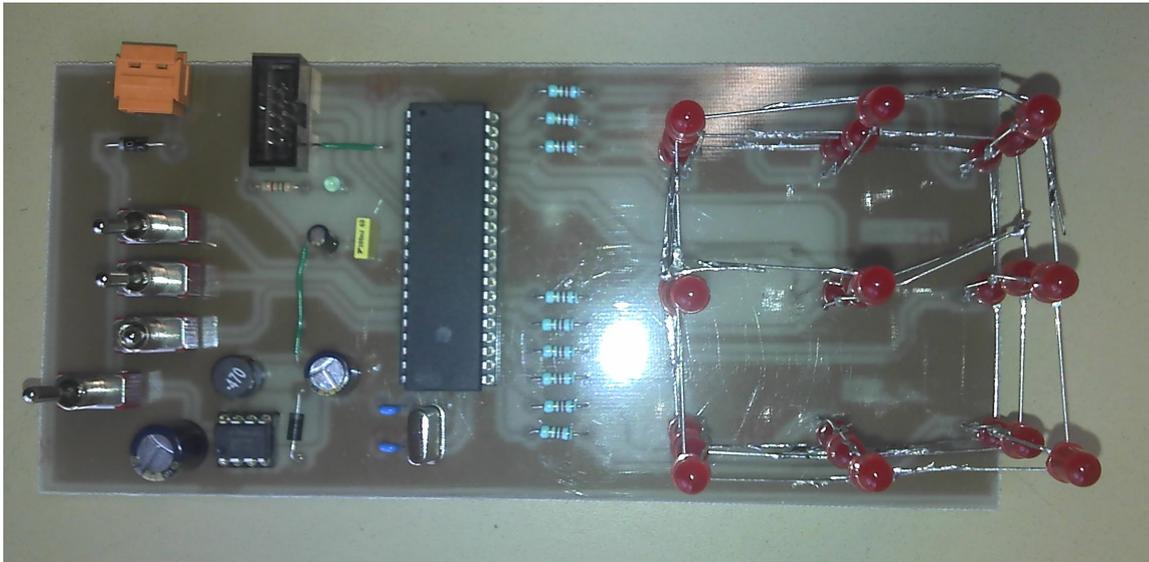


Illustration 10: Typon final



*Illustration 11: Photo de la carte [5]*

#### **3.4.4. Test de la carte**

Nous avons testé la carte afin de vérifier le fonctionnement de l'alimentation à découpage. Pour cela, nous avons alimenté la carte avec une tension de 12 volts sans la présence du LM2574N, afin de voir si nous avons bien 12 volts sur les broches du régulateur. Nous avons ensuite mis le régulateur de tension LM2574N et nous avons bien obtenu les 5 volts désirés à la sortie. Ce test nous a permis de confirmer le bon fonctionnement de la carte et de l'alimentation à découpage afin de pouvoir continuer notre projet.

## **4. Programmation**

### **4.1. Logiciel**

Pour la programmation de l'Atmega, nous avons utilisé le logiciel CodeVisionAVR. Pour programmer avec CodeVisionAVR, il faut tout d'abord créer un nouveau projet, le logiciel demande automatiquement un premier enregistrement. Ensuite, une page s'ouvre pour configurer le projet. Ici nous devons sélectionner l'Atmega 8535 dans l'onglet « chip » de la fenêtre C Compiler. Puis monter le clock à 16Mhz et dans l'onglet « (s)printf Features » sélectionner « float,width,precision ». Il faut ensuite cocher « program the chip » dans la fenêtre « AfterMake ». Afin de finaliser le projet, il faut créer une nouvelle source, la sauvegarder dans le même dossier

mais sous un autre nom et enfin l'ajouter au projet en cliquant sur « Project->Configure->Add »

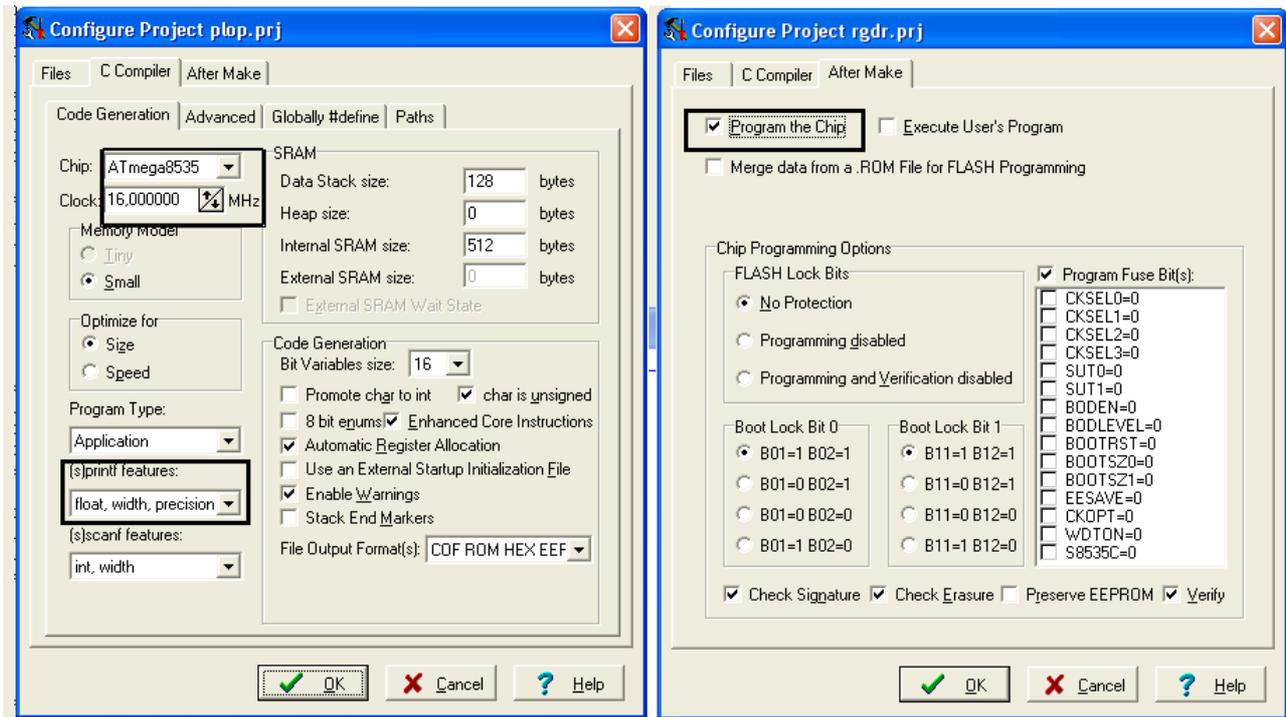


Illustration 12: Configuration de CodeVisionAVR[5]

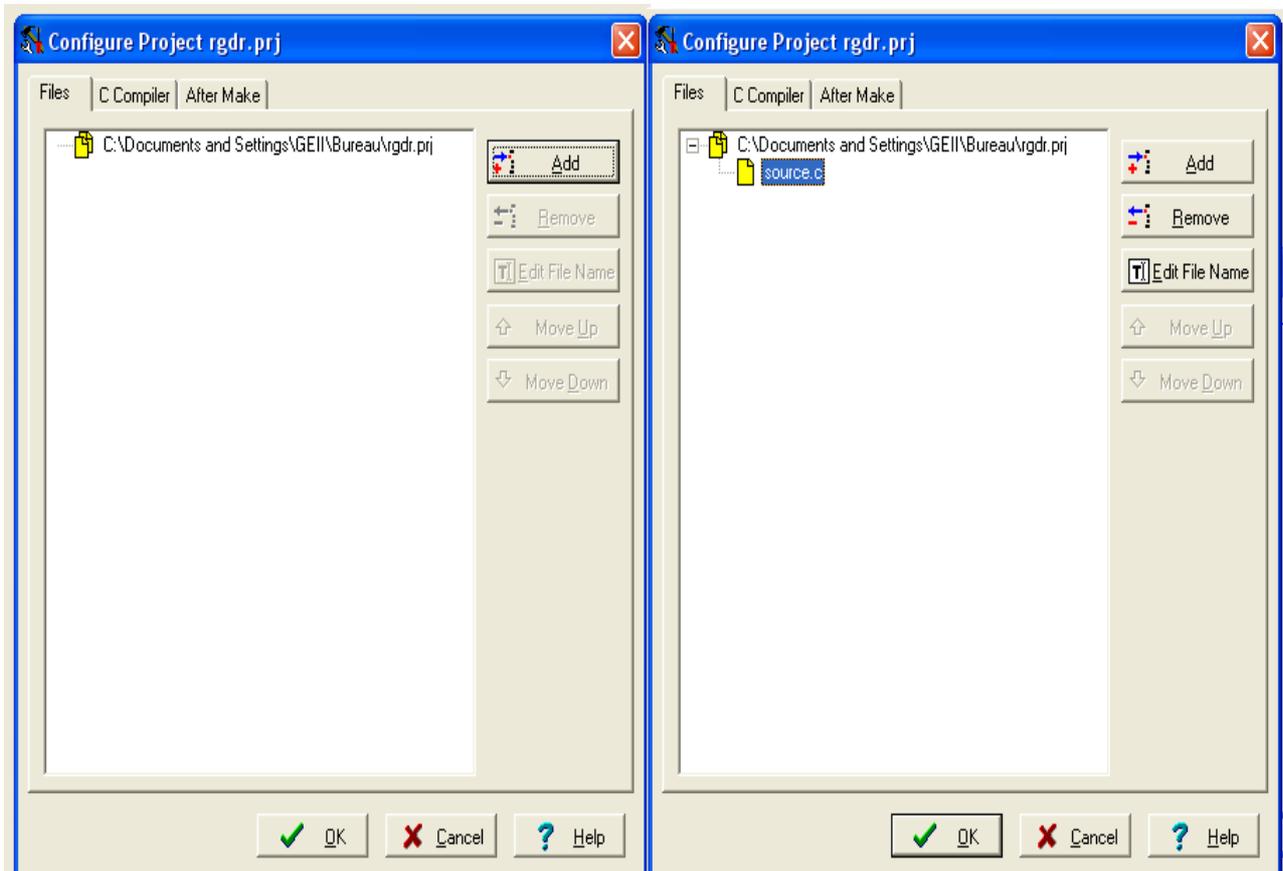


Illustration 13: Configuration de CodeVisionAVR 2[5]

## 4.2. Programme de test

Nous avons à disposition un module de programmation que nous relierons du PC à la carte grâce au connecteur de programmation .

Afin de tester le fonctionnement de la carte nous avons créé un programme simple.

```
while(1)
{
    PORTA=0b00111000;    //On met les étages (Anode des LEDs) à 5V
    PORTC=0b00000000;    //On met toutes les cathodes des LEDs à 0V
}
```

Ce programme permet d'allumer toutes les LEDs quelque soit l'état des entrées. Grâce à ce programme, nous allons voir si toute les sorties sont correctement connectées ainsi que le bon fonctionnement de toutes les LEDs.

Nous avons vu qu'une des LEDs ne fonctionnait pas. Nous l'avons donc remplacée.

## 4.3. Programmation

### 4.3.1. Ordinogramme

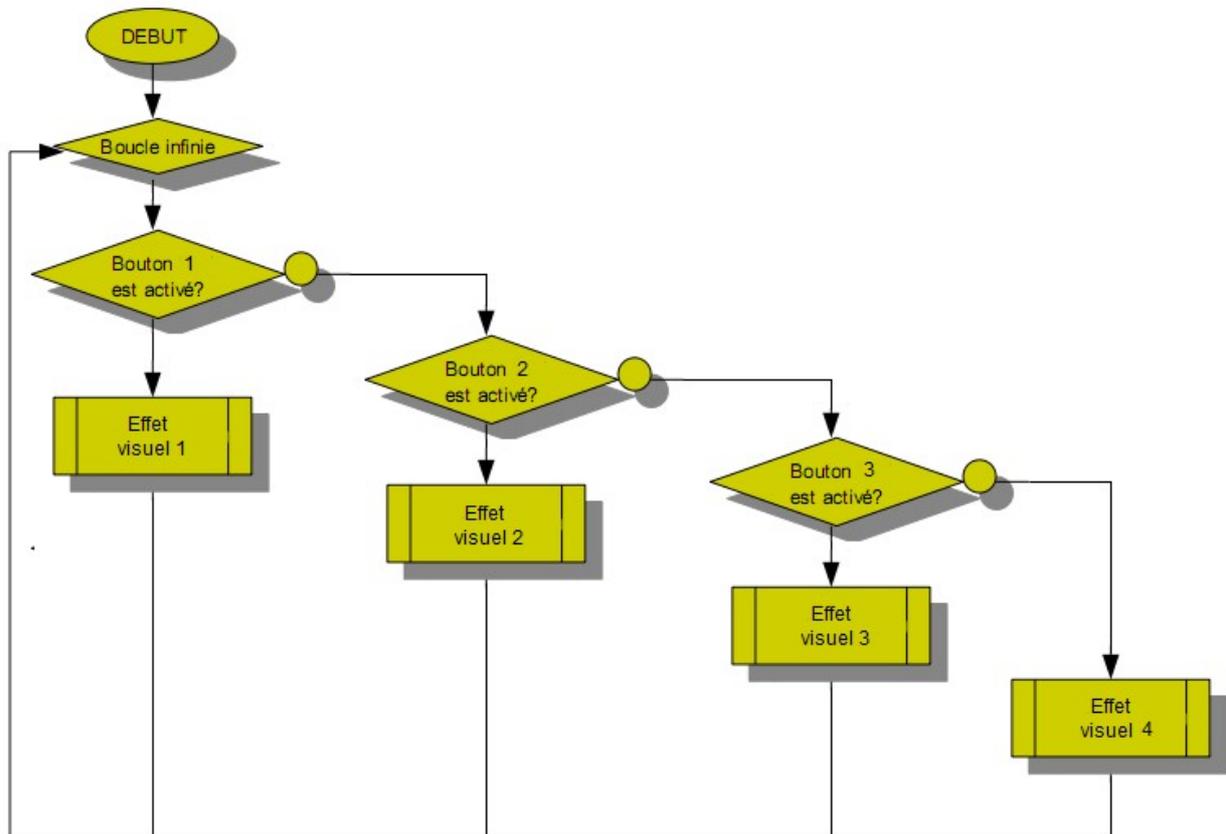


Illustration 14: Ordinogramme [5]

On pourra choisir trois effets visuels différents commandés par les boutons poussoirs ainsi qu'un effet visuel dit « d'attente » lorsqu'aucun des boutons poussoirs n'est activé. On pourra par la suite augmenter le nombre d'effets visuels en combinant plusieurs boutons poussoirs.

### 4.3.2. Programme1

Dans un premier temps on déclare les différentes variables (entrées sorties).

```
//Déclaration des Sorties  
#include<delay.h>  
#define LED1 PORTA.2  
#define LED2 PORTA.1  
#define LED3 PORTA.0  
#define LED4 PORTC.5  
#define LED5 PORTC.4
```

```

#define LED6 PORTC.3
#define LED7 PORTC.2
#define LED8 PORTC.1
#define LED9 PORTC.0
#define Etage1 PORTA.3
#define Etage2 PORTA.4
#define Etage3 PORTA.5

//Déclaration des Entrées
#define BP1 PIND.0
#define BP2 PIND.1
#define BP3 PIND.2

```

Il faut ensuite configurer les quatre ports de l'Atmega 8535 afin de définir les entrées et les sorties.

Chaque port est configuré sur 8 Bits. En mettant un bits à 1 on le configure en sortie et lorsque le bits est à 0 il est configuré en entrée (Sur le registre DDRX).

```

//Initialisation du port A → Port A en sortie
PORTA=0x00;
DDRA=0x00;

//Initialisation du port B → Port B non utilisé
PORTB=0x00;
DDRB=0x00;

//Initialisation du port C → Port C en sortie
PORTC=0x00;
DDRC=0x00;

//Initialisation du port D → Port D en entré
PORTD=0x07; //On initialise les Bps à 1 afin que lorsqu'un Bp ne soit pas activé
DDRD=0xFF; //on ait le PORTD de ce Bp à 1.

MCUCR=0x00; //Interruption externe désactivé
MCUCSR=0x00;

ACSR=0x80;
SFIOR=0x00;

```

Puis on rentre le programme voulu.

Exemple:

```

if(BP1==1)
{
    PORTA=0b00111000 //On alimente les 3 étages en +5v
    PORTC=0b00000000 //On met toutes les cathodes des LEDs à 0
}

```

}}  
Nous avons créé deux fonctions permettant de gérer l'allumage des étages ainsi que l'allumage des LEDs.

```
void Etage(int E1, int E2, int E3)
{
    if(E1==1)
        Etage1=1;    //Si l'on désire activer l'étage 1 on va mettre la sortie PORTA.3 à 1
    if(E2==1)
        Etage2=1;
    if(E3==1)
        Etage3=1;
    if(E1==0)
        Etage1=0;
    if(E2==0)
        Etage2=0;
    if(E3==0)
        Etage3=0;
}
```

Afin d'allumer une LED il faut que son étage soit alimenté en 5V et que sa colonne soit mis à 0V. Si on désire désactiver une LED, il suffit de mettre sa colonne à 5V pour que quelque soit l'état de l'étage la LEDs ne sera pas allumé.

```
void Leds(int D1, int D2, int D3, int D4, int D5, int D6, int D7, int D8, int D9)
{
    if(D1==1)
        LED1=0;    //Si l'on désire activer la LED 1 on va mettre la sortie PORTA.2 à 0
    if(D1==0)
        LED1=1;    //Si l'on désire désactiver la LED 1 on va mettre la sortie PORTA.2 à 1
    if(D2==1)
        LED2=0;
    if(D2==0)
        LED2=1;
```

```

.....
if(D9==1)
    LED9=0;
if(D9==0)
    LED9=1;
}

```

Il nous suffit par la suite d'envoyer à ses fonctions quels étages nous voulons programmer et quelles LEDs de ces étages nous voulons allumer.

```

if(BP1==1)
{
    Etage(1,0,0);      //Ici on désire allumer l'étage 1
    Leds(0,0,0,      // Et la LED du milieu
        0,1,0,
        0,0,0);
    delay_ms(200);
}

```

Pour allumer différentes LEDs sur plusieurs étages on doit utiliser le multiplexage. Cela consiste à allumer une seule LED à la fois, très rapidement. Le traitement se faisant très rapidement, l'œil voit les différentes LEDs allumées en même temps. Pour cela nous utilisons une boucle « while » qui nous permettra également de choisir le temps d'affichage de la séquence.

Exemple

```

while(i<No) //Le nombre que l'on choisi définira le temps (No correspond au temps
            // que l'on veut multiplier par (N x n) en ms ).
{          //n étant le nombre de delay_ms dans la boucle while .
    Etage(1,0,0);      //Ici on désire allumer la LED en bas à gauche de l'étage 1.
    Leds(0,0,0,
        0,0,0,
        1,0,0);
    delay_ms(N);      //N est le temps en ms.
    Etage(0,1,0);     //On allume la LED du milieu de l'étage 2.
    Leds(0,0,0,

```

```

        0,1,0,
        0,0,0);
    delay_ms(N);
    i++;
}
i=0;           //On réinitialise i pour les prochaines séquences.

```

Dans cet exemple la LED en bas à gauche de l'étage 1 (du bas) et la LED du milieu de l'étage 2 (du milieu) s'allumeront alternativement très rapidement afin de les voir allumées en même temps.

### 4.3.3. Programme2

Nous avons décidé de créer un programme exécutable permettant de choisir les LEDs que nous désirons allumer et le temps entre chaque séquence, de visualiser la séquence et d'écrire le programme. Pour cela le programme2 va demander la saisie d'un nom de fichier puis va créer un fichier « .c » et écrire le programme1 dans ce fichier. Par la suite il suffira de lancer ce fichier grâce à CodeVisionAVR et l'envoyer directement dans le microcontrôleur. Pour ce faire, nous utilisons la fonction `fopen(Nom,"a");` où `Nom` est le nom que l'on veut donner au fichier ainsi que sa destination et "a" signifie que l'on se trouve en mode 'Ajouter'. C'est à dire qu'à chaque appel à la fonction d'écriture '`fprintf`' la partie à écrire sera écrite à la suite de ce qui est déjà écrit dans le fichier. Pour utiliser la fonction « `fprintf(fichier,"Texte");` » il faudra indiquer le nom du fichier dans lequel on désire écrire et le texte à écrire.

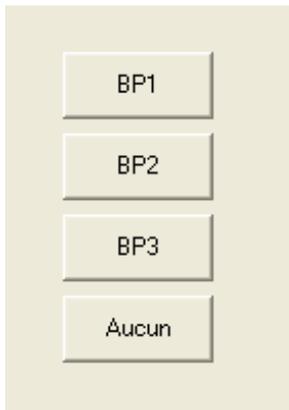
Déroulement du programme2

Illustration 15: Programme2 Déroulement 1 [5]

Ici on choisit le nom du fichier dans lequel va être écrit le programme1

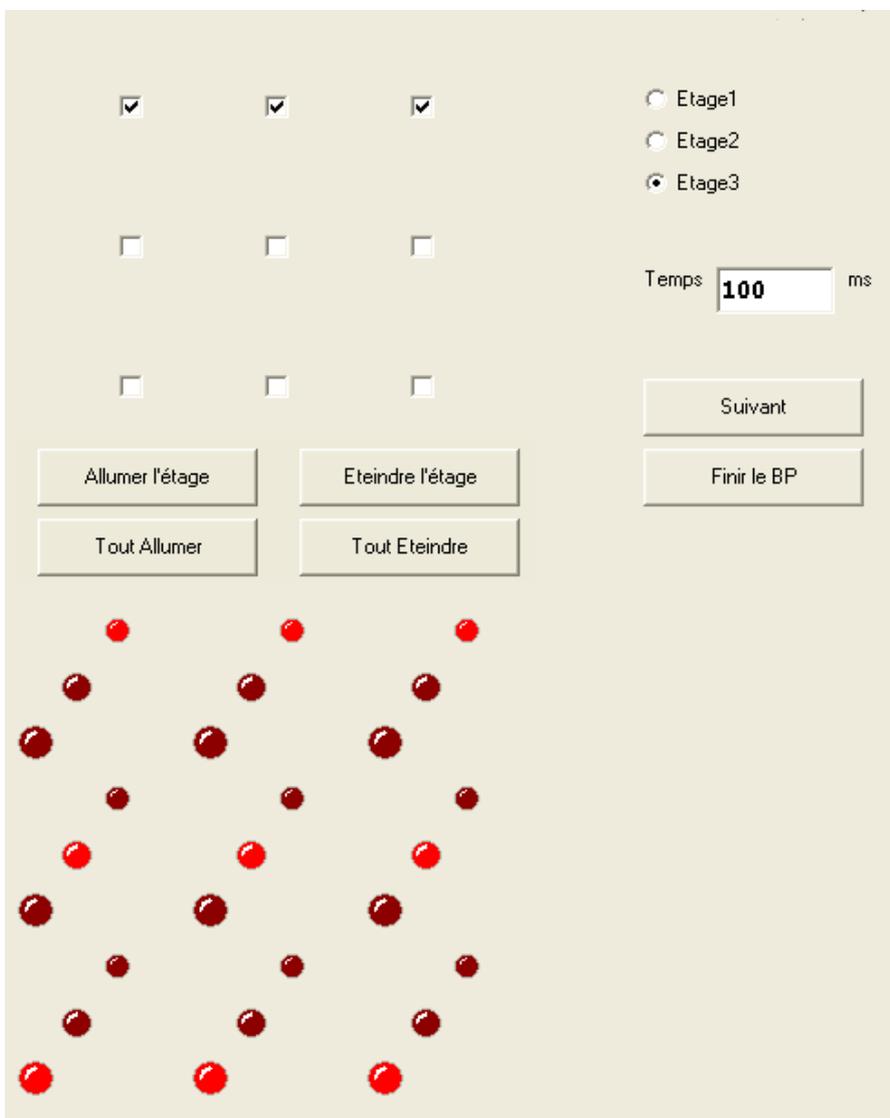
Lors de l'appui sur « commencer », le programme2 va créer le fichier et écrire dedans toutes l'initialisation du programme1,

c'est à dire la configuration des ports de l'Atmega, la déclaration des entrées sorties et les 2 fonctions "Etage" et "Leds".



Ici on va pouvoir choisir sur quel bouton poussoir on va programmer notre séquence. L'appui sur un des bouton va écrire "if(BPx==1) {" avec x la valeur du bouton poussoir, et va afficher la suite.

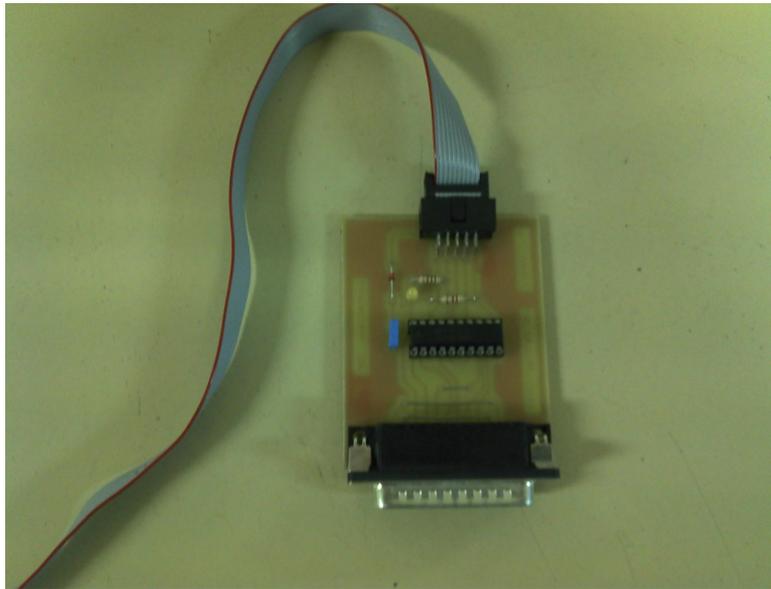
*Illustration 16:  
Programme2  
Déroulement 2 [5]*



*Illustration 17: Programme2 Déroulement 3 [5]*

Ici on va pouvoir sélectionner les différentes LEDs des différents étages que l'on désire allumer, le temps entre deux séquences et on pourra visualiser ce que cela représente sur le cube. Le bouton « Finir le BP » va nous permettre de finir la programmation du bouton poussoir choisi et d'en choisir un autre ou de finir complètement le programme2. Ensuite il suffira de choisir, dans CodeVisionAVR, "configure" dans l'onglet "Project" puis "Add" et sélectionner le fichier que l'on vient de créer.

Afin de pouvoir programmer l'Atmega nous utiliserons un connecteur de programmation mis à notre disposition.



*Illustration 18: Connecteur de programmation[5]*

#### **4.4. Améliorations**

Afin d'améliorer le projet, quelques modifications sont à apporter. Tout d'abord il est toujours possible de rapprocher les composants entre eux afin de réduire la taille de la carte au minimum. Il faut revoir le système de straps<sup>5</sup> ( connexion via un fils non intégré à la carte) afin soit de les éliminer, soit d'optimiser leurs insertions. Il est possible d'ajouter des LEDs afin d'avoir un cube plus important ainsi que des effets visuels plus optimisés ou de choisir des LEDs RGB (Red Green Blue) qui permettront de choisir la couleur des LEDs que l'on allume. Enfin, il est possible de

---

<sup>5</sup> Straps: Connexion de deux pastilles via un fil.

créer un boîtier permettant d'y insérer la carte et par conséquent d'éviter toute détérioration de celle-ci lors de l'utilisation ainsi qu'un boîtier en verre afin de protéger le cube qui est très fragile.

## 5. Liste des composants

NOM	Référence	Empreinte	Valeur	Quantité	Prix Unitaire	Prix Total
Résistances	R1, R2, R3, R4, R5, R6, R7, R8, R9	RC04L	0,870 K $\Omega$	9	0,05 €	0,30 €
Condensateurs	C1, C2	CK 06	22 pF	2	0,51 €	1,02 €
	C3	RADIAL 06	10 uF	1	0,45 €	0,45 €
	C4	CK 06	100 nF	1	0,51 €	0,51 €
	C5	RADIAL 08	100 uF	1	0,45 €	0,45 €
	C6	RADIAL O6L	470 uF	1	0,45 €	0,45 €
LEDs	D1	LED3	3mm	1	0,09 €	0,09 €
	D2-D28	LED05	5mm	27		
Diodes	D12	D041	1N5819	1	2,00 €	2,00 €
Inductances	L1	10 uH	RADIAL06 L	1	0,73 €	0,73 €
Quartz	Q9	HC18UV	16 MHz	1	0,39 €	0,39 €
Micro-Contrôleur	U1	40DIP600L	Atmega 8535	1	5,93 €	5,93 €
Régulateur	U2	08DIP300L	LM2574N	1	1,80 €	1,80 €
Interrupteurs	SW1, SW2, SW3, SW4	SY241		4		
Borniers	JP1	CON ISP		1	2,40 €	2,40 €
	JP2	WEID2		1	2,30 €	2,30 €
<b>TOTAL</b>				<b>54</b>		<b>18,37 €</b>

## 6. Planning final

N° Semaine	5	6	7	8	9	10	11	12	13	14
Choix du sujet	Prévisionnel		Vacance	Vacance						
Cahier des charges	Prévisionnel		Vacance	Vacance						
Recherche de solutions		Prévisionnel	Vacance	Vacance	Prévisionnel					
Realisation du typon		Prévisionnel	Vacance	Vacance	Prévisionnel					
Programmation			Vacance	Vacance	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel		
Test			Vacance	Vacance		Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel	
Prototype			Vacance	Vacance	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel		
Redaction synthese	Prévisionnel	Prévisionnel	Vacance	Vacance	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel	Prévisionnel	
Remise du dossier			Vacance	Vacance					Prévisionnel	
Oral			Vacance	Vacance						Prévisionnel

	Prévisionnel		Vacance		Réel
--	--------------	--	---------	--	------

*Illustration 19: Planning final*

## Conclusion

La réalisation de notre carte qui a été basée principalement sur le microcontrôleur Atmega 8535 nous a permis d'étudier trois éléments essentiels: La partie alimentation à découpage que nous avons appris dans le cours MC-ET2, la partie programmation en cours d'informatique et la partie composants électroniques dans le cours EN1.

Nous remercions sincèrement notre professeur M<sup>r</sup> Thierry LEQUEU, qui nous a aidé pendant la réalisation de notre carte électronique.

A ce jour, nous avons eu le temps de réaliser le test de la programmation ainsi que plusieurs programmes permettant différents effets visuels. La carte ainsi que le cube final correspond à nos attentes. Nous avons aussi réaliser un programme exécutable nous permettant d'écrire le programme en sélectionnant les différents effets choisies.

Pour conclure, nous pouvons dire que ce projet nous a permis de mettre en œuvre nos compétences vues tout au long de l'IUT. Ainsi que de travailler en équipe tout en respectant un cahier des charges et des contraintes dues à la durée. Tout cela nous a permis de mieux nous préparer à notre stage de fin d'année ainsi qu'à la suite de nos études.

## Résumé

Nous avons choisi comme sujet la fabrication d'un cube de LEDs 3x3x3 ainsi que d'une carte électronique permettant sa programmation. Nous avons créé la carte électronique comportant un microcontrôleur programmable, comme composant principal de notre projet, qui sert à commander les sorties en fonction des différentes entrées. Les entrées sont composées de plusieurs boutons poussoirs à deux positions qui sont placés sur la carte afin de commander les différents effets visuels. Nos sorties sont reliées aux anodes et cathodes des différentes LEDs afin d'allumer les LEDs voulues. Nous avons utilisé le système de multiplexage dans notre programmation afin de contrôler les différentes LEDs et réduire le nombre de sortie utilisées par le microcontrôleur. Nous avons utilisé une alimentation à découpage de type BUCK afin d'obtenir une tension continue de 5 Volts à partir d'une tension continue de 12 Volts pour alimenter notre microcontrôleur. Grâce au logiciel « Orcad Capture et Layout », nous avons pu réaliser la carte électronique en créant le schéma électrique sur « Orcad Capture » et nous avons terminé le routage des liaisons par « Orcad Layout ». Pour implanter le programme dans le microcontrôleur Atmega 8535, nous avons dû utiliser le logiciel « Code Vision AVR » qui est basé principalement sur le langage C. Nous avons par la suite, décidés d'utiliser « C++ Builder » afin de créer un exécutable permettant d'écrire le programme automatiquement en choisissant les LEDs que l'on souhaite allumer ou éteindre ainsi que le temps entre chaque séquences.

239 mots.

## Index des illustrations

Illustration 1: Schéma fonctionnel [5].....	7
Illustration 2: Planning prévisionnel [5].....	8
Illustration 3: Schéma électrique du cube [2][5].....	9
Illustration 4: Schéma du régulateur LM2574 [3].....	10
Illustration 5: Schématique de l'alimentation à découpage[3][5].....	11
Illustration 6: Photo de l'Atmega[1].....	12
Illustration 7: Assignation des broches de l'Atmega[1].....	12
Illustration 8: Schématique des sorties[5].....	13
Illustration 9: Schématique d'un bouton poussoir [5].....	14
Illustration 10: Typon final .....	15
Illustration 11: Photo de la carte [5].....	16
Illustration 12: Configuration de CodeVisionAVR[5].....	17
Illustration 13: Configuration de CodeVisionAVR 2[5].....	17
Illustration 14: Ordinogramme [5].....	19
Illustration 15: Programme2 Déroulement 1 [5].....	23
Illustration 16: Programme2 Déroulement 2 [5].....	24
Illustration 17: Programme2 Déroulement 3 [5].....	24
Illustration 18: Connecteur de programmation[5].....	25
Illustration 19: Planning final.....	27

## Bibliographie

[1] LEQUEU Thierry, <http://www.thierry-lequeu.fr>, (page consultée le 07/04/2010).

[2] <http://www.instructables.com/id/LED-Cube-3x3x3-with-ATMEGA8>, (page consultée le 07/04/2010).

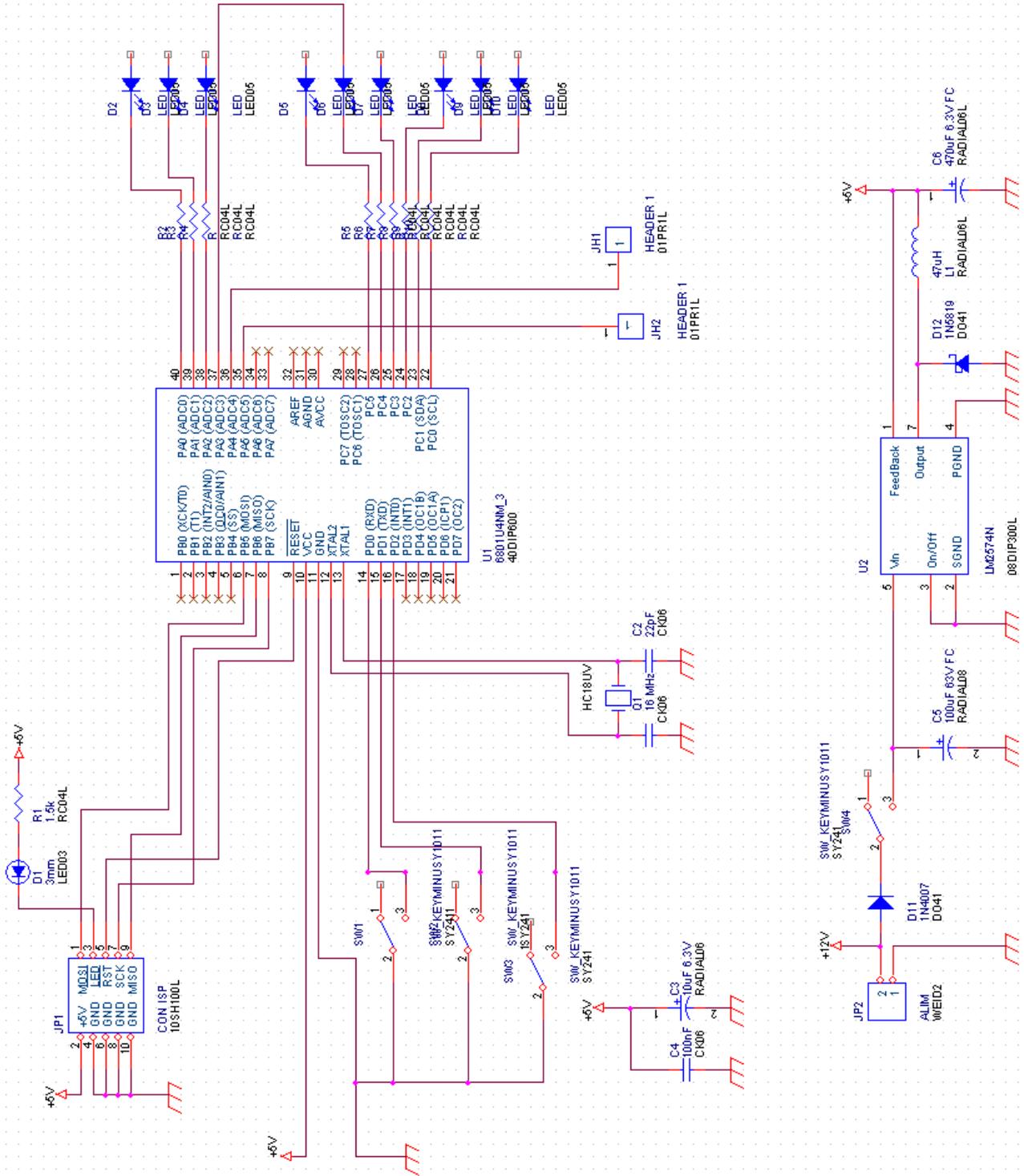
[3] LEQUEU Thierry, Cours de MC-ET2, Chapitre sur le BUCK.

[4] BESSE Dominique, Module complémentaire microprocesseur.

[5] Éléments personnels.

# Annexes

## Annexe 1: Schéma capture de la carte centrale du système



## Annexe 2: Programmation

```
#include <delay.h>
#include <mega8535.h>
#define LED1 PORTA.2
#define LED2 PORTA.1
#define LED3 PORTA.0
#define LED4 PORTC.5
#define LED5 PORTC.4
#define LED6 PORTC.3
#define LED7 PORTC.2
#define LED8 PORTC.1
#define LED9 PORTC.0
#define Etag1 PORTA.3
#define Etag2 PORTA.4
#define Etag3 PORTA.5
#define BP1 PIND.0
#define BP2 PIND.1
#define BP3 PIND.2
void Etag(int E1, int E2, int E3)
{
if(E1==1)
    Etag1=1;
if(E2==1)
    Etag2=1;
if(E3==1)
    Etag3=1;
if(E1==0)
    Etag1=0;
if(E2==0)
    Etag2=0;
if(E3==0)
    Etag3=0;
}
```

```

void Leds(int D1, int D2, int D3, int D4, int D5, int D6, int D7, int D8, int D9)
{
if(D1==1)
    LED1=0;
if(D1==0)
    LED1=1;
if(D2==1)
    LED2=0;
if(D2==0)
    LED2=1;
...
    LED8=1;
if(D9==1)
    LED9=0;
if(D9==0)
    LED9=1;
}
void main(void)
{
int i=0;
PORTA=0x00;
DDRA=0xFF;
PORTB=0x00;
DDRB=0x00;
PORTC=0x00;
DDRC=0xFF;
PORTD=0x07;
DDRD=0x00;
MCUCR=0x00;
MCUCSR=0x00;
PORTA=0b00000111;
PORTC=00111111;
while(1)
{

```

```

if((BP1==1)&&(BP2==0)&&(BP3==0))
{
    while(i<13)
    {
        Etage(1,0,0);
        Leds(1,1,1,1,0,1,1,1,1);
        delay_ms(5);
        Etage(0,1,0);
        Leds(1,1,1,1,0,1,1,1,1);
        delay_ms(5);
        Etage(0,0,1);
        Leds(1,1,1,1,0,1,1,1,1);
        delay_ms(5);
        i++;
    }
i=0;
    while(i<13)
    {
        Etage(1,0,0);
        Leds(0,0,0,0,1,0,0,0,0);
        delay_ms(5);
        Etage(0,1,0);
        Leds(0,0,0,0,1,0,0,0,0);
        delay_ms(5);
        Etage(0,0,1);
        Leds(0,0,0,0,1,0,0,0,0);
        delay_ms(5);
        i++;
    }
i=0;
}
}
}

```