Lampadaire à LED RGB

Tanguy PEPIN
Groupe K4
Année 2014/2015

M. Thierry LEQUEU
M. Philippe AUGER

Table des matières

Introduction

Dans le cadre de cadre du cours d'étude et réalisation du semestre 4, j'ai décidé de réaliser un lampadaire à LED RGB. Ce projet est un vrai produit qui manque dans le commerce, car aujourd'hui il n'en existe que très peu de modèles et souvent extrêmement chers, de 600 à 1600€ en général. Ce prix excessif est principalement dû au design du produit qui souvent est très poussé.

Le projet est donc principalement de réaliser la partie électronique du lampadaire avec un coût de revient assez bas placé ici à 200€ soit 3 fois moins que le produit sur le marché.

Ce rapport a donc pour but de vous expliquer le fonctionnement du lampadaire, ainsi que chacune des étapes permettant la réalisation de ce lampadaire.

Suite à notre cahier des charges, nous allons vous expliquer le principe physique que nous allons utiliser pour nos LED qui sont la synthèse additive. Dans une seconde partie, nous allons vous montrer l'étude de nos schémas, pour ensuite passer à la réalisation de carte. Puis pour finir, nous vous présenterons l'avenir de ce projet.

1. Cahier des charges

Le projet réside dans la réalisation d'un lampadaire à LED de puissances via des LED de puissances de couleur rouge, verte et bleu afin de réaliser le principe de synthèse additive. Le lampadaire devra répondre aux critères suivants :

- Utilisation de LED de puissance de couleur rouge, verte et bleu
- Avoir une haute intensité lumineuse
- Consommation électrique la plus faible possible
- Coût de revient inférieur à 200€
- Utilisation d'un gradateur à microcontrôleur
- Optionnel : Commande sans fil du lampadaire

2. Principe de la synthèse additive

Je vais donc commencer par vous expliquer la synthèse additive, car c'est ce principe que j'utilise avec les LED de couleur rouge, verte et bleu pour le lampadaire. Ce procédé consiste à combiner les lumières de plusieurs sources colorées, et cela a pour but d'obtenir au final une lumière colorée en l'addition de ces sources lumineuses.

Ce principe est aujourd'hui implanté dans des produits que nous utilisons tous les jours que sont les écrans de télévision, ordinateurs ou de nos portables.

On représente souvent la synthèse additive par ce schéma ou on peut voir la superposition des couleurs rouge, verte et bleu pour obtenir les couleurs jaunes, cyan ainsi que bleu. Au final la superposition des trois couleurs de base, permettent d'avoir la couleur blanche.

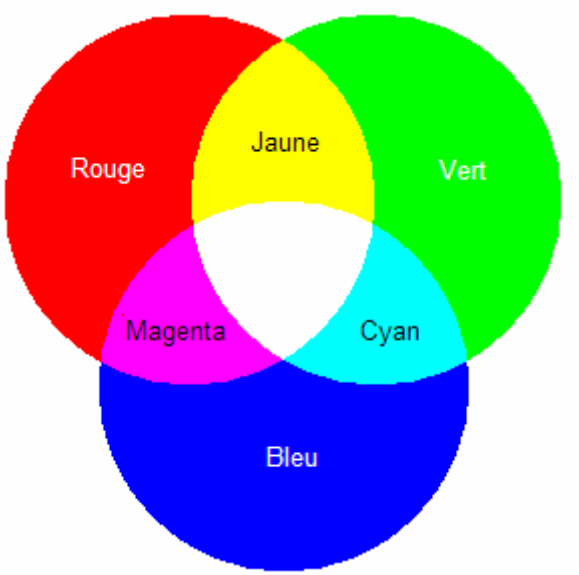


Figure 1 Principe de la synthèse additive

Avec ce principe de la synthèse additive, on peut donc avoir toutes les nuances de couleur en n'ayant qu'à la base les couleurs rouge, vertes et bleu.

Suite à cette explication sur la synthèse additive je vais vous montrer l'étude du schéma permettant le fonctionnement du lampadaire.

3. Étude du schéma

Le fonctionnement du lampadaire réside en deux circuits, celui de commande effectué avec un microcontrôleur ATMEGA8535 et le second circuit celui de puissance effectué via des transistors de type MOSFET.

1. Circuit de commande

Le circuit de commande tourne autour d'un microcontrôleur ATMEGA8535. Le choix de microcontrôleur s'est effectué par sa disponibilité dans le magasin de l'IUT, mais aussi, car il répondait parfaitement à mes attentes qui étaient 3 sorties PWM, 4 entrées analogiques ainsi qu'un port pour ajouter un écran LCD.

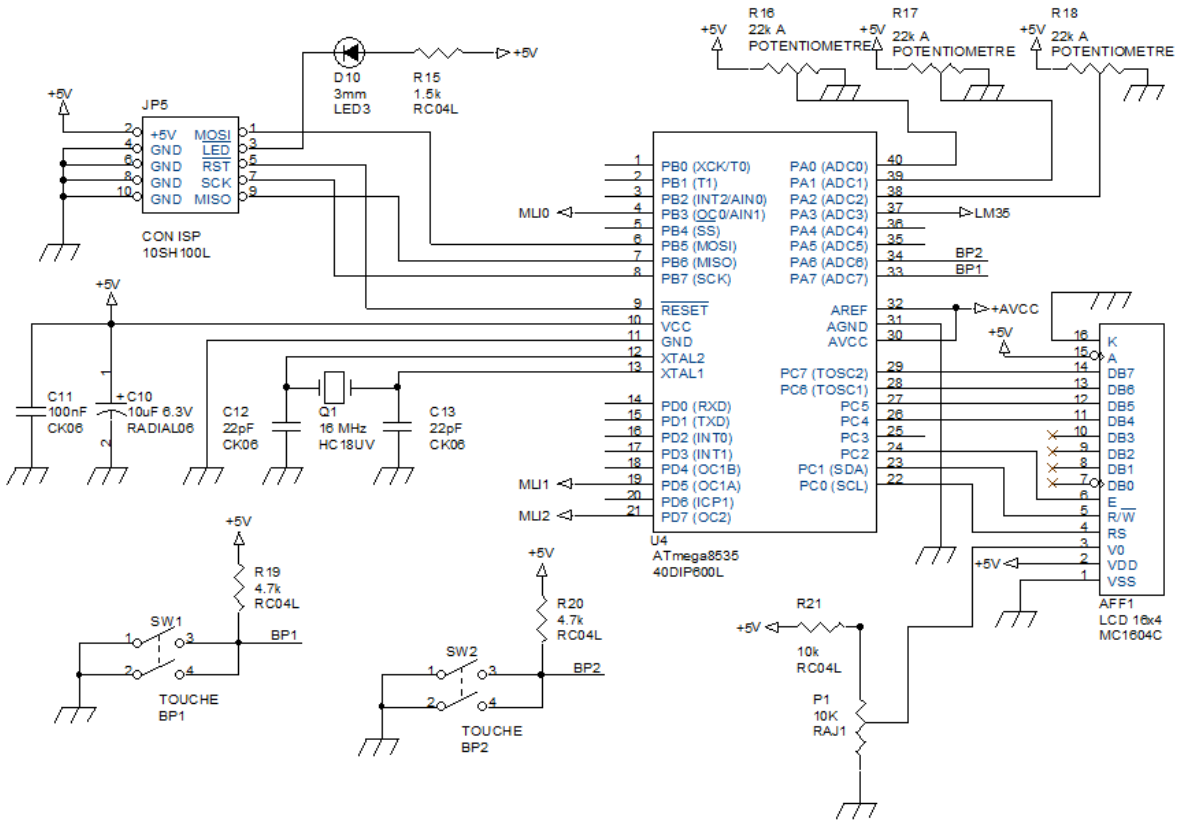


Figure 2 Schéma du circuit de commande

Comme on peut voir sur ce schéma, nous avons l'ATMEGA8535 (U4) avec tous les circuits annexes pour son bon fonctionnement. Le quartz (Q1) permet de définir la fréquence de fonctionnement du microcontrôleur qui ici est de 16,000MHz. Ensuite on peut voir le connecteur ISP (In-System Programming) qui permet de programmer directement sur la carte l'ATMEGA, ceci est très utile pour le développement ou la maintenance du programme. Enfin on peut voir que l'écran LCD (AFF1) est lui relié au port C de l'ATMEGA.

Pour pouvoir contrôler l'intensité lumineuse de chaque couleur, nous allons modifier le rapport cyclique d'un signal de PWM, en effet le rapport cyclique étant proportionnel à la moyenne d'un signal. Nous allons donc faire varier la tension aux bornes des LED et ainsi réduire ou augmenter leur intensité lumineuse. On pourra donc contrôler via des différents menus commandés par les boutons SW1 et SW2. Ces boutons ont une résistance de pull-up elles permettent de fixer le niveau à 1 si le bouton n'est pas actif. Cela a pour effet d'éviter l'effet antenne et donc rendre instable cette entrée de l'ATMEGA. Dans un de ces menus, chaque couleur de LED sera gérée via les potentiomètres R15, R17 et R18, ils pourront ne faire ne varier d'aucune luminosité jusqu'à la luminosité maximale.

Pour l'écran LCD nous l'avons relié au port C de l'ATMEGA, car c'était le seul port auquel aucune des sorties ne m'intéressait pour une autre application. Nous avons donc les 4 bits de data (DB7 à DB4), nous avons aussi l'alimentation de l'écran sur VDD et VSS, mais nous avons aussi l'alimentation de la LED de fond qui permet d'éclairer le fond de l'écran. Cette LED de fond est-elle alimentée par son anode (A) et par sa cathode (K). Il y a aussi un potentiomètre (P1) qui permet de régler la luminosité des caractères qui seront affichés sur l'écran.

Pour pouvoir allumer les LED du lampadaire, il faut donc un circuit de puissance permettant comme l'indique son nom d'apporter la puissance électronique nécessaire au bon fonctionnement de celles-ci.

2. Circuit de puissance

Ce circuit de puissance utilise principalement des transistors de type MOSFET. Ces transistors vont se fermer ou non selon la tension qui apparaît sur sa grille.

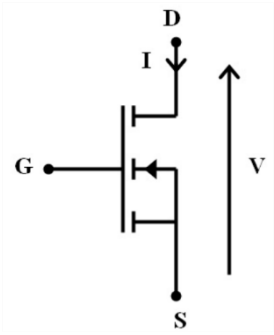


Figure 3 Schéma d'un transistor MOSFET

Ceci va permettre de gérer l'alimentation ou non des LED et donc qu'elle soit allumée ou non. Comme j'ai pu, le dire avant nous aura donc une tension V qui variera selon le rapport cyclique du signal.

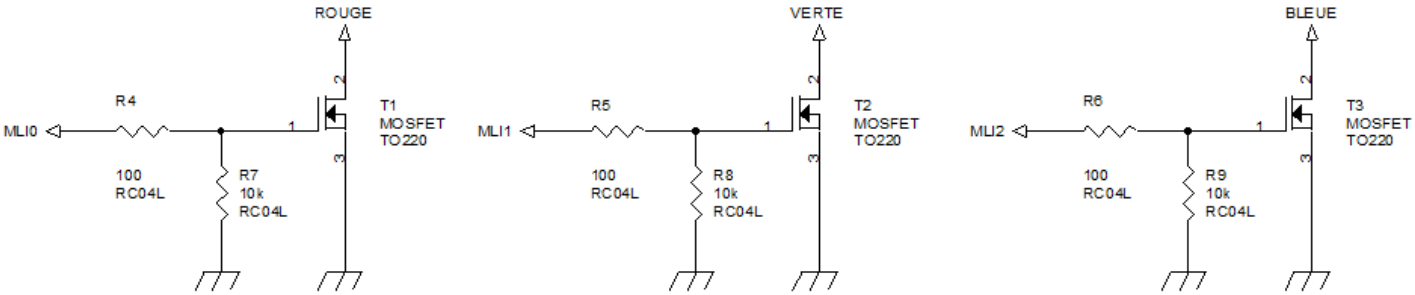


Figure 4 Schéma de puissance

Sur ce schéma on peut voir trois fois le même montage. Ce montage ce compose d'une résistance pour réguler le courant sur la grille du MOSFET, ici R4, R5 et R6. Il y a aussi une résistance de pull-down pour forcer le blocage du transistor pour ne pas alimenter par erreur les LED.

Nous devons maintenant alimenter chaque circuit ainsi que les LED dont voici le fonctionnement du circuit d'alimentation.

3. Circuit d'alimentation

Pour pouvoir alimenter le lampadaire il nous fallait plusieurs tensions qui sont :

- +5V pour l'ATMEGA ainsi que l'écran LCD
- +12V pour le ventilateur de refroidissement des LED
- +24V pour l'alimentation de la LED rouge
- +36v pour l'aliment des LED verte et bleu

Nous avons donc fait le choix d'acheter une alimentation à découpage, car peu chère (20€) et peu encombrante. Elle nous fournit une alimentation de +24V 50W soit un courant de sortie d'un peu plus de 2A, ce qui était fortement nécessaire pour le lampadaire.

Figure 5 Alimentation +24V 50W

Pour la suite nos calculs nous avons pu mesurer la puissance nécessaire pour chaque composant de notre montage :

- ATMEGA : 0.1W
- Écran LCD : 2W
- LED rouge : 10W
- LED bleu et verte : 12.6W
- Ventilateur : 2W

Nous avons donc au total une puissance de 42W que devra fournir l'alimentation.

Comme nous avons pu le voir précédemment nous avons besoin de plusieurs tensions d'alimentation. Partant donc d'une tension +24V il nous a fallu 3 circuits différents pour fournir le +5V, le +12V et aussi le +36V.

Pour l'alimentation +5V et les +12V nous ont utilisé deux montages abaisseurs DC-DC de type Buck ces montages sont assez simple d'utilisation il ne suffit que de quelques composants passifs, pour avoir une tension continue régulée en sortie.

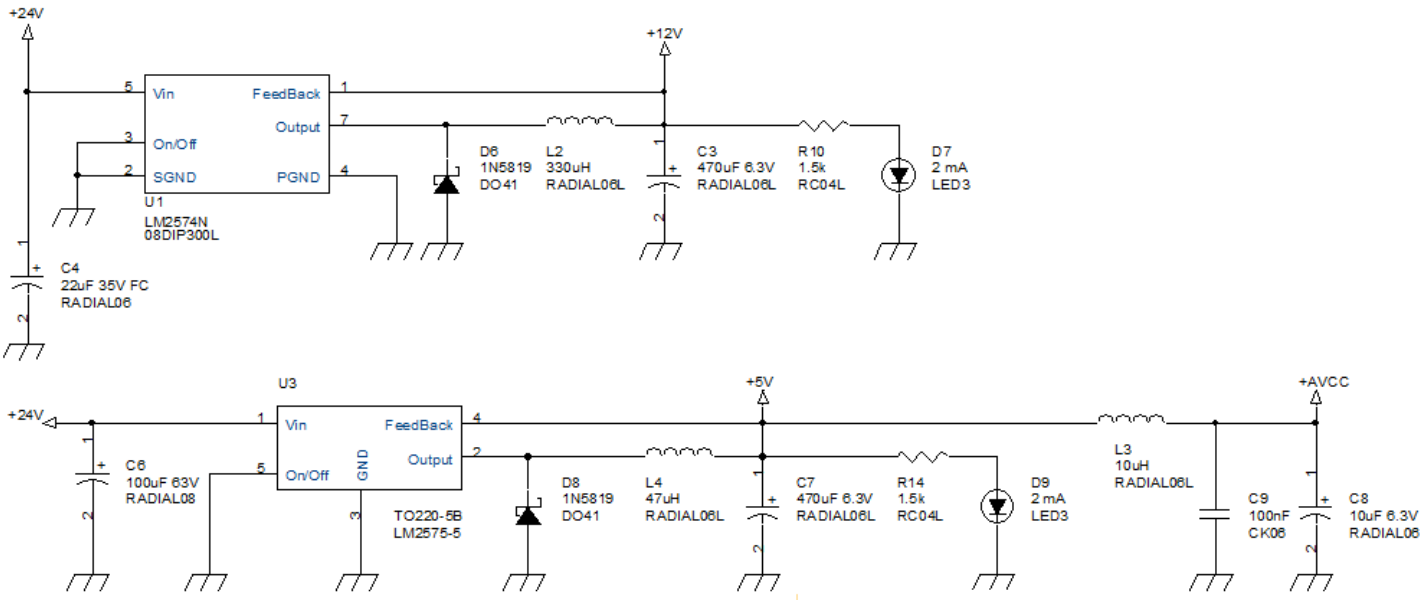


Figure 6 Alimentation +5V et +12V

Pourtant comme vous pouvez voir sur ce schéma que les deux alimentations sont bien différentes, et bien en réalité ils reposent sur la même base, leurs seules différences sont juste pour l'intensité à fournir en sortie. Et oui car pour les +12V il n'y que le ventilateur qui l'utilisera et donc ne consommant que 0.16A nous n'avons pas besoin d'un composant fournissant 1A en sortie. Alors que pour la tension de +5V nous avons besoin déjà de 0,4A pour l'écran LCD et ajoutant tous les circuits alimentés en +5V on s'approche vite des 0.5A. Donc par précaution nous avons préféré choisir une alimentation +5V pouvant débiter 1A. Donc nous avons un LM2575-5V, pour la tension +5V et un LM2475-12V pour la tension +12V.

Maintenant il nous faut une alimentation +36V pour pouvoir alimenter nos LED bleu et verte. Pour cette tension il faut donc augmenter la tension que l'on a en entré nous allons donc utiliser un montage élévateur de type Boost qui va nous fournir une tension de sortie de +36V. Nous allons donc utiliser le composant LM2577-ADJ, ce composant permet d'augmenter la tension qu'il y a en entrer, il a aussi l'avantage de pouvoir choisir sa tension de sorti, les trois dernières lettres de sa référence qui sont ADJ permettent de le définir ainsi. Nous avons donc un composant, dont la tension de sortie qui dépend du rapport entre deux résistances.

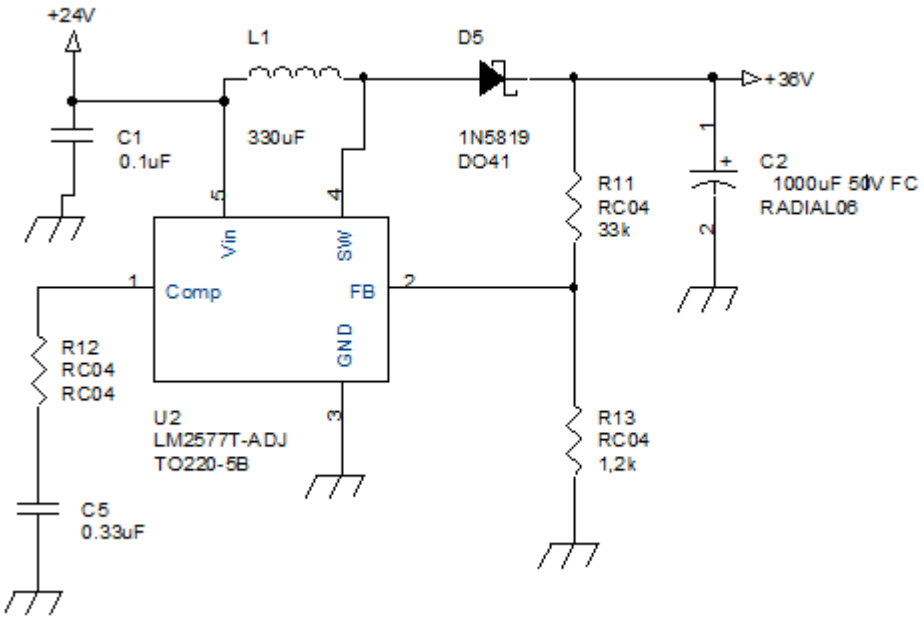


Figure 7 Montage alimentation +36V

Ce rapport est donc défini par les résistances R11 et R13. On peut trouver la formule suivante dans la datasheet du composant :

$$\frac{R11}{R13} = \frac{Vout}{1.23} - 1$$

On finit par trouver R1=33kΩ et R2=1.2kΩ nous avons donc

$$Vout = 1.23\left(1 + \frac{33}{1.2}\right)$$
$$Vout = 1.23 * 28.5$$
$$Vout = 35.055\ V$$

Alors avec ces valeurs de résistance nous avons en sortie une tension de 35V ce qui suffisant pour l'alimentation de nos LED.

4. Réalisation de la carte

4. Réalisation du schéma et du typon de la carte

Pour réaliser le schéma ainsi que le typon, nous avons utilisé le logiciel Orcad en version 9.0. Ce choix c'est effectué avec plusieurs critères qui sont la disponibilité sur chaque ordinateur de l'IUT, mais aussi, et surtout que M. LEQUEU m'a fourni une bibliothèque de composant, ainsi qu'un début de schéma qui m'ont grandement aidé à avancer dans mon projet, et je l'en remercie.

Donc après avoir réalisé le schéma sur Orcad, nous avons dû faire le routage de cette carte.

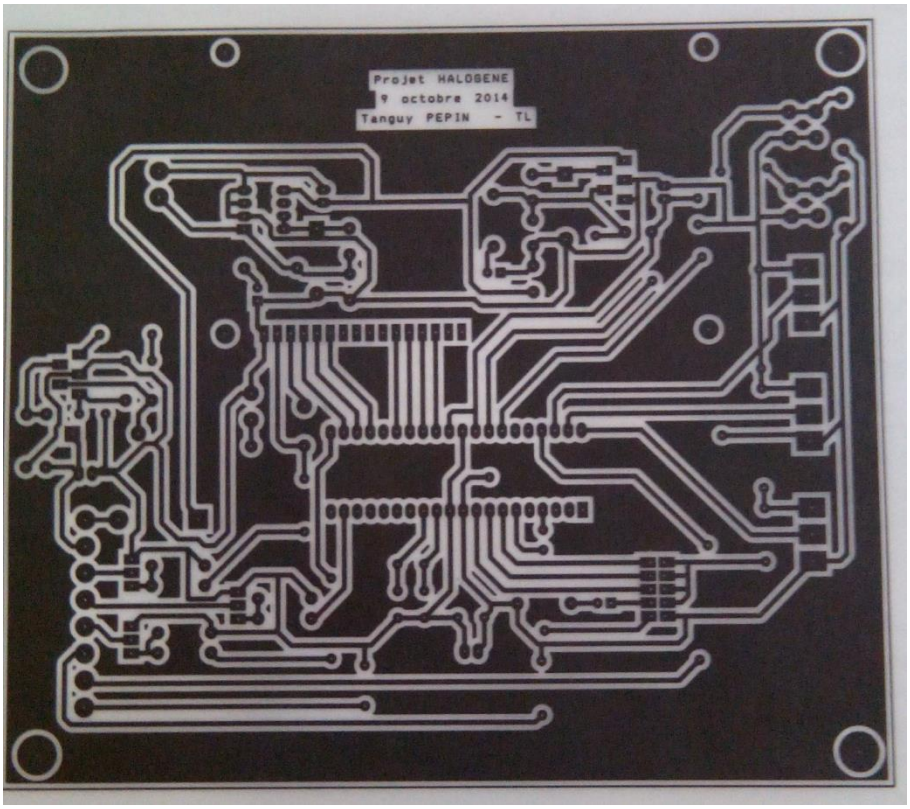


Figure 8 Routage de la carte

Maintenant nous avons fabriqué notre carte qui comprend l'isolation, la gravure et la révélation. Suite il ne suffit plus qu'à la percer et souder chaque composant. Il ne reste plus qu'à implanter un programme dans le microcontrôleur.

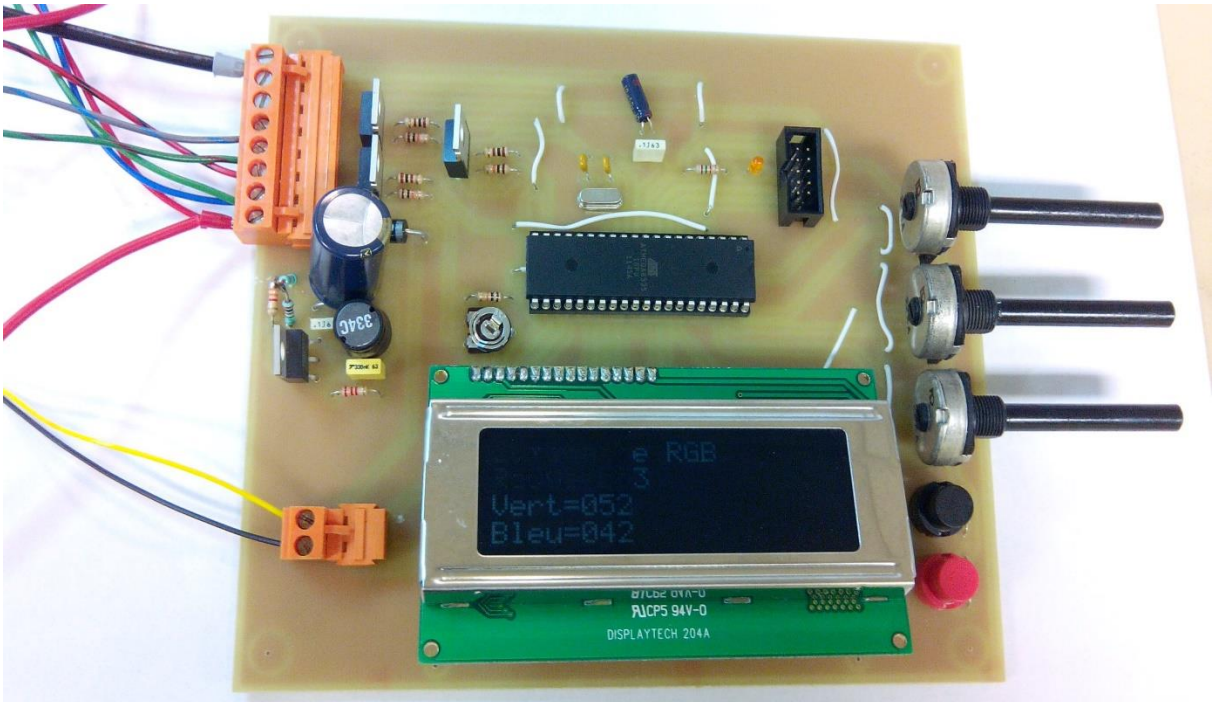


Figure 9 Carte du lampadaire

5. Programmation

La programmation est plutôt simple la partie la plus difficile est la configuration des différentes options auquel nous avons besoin. C'est option sont :

- Utilisation de 3 Timers en PWM
- Lecture des entrées analogique
- Utilisation du PORT C pour l'écran LCD

Utilisant le logiciel CodeVision, toute cette configuration est simplifiée, car nous utilisons l'aide CodeWizard, celle-ci permet de configurer selon des menus chaque option. À la fin de nos choix nous devons justes faire générer le code afin que le logiciel nous donne donc les lignes de code pour pouvoir utiliser chaque option.

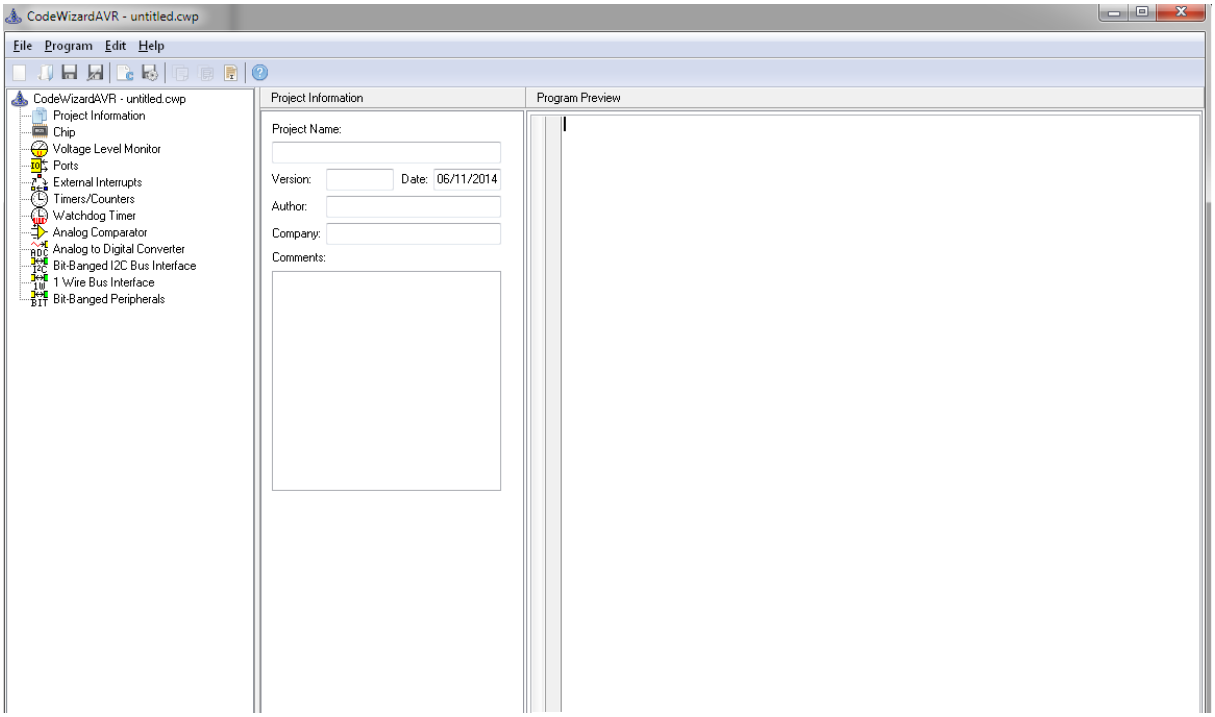


Figure 10 Fenêtre de l'option CodeWizard du logiciel Codevision

Maintenant nous allons vous exposer chaque partie du code servant au bon fonctionnement du lampadaire.

```
#include <mega8535.h>

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

#define ADC_VREF_TYPE 0x20

int pota, potb, potc;
unsigned char tampon[20];
int i;
int BP;
```

On peut voir ici que l'on inclut les librairies de l'ATMEGA, mais aussi pour l'écran LCD, pour avoir des fonctions délai. Nous avons ensuite la déclaration des variables pota, potb et potc elles sont de type entier, ce choix c'est par simplicité, car ils auront la valeur des potentiomètres et comme un entier est codé sur 255 valeurs nous pourrons donc l'utiliser directement pour la PWM de notre signal.

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

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x08;

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

// Port D initialization
// Func7=Out Func6=In Func5=Out Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=0 State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xA0;
```

On peut voir ici la configuration des ports d'entrée ou de sortie de l'ATMEGA.

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Phase correct PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x61;
TCNT0=0x00;
OCR0=0x00;
```

Configuration en PWM phase correct du Timer 0

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x81;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

Configuration en PWM phase correct du Timer 1 avec la sortie sur la pin A du Timer

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

Configuration en PWM phase correct du Timer 2

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

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

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

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
SFIOR&=0xEF;

// LCD module initialization
lcd_init(16);

delay_ms(1000);
```

Désactivation des interruptions externe

Activation des comparateurs analogiques

Activation des entrées analogiques du port D de l'ATMEGA

Initialisation du module pour l'écran LCD

On peut voir que nous avons ajouté un délai de 1 seconde pour qu'il n'y ait pas un apport de puissance trop important d'un coup. Ceci permet donc à l'alimentation de se mettre en régime continu pour ensuite fournir la puissance.

Maintenant voici le programme principal du lampadaire

```
while (1)
      {
      delay_ms(10);


            pota=read_adc(0);
             potb=read_adc(1);
            potc=read_adc(2);

             OCR0=pota;
             OCR1A=potb;
             OCR2=potc;

             lcd_gotoxy(0,0);
             lcd_putsf("Luminaire RGB");
             lcd_gotoxy(0,1);
             sprintf(tampon,"Rouge=%d",pota);
             lcd_puts(tampon);
             lcd_gotoxy(0,2);
             sprintf(tampon,"    Vert=%d",potb);
             lcd_puts(tampon);
            lcd_gotoxy(0,3);
            sprintf(tampon,"    Bleu=%d",potc);
            lcd_puts(tampon);
```

Ce programme tourne avec une boucle infinie while (1). Nous avons un délai de 10 ms pour chaque boucle. Les fonctions read_adc(x) permettent de lire l'entré x du des entrées analogiques du port D. ensuite les variables OCR sont les rapports cycliques de chaque PWM codée de 0 à 255. Nous donnons chaque valeur d'un potentiomètre à un des rapports cycliques d'une PWM. Nous avons enfin l'affichage de la valeur de chaque potentiomètre sur l'écran LCD.

Maintenant que le programme est fini, nous allons passer aux tests et dépannages de la carte.

6. Tests et dépannages

Lors de la mise sous tension, aucune alimentation ne fonctionnait, il a fallu revoir donc chaque alimentation une par une. Pour les alimentations +5V et +36V le problème commun était soudé dans le mauvais sens . Après les avoirs remis dans le bon sens l'alimentation +5V fonctionne très bien en fournissant une tension de +4.95V, mais pour la tension de +36V un autre problème survint sur une erreur de schéma et donc du routage, car il n'y avait pas la tension de +24V en entrée. Un strap à suffit à résoudre ce problème et donc l'alimentation fonctionnait parfaitement fournissant une tension de +35.4V ce qui se rapproche de nos calculs théoriques. Pour fini l'alimentation de 12V le problème était dans le routage d'une piste qui touchait la pin 8 du composant et donc créant une surchauffe du composant entraînant sa destruction.

Ayant toutes les tensions d'alimentation j'ai décidé de brancher l'ATMEGA sur la carte pour vérifier son fonctionnement, mais aussi savoir s'il l'on pouvait bien le programmer sur la carte directement. Tous s'étant bien déroulé je me suis aperçue que l'écran LCD ne s'allumait pas, j'ai pu donc m'apercevoir que la LED du fond de l'écran n'avait pas été alimenté, donc j'ai résolu ce problème avec deux straps pour la masse et le +5V.

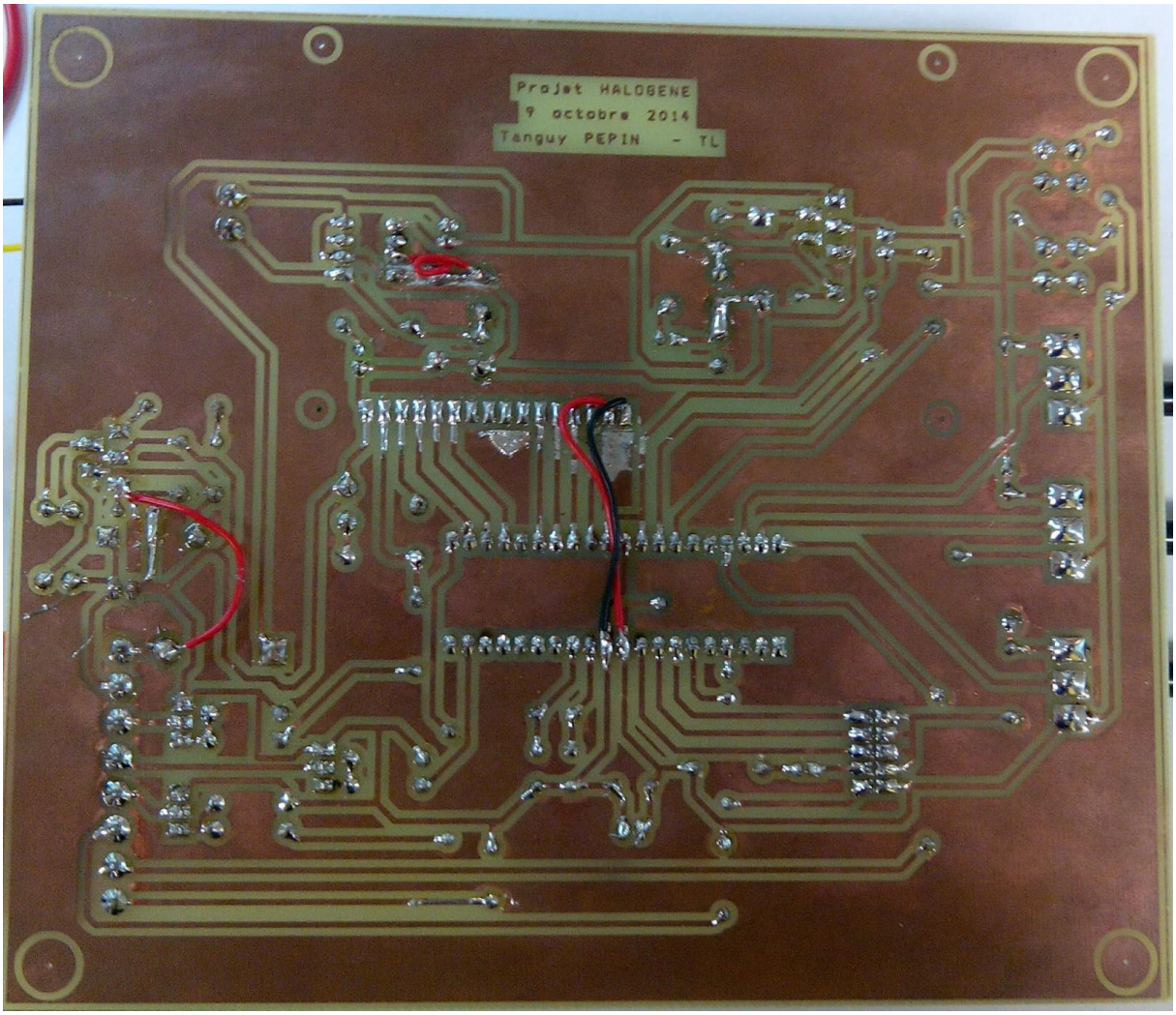


Figure 11 Arrière de la carte après les réparations

Suite au test avec le programme d'implanté dans l'ATMEGA, nous nous sommes aperçu que la consommation était excessive lorsque les trois LED étaient à leur puissance maximale. En effet cette consommation était de 2,3A alors que normalement elle devrait tourner autour des 1,5A au maximum. Nous avons donc décidé d'ajuster la tension des diodes, c'est-à-dire que nous avons alimenté la LED rouge en +22V au lieu de +24V, et pour les LED bleu et vertes nous les alimentons en +30V au lieu de +36V. D'après la datasheet nous nous permettons cet écart, car elle est permise pour pouvoir contenir la puissance consommée.

Pour créer la tension de +30V il nous a fallu ajuster le rapport entre les résistances R11 et R13 comme nous avons pu le voir dans la partie des alimentations. Nous avons au final R11=77kΩ et R13=3,3kΩ.

Maintenant que le montage fonctionnait il nous a fallu passer d'une alimentation de laboratoire à notre alimentation à découpage. Après avoir changé d'alimentation, car la première était défectueuse, nous avons réglé la tension de cette alimentation à +22V nous l'avons donc branché sur notre carte. Donc suite à cela nous avons pu vérifier que tout notre montage, ainsi que le programme fonctionnait parfaitement bien.

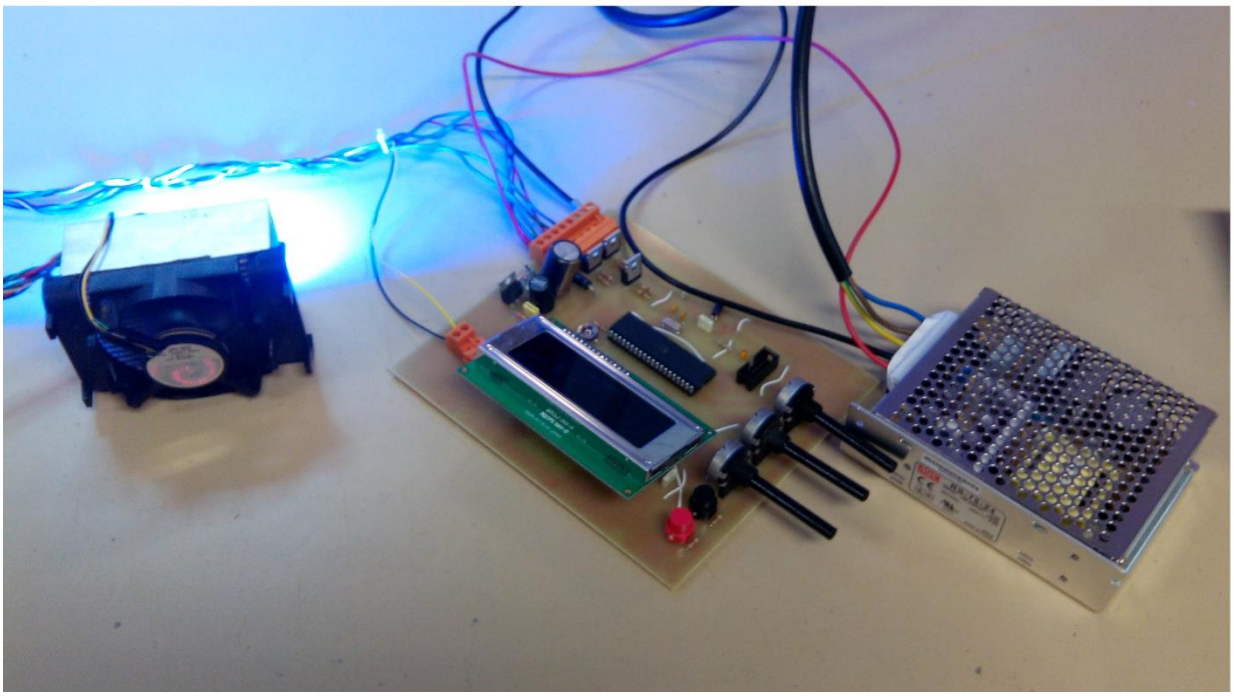


Figure 12 Montage complet du luminaire

7. Avenir du projet

On pourrait effectuer plusieurs améliorations du projet, mais la principale serait faire une carte plus petite, mais aussi mieux conçue, car sur celle-ci nous avons des composants qui pose problème avec l'écran LCD, mais aussi certain ce superpose.

En second point, nous pourrions nous penché vers la programmation, car par manque de temps je n'ai pu faire mon programme de menus. Mais aussi celui de déporter la commande des couleurs vers une application pour smartphone afin de pouvoir les contrôler à distance.

On pourrait aussi envisager de mettre en boîte afin de voir la diffusion de la lumière lorsqu'elle sera « prise aux pièges » dans une boîte. Car il ne faut pas oublier que le premier but du luminaire est d'éclairer un endroit ou pièce.

Conclusion

Maintenant que le projet est fini, nous pouvons donc conclure ce projet que nous avons mis en œuvre.

On peut dire que ce projet était un projet très instructif vu qu'ils y avaient beaucoup d'aspects à prendre en compte, et tout cela en partant de zéro. En étant seul, j'ai appris beaucoup sur la gestion du temps, ainsi que devoir travailler des heures supplémentaires afin que les projets soient finis dans les temps prévu.

Toutes les applications électroniques ont permis de montrer tous les aspects de la formation GEII, car elle comprend de la petite électronique, de l'électronique de puissance et aussi de la programmation.

Les problèmes rencontrés sont encourageant dans le sens où ils ne sont que de petits ajustements ou des erreurs de câblages. Et donc que les problèmes rencontrés sur d'autres projets ont été pris en compte et donc non reproduit durant celui-là.

Mots Clefs

PWM : Pulse-Width Modulation ou modulation par largueur d'impulsion

LED : Light-Emitting Diode ou une diode électroluminescente est un composant capable d'émettre de la lumière avec de faibles courants et donc très économique.

Microcontrôleur : Ceci est un composant programmable, il est composé d'un microprocesseur ainsi que de la mémoire et plusieurs interfaces comme des entrées analogiques.

Synthèse additive : Principe physique de superposer des couleurs pour en obtenir une autre.

Index des illustrations

Annexes

Schéma de commande :

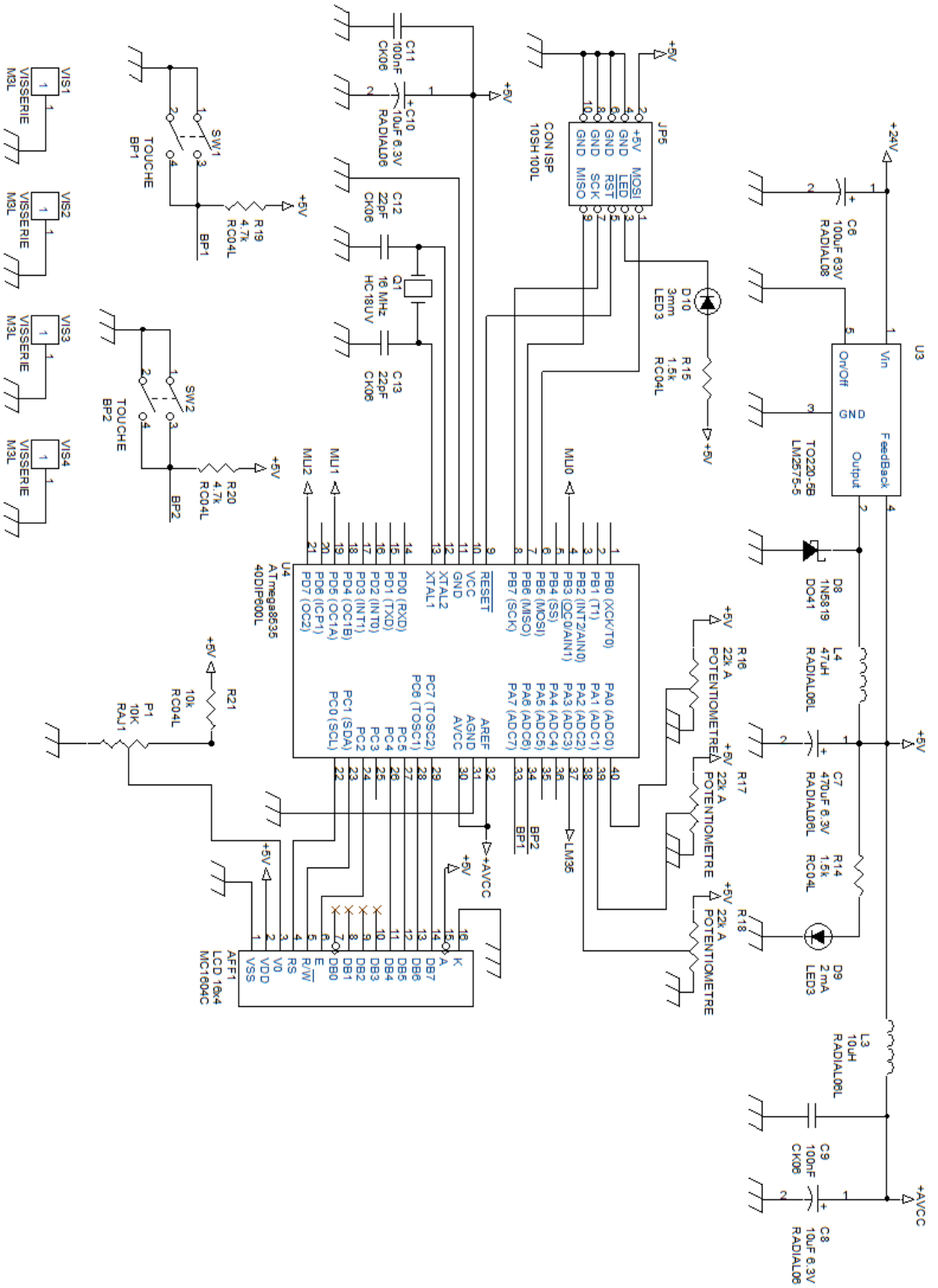
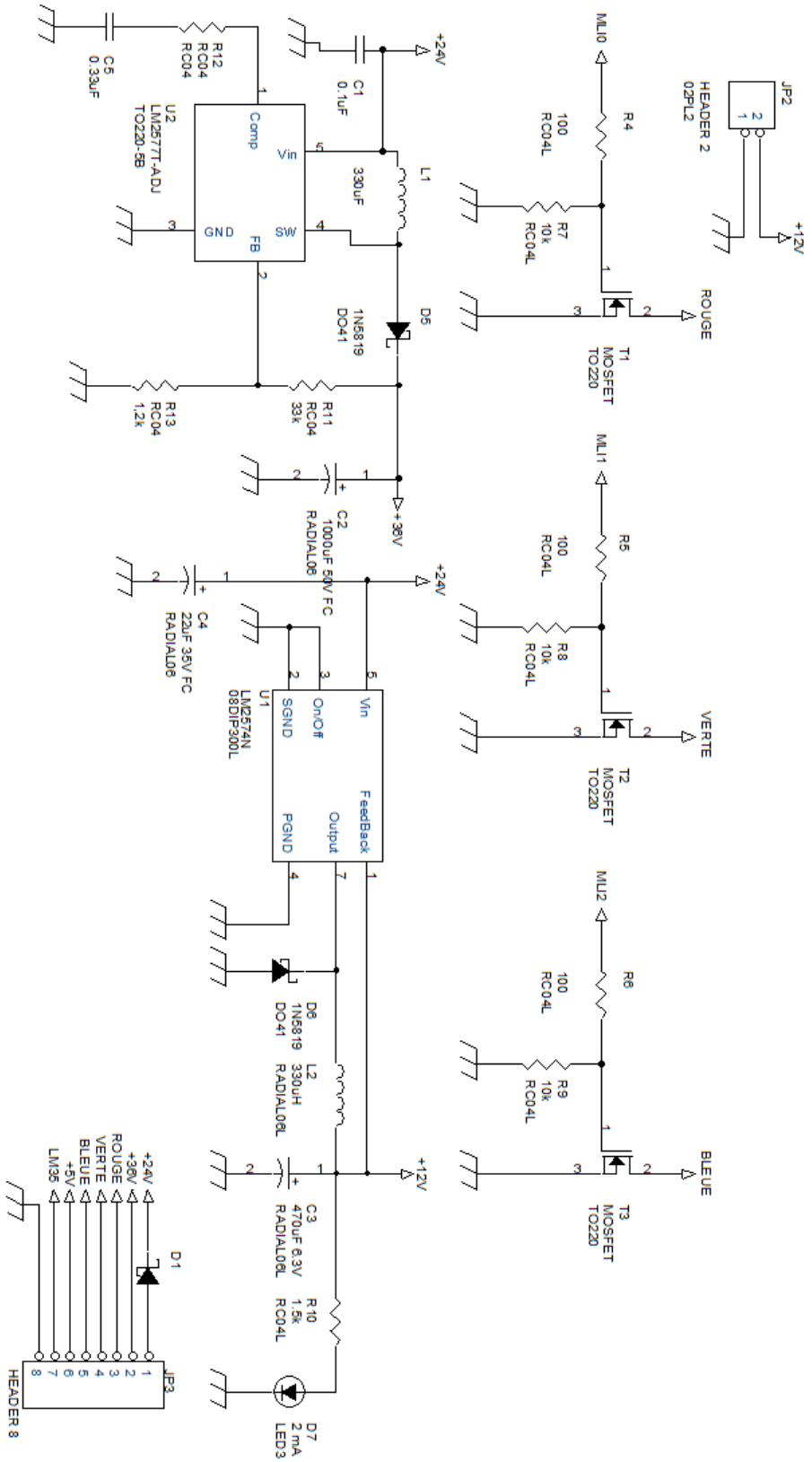
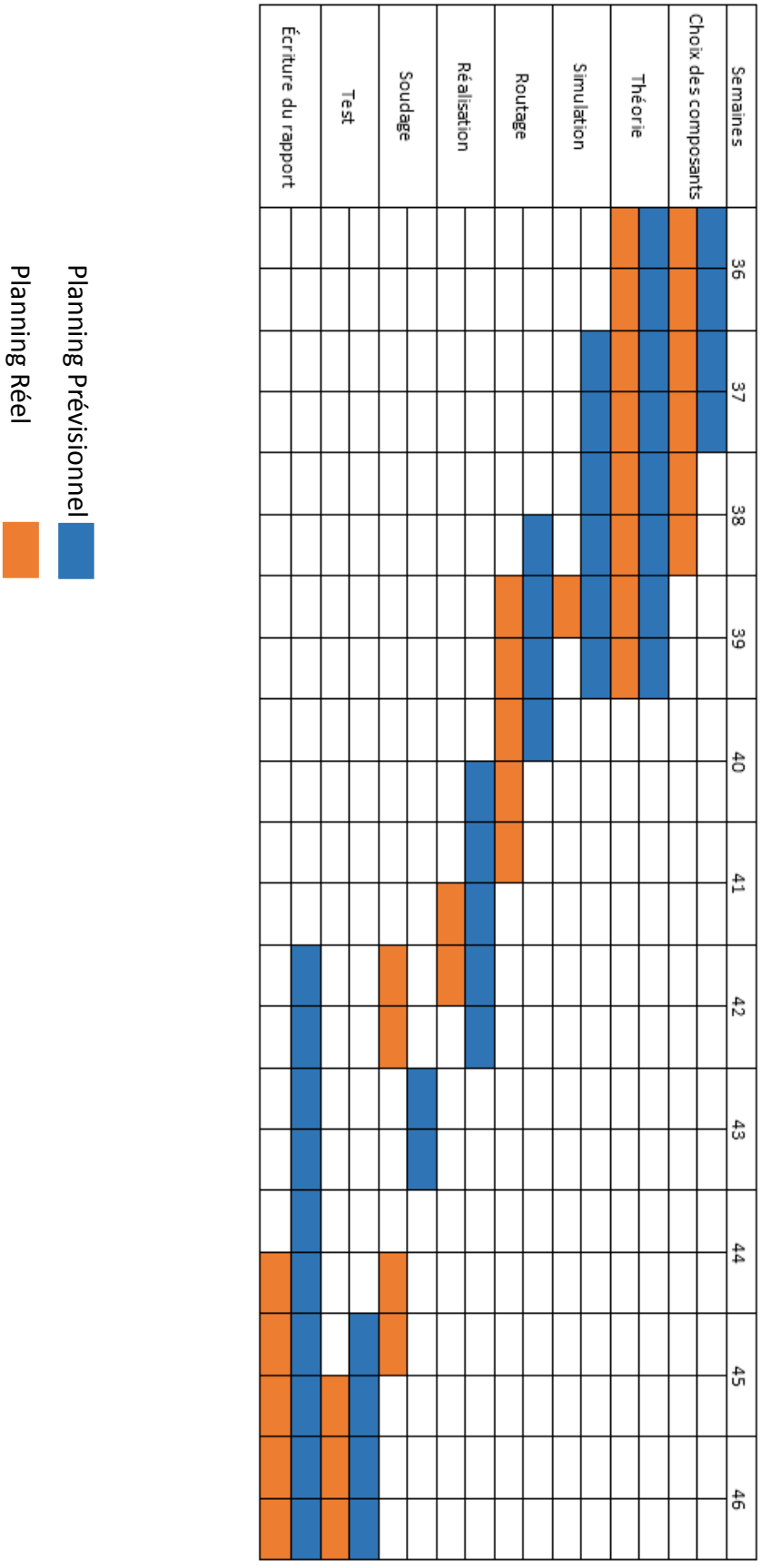


Schéma de puissance :



Planning :


Semaines
36
37
38
39
40
41
42
43
44
45
46
Choix des composants
Théorie
Simulation
Routage
Réalisation
Soudage
Test
Écriture du rapport
Planning Prévisionnel
Planning Réel

Programme entier :

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

Project :
Version :
Date    : 06/11/2014
Author  :
Company :
Comments:


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

#include <mega8535.h>

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

#define ADC_VREF_TYPE 0x20

int pota, potb, potc;
unsigned char tampon[20];
int i;
int BP;


// Read the 8 most significant bits
// of the AD conversion result
```

```
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here

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

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

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x08;

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

// Port D initialization
// Func7=Out Func6=In Func5=Out Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=0 State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xA0;

// Timer/Counter 0 initialization
// Clock source: System Clock
```

```
// Clock value: 16000,000 kHz
// Mode: Phase correct PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x61;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 16000,000 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x81;
TCCR1B=0x01;
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: 16000,000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x61;
TCNT2=0x00;
OCR2=0x00;

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

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

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

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
SFIOR&=0xEF;

// LCD module initialization
lcd_init(16);
delay_ms(1000);
while (1)
    {
    delay_ms(10);
        pota=read_adc(0);
         potb=read_adc(1);
        potc=read_adc(2);
         OCR0=pota;
         OCR1A=potb;
         OCR2=potc;
         lcd_gotoxy(0,0);
         lcd_putsf("Luminaire RGB");
         lcd_gotoxy(0,1);
         sprintf(tampon,"Rouge=%d",pota);
         lcd_puts(tampon);
         lcd_gotoxy(0,2);
         sprintf(tampon,"    Vert=%d",potb);
         lcd_puts(tampon);
        lcd_gotoxy(0,3);
        sprintf(tampon,"    Bleu=%d",potc);
        lcd_puts(tampon);
    };
}
```