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

Gestion des feux de signalisation pour une voiture électrique

BIN ROSLI Mohd Nazrin
LAMBERT Florian
Groupe Q2
Année 2010-211

Enseignants

LEQUEU Thierry
PAPAZIAN Patrick

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

Gestion des feux de signalisation pour une voiture électrique

BIN ROSLI Mohd Nazrin
LAMBERT Florian
Groupe Q2
Année 2010-211

Enseignants

LEQUEU Thierry
PAPAZIAN Patrick

Sommaire

Introduction

Au cours du troisième semestre nous devons réaliser un projet d'étude et réalisation. Nous avons choisi parmi plusieurs sujets proposés et nous avons opté pour un projet en rapport avec le karting électrique. Ce projet a pour but de créer un système d'éclairage, pour le kart, comprenant des feux clignotants, des feux avant progressifs en fonction de la lumière ambiante et des feux de stop progressifs en fonction de l'appui sur la pédale de frein.

Dans un première temps nous étudierons des projets similaires existants puis nous essayerons d'améliorer le projet. Nous créerons une carte électronique afin de commander les différents composants et nous réaliserons un programme permettant le fonctionnement voulu.

1. Présentation du projet

Le projet consiste a créée une carte électronique permettant le contrôle de feux de signalisation avant et arrière pour un Kart électrique. Nous allons donc réaliser des feux avants, des feux arrières, des feux clignotants, des feux avant progressifs en fonction de l'intensité de la lumière ambiante et des feux de stop progressif en fonction de l'appui sur la pédale de frein. Nous examinerons dans un premier temps les modèles de carte déjà existants puis nous apporterons des modifications afin de la rendre plus performantes. Nous programmerons notre carte avec l'ancienne version du programme afin de vérifier le bon fonctionnement de la carte puis nous apporterons les modifications nécessaires en fonction de notre carte. 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 système d'éclairage pour un kart électronique.
- Objectifs :
  - Création de différents feux :
  - Feux de stop progressifs en fonction de la pédale de frein
  - Feux avant progressifs en fonction de la lumière ambiante (feux de croisement et feux de route)
  - Feux clignotants et feux de détresse
  - Feux de recul
- Contraintes :
  - Alimentation du microcontrôleur 0/5v
  - Alimentation de la carte 0/12v
  - Programmation du microcontrôleur
  - Réalisation du circuit de commande
  - Réalisation de la carte électronique
  - Programmation du microcontrôleur
  - Gestion des différents interrupteurs contrôlant les feux

1.2. Schéma fonctionnel

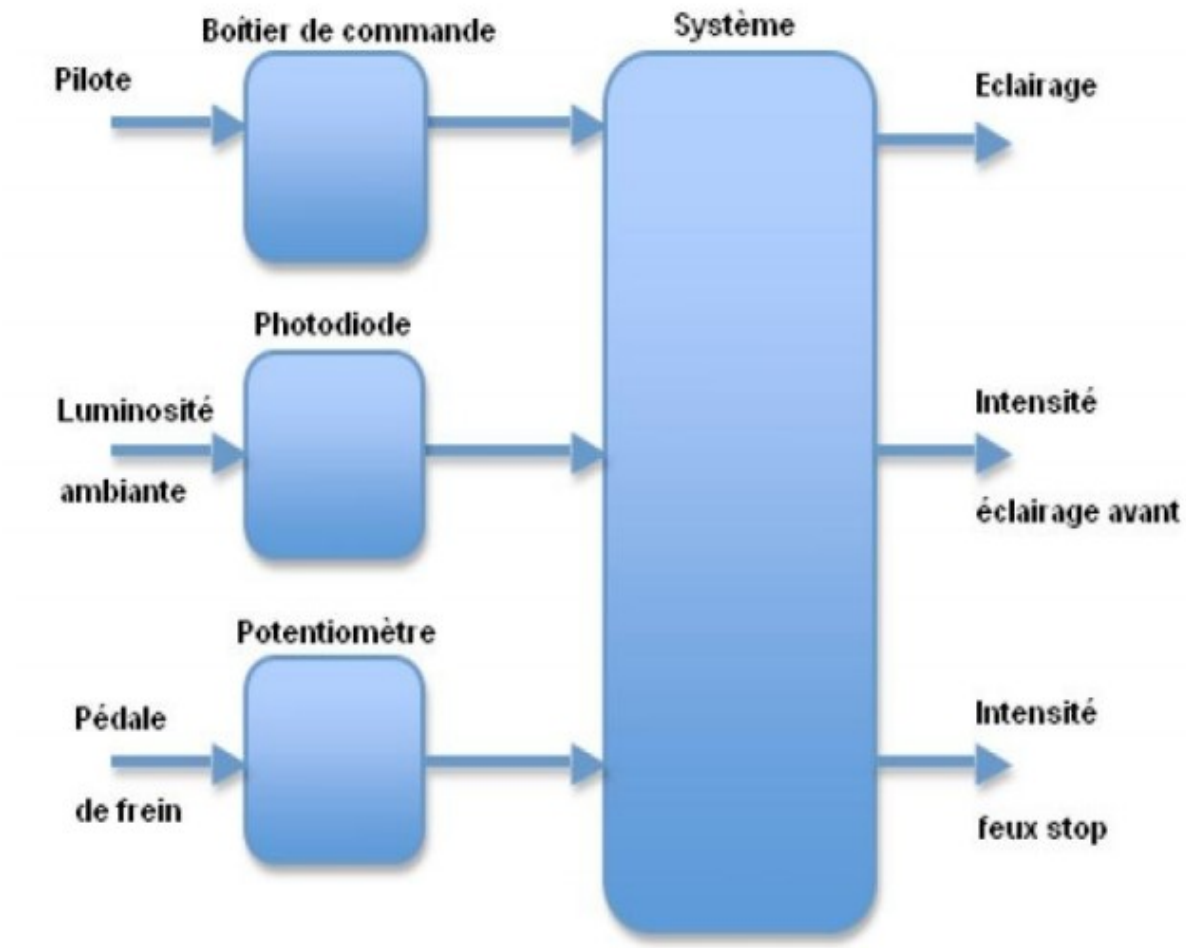


Illustration 1: Schéma fonctionnel de niveau 2

1.3. Planning prévisionnel

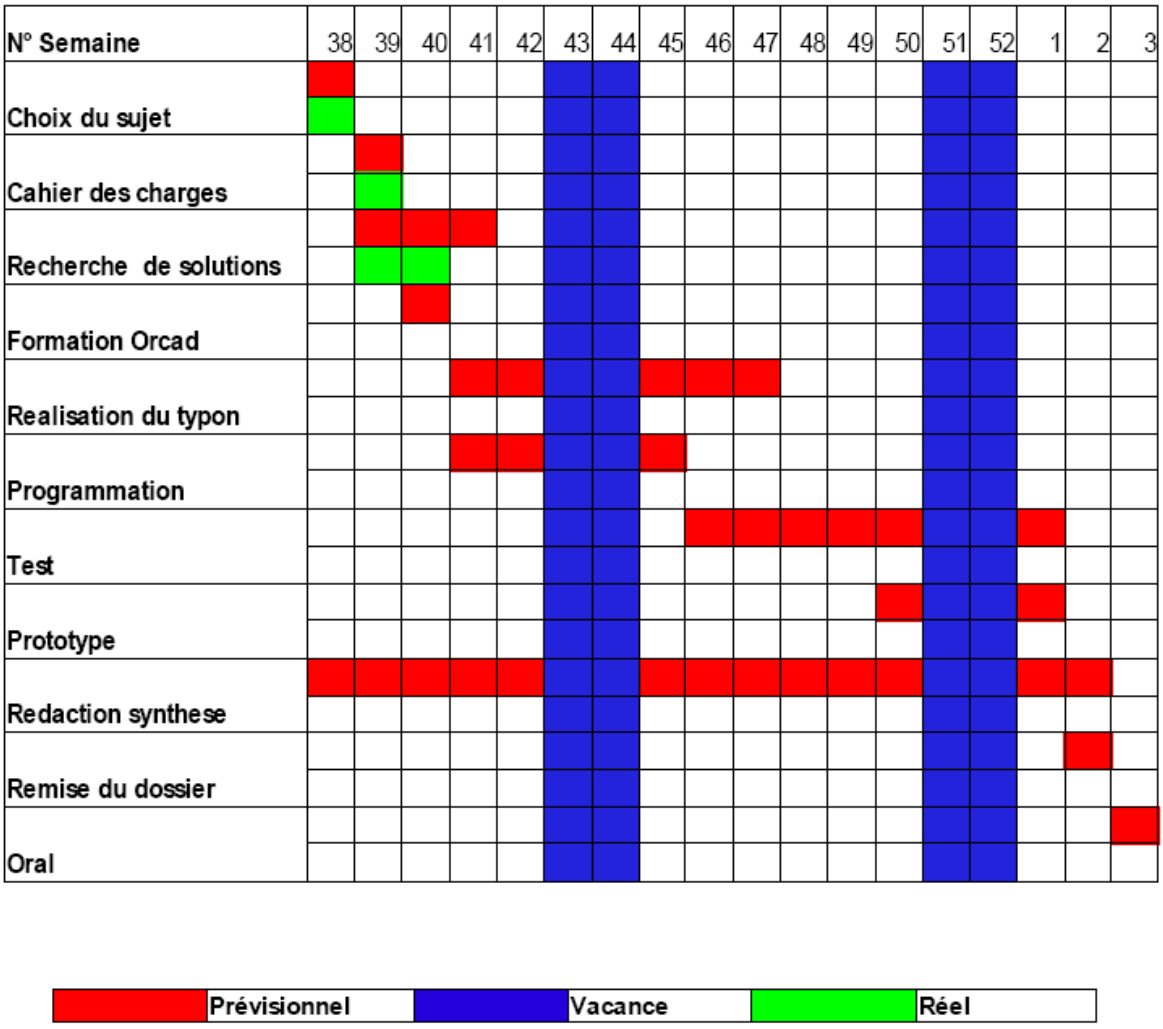


Illustration 2: Planning prévisionnel

2. Etude théorique

2.1.1. Les lampes

Dans les précèdent sujet des lampes à LED ont été utilisées afin de permettre l'éclairage du kart. Ces lampes à LED ont une tension d'alimentation de 12V, ce qui correspond à la tension d'alimentation de la carte électronique (par le kart), et une puissance de 1W. Les lampes à LED nous permettent aussi de varier les couleurs afin d'avoir des feux de la couleur correspondant à la demande. De plus les lampes à LED ont une faible consommation d'énergie, ce qui permet de préserver l'autonomie de la batterie. Pour commander les lampes nous utiliserons des transistors MOSFET. Nous avons 20 lampes à disponibilité .

2.2. Les transistors

Pour commander les lampes nous avons choisi d'utiliser des transistors MOSFET[1] montés de la façon suivante.

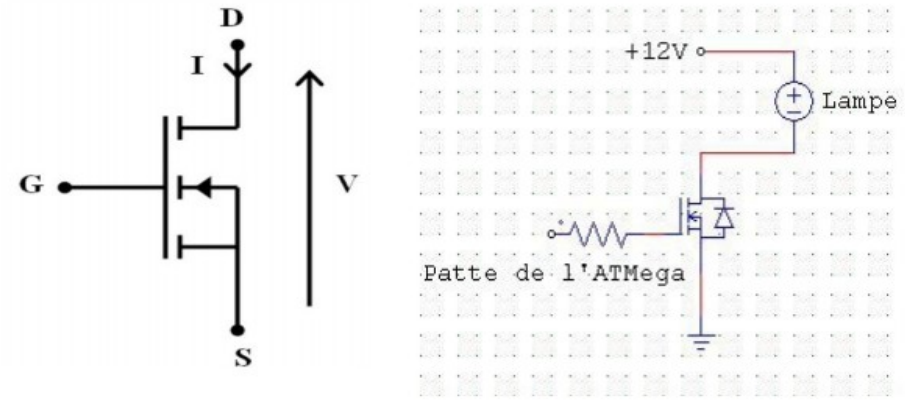


Illustration 3: Schéma du transistor MOSFET

Ce montage permet de commander les lampes en activant ou non les sorties de l'Atmega 8535. En effet lorsque la sortie de l'Atmega 8535 est active, la tension appliquer à la grille du transistor ferme le contact entre le drain et la source, ce qui relie la lampe entre la masse et le +12V. La lampe est donc alimentée lorsque l'on active la sortie de l'Atmega 8535 .Au contraire quand la sortie du microcontrôleur est inactive le contact entre le drain et la source est ouvert la lampe n'est plus reliée à la masse elle n'est donc pas alimentée et éteinte.

Pour le choix des transistors nous devons faire attention à l'intensité qu'ils doivent pouvoir supporter.

I=P/U=1/12=20mA

Nous devons donc choisir des transistors pouvant supporter 20mA. Nous avons choisi des transistors MOSFET qui sont compatibles avec notre montage. Un transistor servira de commande à 2 lampes. Il nous faut donc 10 transistors.

2.3. Atmega

Afin de commander les transistors et le reste de la carte électronique nous utiliserons un microcontrôleur Atmega 8535. Ce microcontrôleur offre 4 ports de 8 broches paramétrables soit en

1 En anglais : Metal Oxide Semiconducteur Field Effet Transistor

entrée soit en sortie selon notre souhait. Ce qui nous permet de commander notre carte facilement. De plus le microcontrôleur Atmega 8535 offre 3 sorties pouvant être réglées en modulation de largeur d'impulsion (MLI) qui vont nous permettre de régler la tension d'alimentation des feux avants et arrières. Ces sorties vont nous permettre de faire varier l'intensité lumineuse de nos feux avants et de nos feux stops.

2.4. L'alimentation 12/5V

La carte électronique sera alimentée par la batterie du kart qui fourni une tension continue de 12V. Cette tension nous permettra d'alimenter les lampes sans changement. 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 LM2575. Nous avons vu en cours de MC- ET2 comment réaliser cette alimentation à découpage.

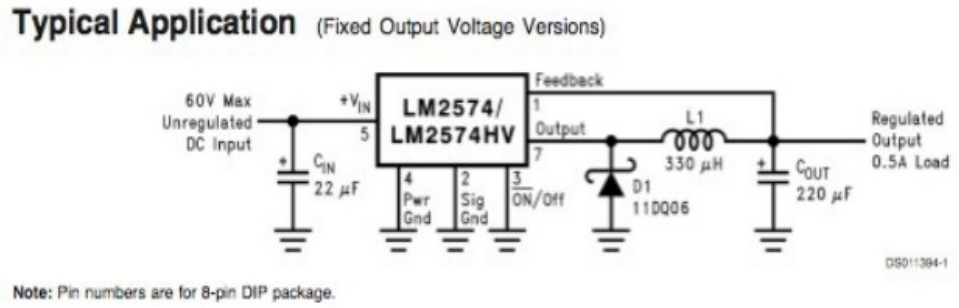


Illustration 4: Schéma du régulateur LM2575

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

2.5. Photorésistante/Potentiomètre

Afin de pouvoir commander les feux de façon progressif en fonction de la lumière ambiante nous utiliserons la fonction MLI du microcontrôleur ainsi qu'une photorésistante qui permet la détection de cette lumière. La photorésistante nous fourni une tension qui varie selon la luminosité.

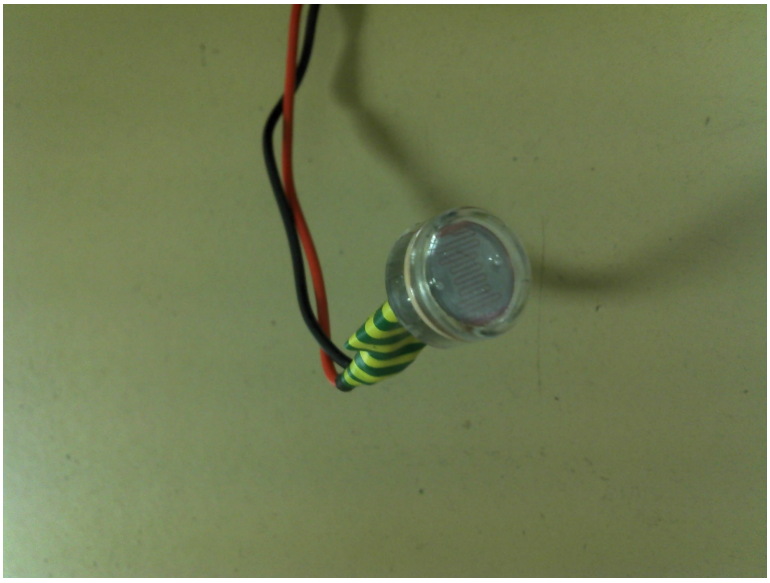


Illustration 5: Photorésistance

Afin de pouvoir commander les feux stops en fonction de l'appui sur la pédale de frein nous utiliserons aussi la fonction MLI du microcontrôleur ainsi qu'un potentiomètre mécanique placé sur la pédale. Ce potentiomètre nous fournira une tension variable selon la pente de la pédale de frein. C'est cette variation de tension qui nous permettra de faire varier l'intensité lumineuse des feux de stops.

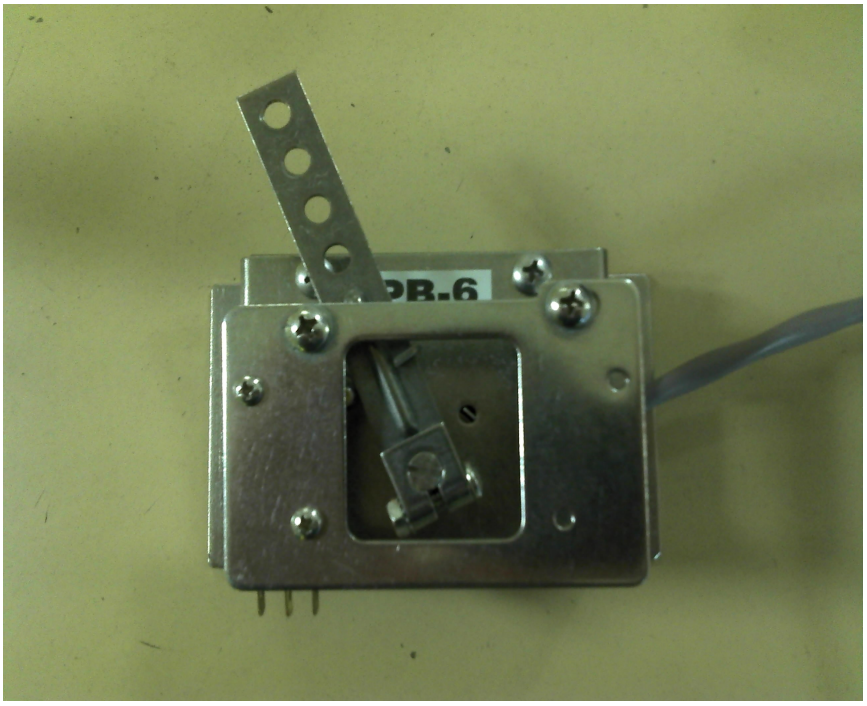


Illustration 6: Pédale de frein avec potentiométre

3. 3 Réalisation

3.1. La maquette

Grâce au groupe précèdent nous avons a disposition un prototype basé sur une maquette en bois permettant le test la carte électronique ainsi que le bon fonctionnement des lampes a LED. Cette maquette représente le châssis du kart sur lequel sera fixé 10 lampes à l'avant(5gauches, 5droites), 10 lampes à l'arrière (5 gauches, 5 droites) et différents composants du montage tel que le dispositif permettant l'alimentation par la batterie.

Illustration 7: Photo de la maquette

Afin de commander les différents feux nous avons à notre disposition un prototype de boîtier de commandes permettant la commande des différentes entrées. Ce boitier comporte :

-Un BP[2] d'alimentation générale ON/OFF

-Un BP trois positions commandant les clignotants(gauche,droite,repos)

-Un BP commandant les warnings (ON/OFF)

-Un BP permettant le choix du mode d'éclairage (automatique/manuel)

-Un BP trois positions commandant les différents feux (position/ route)

2 Bouton Poussoir

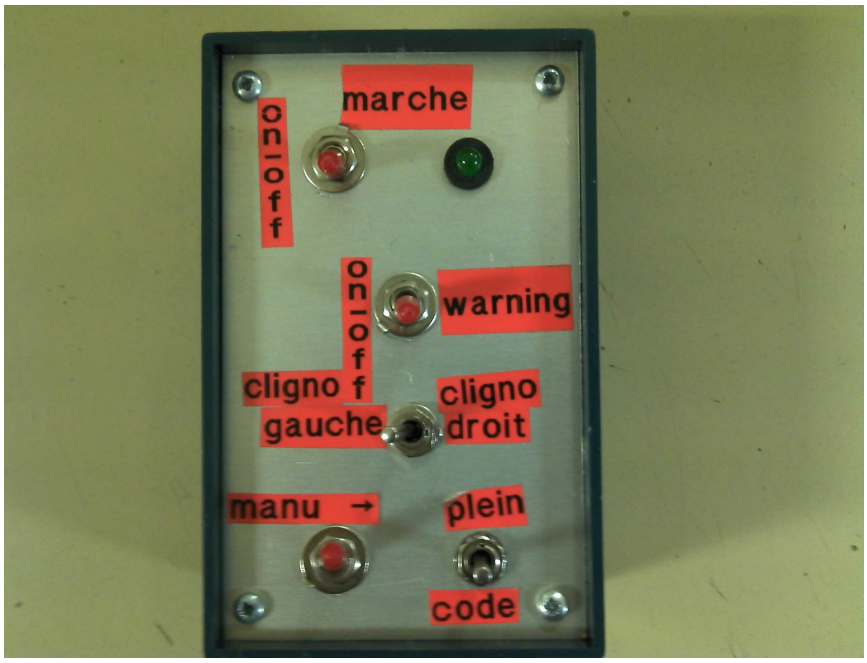


Illustration 8: Photo du boîtier

Le bornier des entrées, sur la carte électronique, était, sur les projet précèdent un bornier 15 broches.

Dans notre étude nous avons vu que nous avions besoin de 12 broches, c'est pourquoi nous avons changé le bornier. Par conséquent nous avons du refaire un câble permettant la liaison entre le boitier et la carte électronique

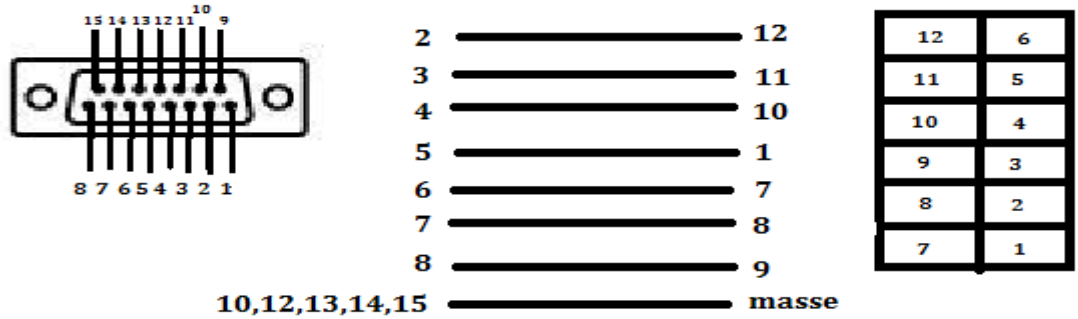


Illustration 9: Equivalence des broches

Boîtier Carte

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, on le transforme en un fichier qui s'appelle Netlist qui fait une liaison avec tous les composants que nous devons utiliser. Pour utiliser ce logiciel, il faut imposer d'abord tous les composants et puis, mettre les liaisons entre eux.

3.3. Schéma et ordinogramme 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 le commande de Atmega a besoin d'une tension continue de 5 Volts à partir de la tension continue de 12 Volts qui sort de la batterie, alors nous avons décidé d'utiliser ce type d'alimentation.

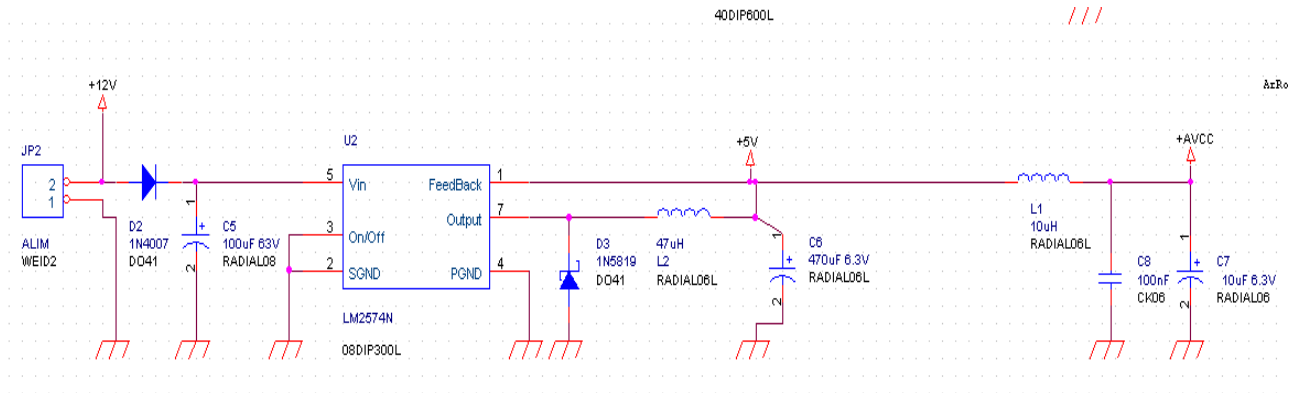


Illustration 10: Schéma alimentation à découpage

3.3.2. Atmega

Illustration 11: Photo de l'Atmega

3.3.3. Assignation des pattes.

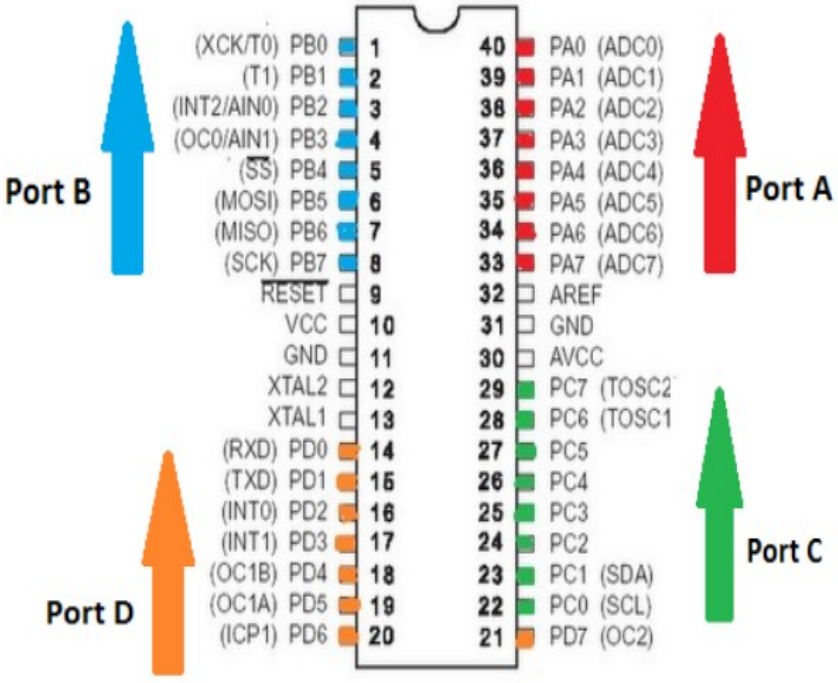


Illustration 12: Assignation des broches de l'Atmega

On placera les sorties sur le port D et les entrées sur le port C. Contrairement au projet précèdent on mettra les feux avant progressifs sur la sortie 3 du port B (Voir Programme).

3.3.4. les sorties

Afin de commander les lampes par les connecteurs MOLEX , nous avons besoin de transistors qui fonctionnent à partir de l'état de sortie des pattes de l'Atmega. Ainsi, 10 transistors sont utilisés et un transistor est relié à 2 broches de connecteur (autrement dit, nous avons utilisé un seul transistor pour commander deux lampes en même temps)

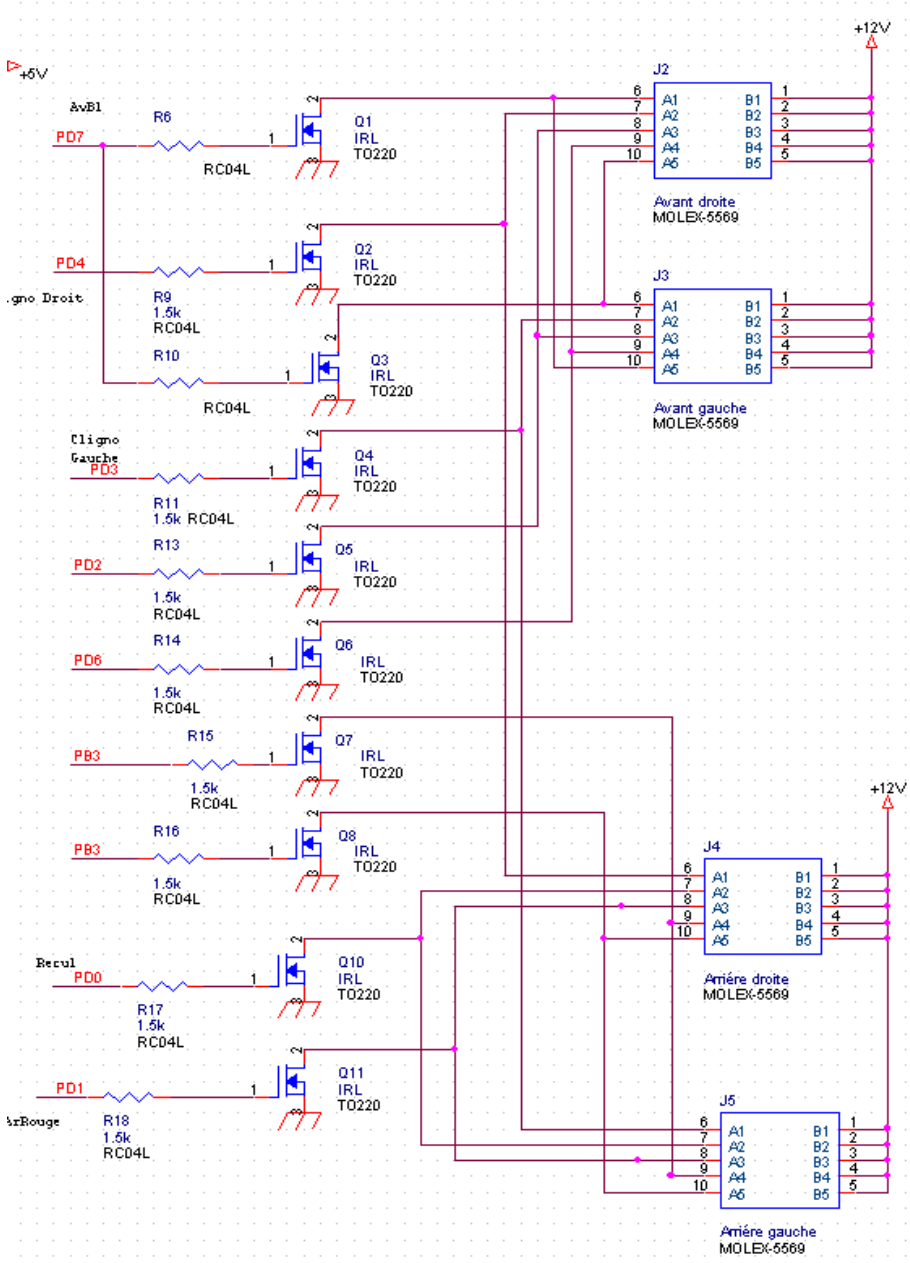


Illustration 13: Schéma des transistors

3.3.5. Les entrées

Pour réaliser notre carte, nous avons besoin 8 entrées sur le schéma électrique :

- Photorésistante
- Potentiomètre de frein
- Interrupteur général
- Feux de détresse (ou WARNING)
- Clignotants (2 entrées)
- Mode manuel / automatique
- Feux route / positon

Donc, un connecteur de 5 broches est utilisé pour les entrées variables (dans notre cas, les entrées variables sont le photorésistante et le potentiomètre de frein) et un connecteur DB12 femelle pour les entrées tout ou rien. Nous avons décidé aussi de changer les broches (de 15 broches à 12 broches) pour les entrées tout ou rien car nous n'avons besoin que 12 broches pour les entrées. Cela nous a permis de gagner la place sur la carte. De plus, le passage du courant du boîtier vers l'Atmega est assuré par le pont de résistance.

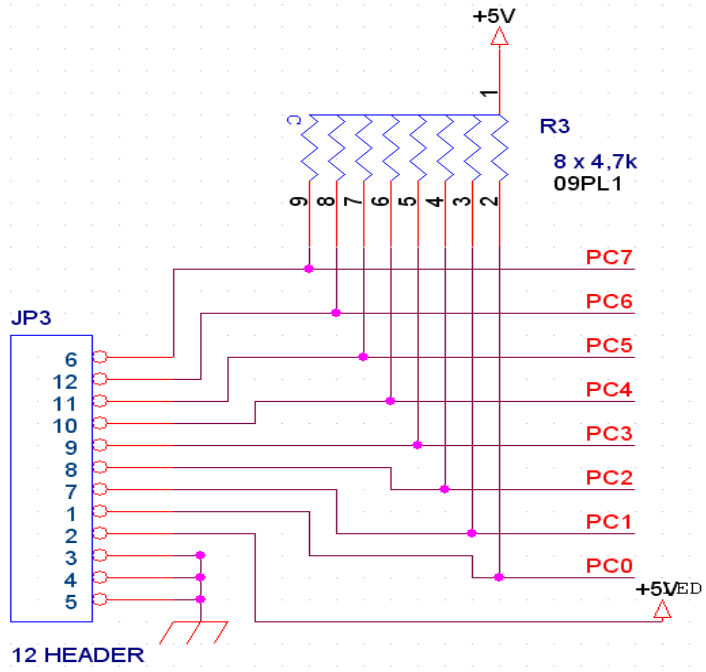


Illustration 14: Schéma des entrées

3.4. Layout Plus

3.4.1. Typon

Après avoir terminé le schéma électrique sur Capture, nous avons créé la Netlist et nous l'envoyé vers l'Orcad Layout Plus pour réaliser le typon afin d'utiliser sur la circuit imprimé.

3.4.2. Contraintes

Pour réaliser le typon, nous avons pris le schéma électrique des précédents projets et il y avait beaucoup de modifications à faire afin d'améliorer les performances de notre éclairage. Ceci explique le temps important que nous avons passé pendant la séance d'étude et réalisation.
Dans cette étape, il faut bien limiter la taille de la carte pour que nous puissions la mettre dans la boîtier. 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, chaque transistor est positionné près de connecteur correspondant et tous les connecteurs sont placés sur les bords de la carte pour faciliter les câbles électriques. Grâce au rapport des anciens étudiants, nous avons remarqué qu'il fallait placer le quartz plus près du microcontrôleur afin d'avoir le bon fonctionnement de l'Atmega. Nous avons étudié aussi en cours Physique (CEM) que le plan de masse est une bonne solution pour éviter le CEM, cela explique l'utilisation de la plan de masse dans notre carte.

3.4.3. Résultat final

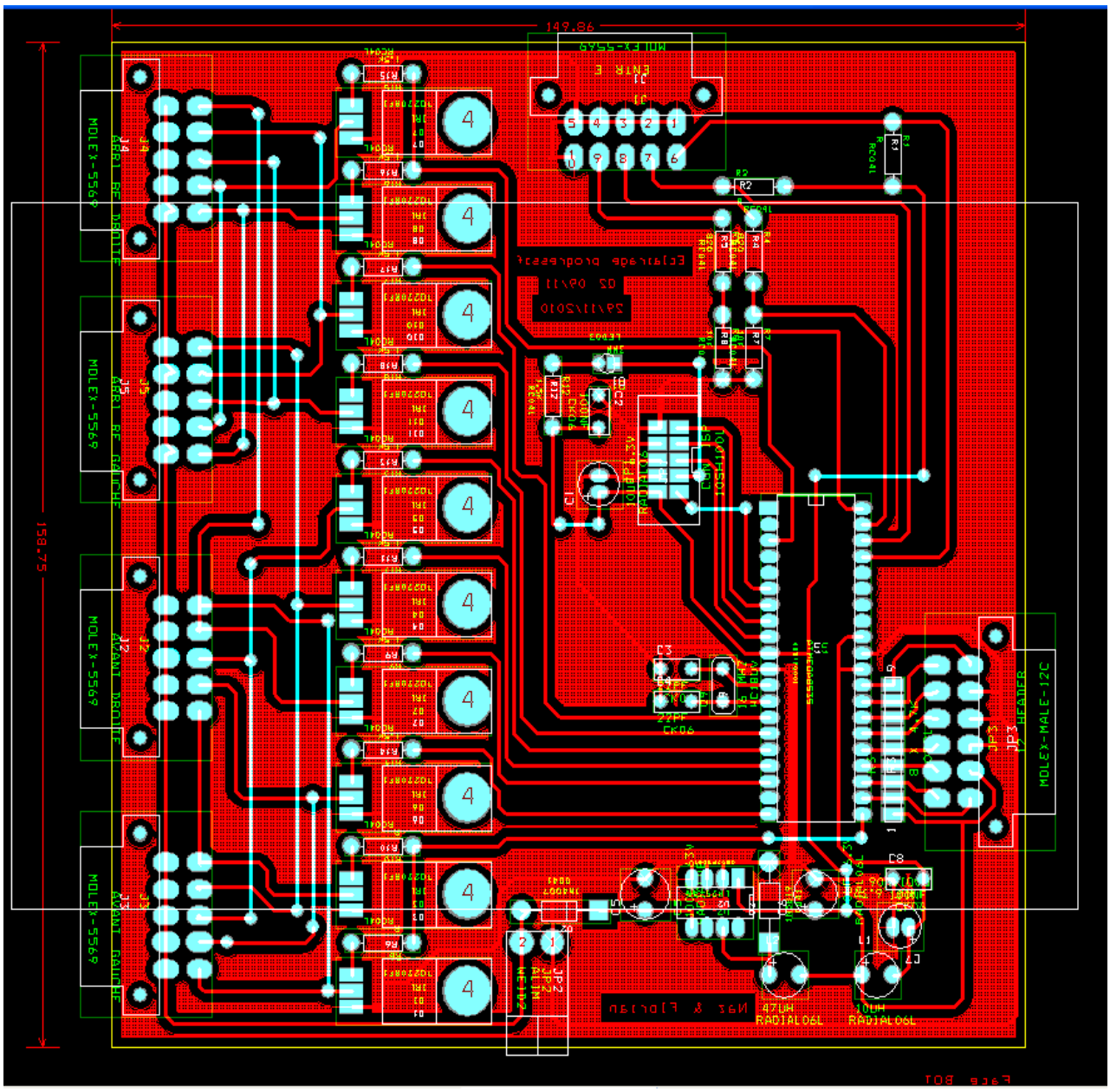


Illustration 15: Typon final

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 LM2575N 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'Atmega8535 dans l'onglet chip de la fenêtre C Compiler. Puis monter le clock à 8Mhz 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 »

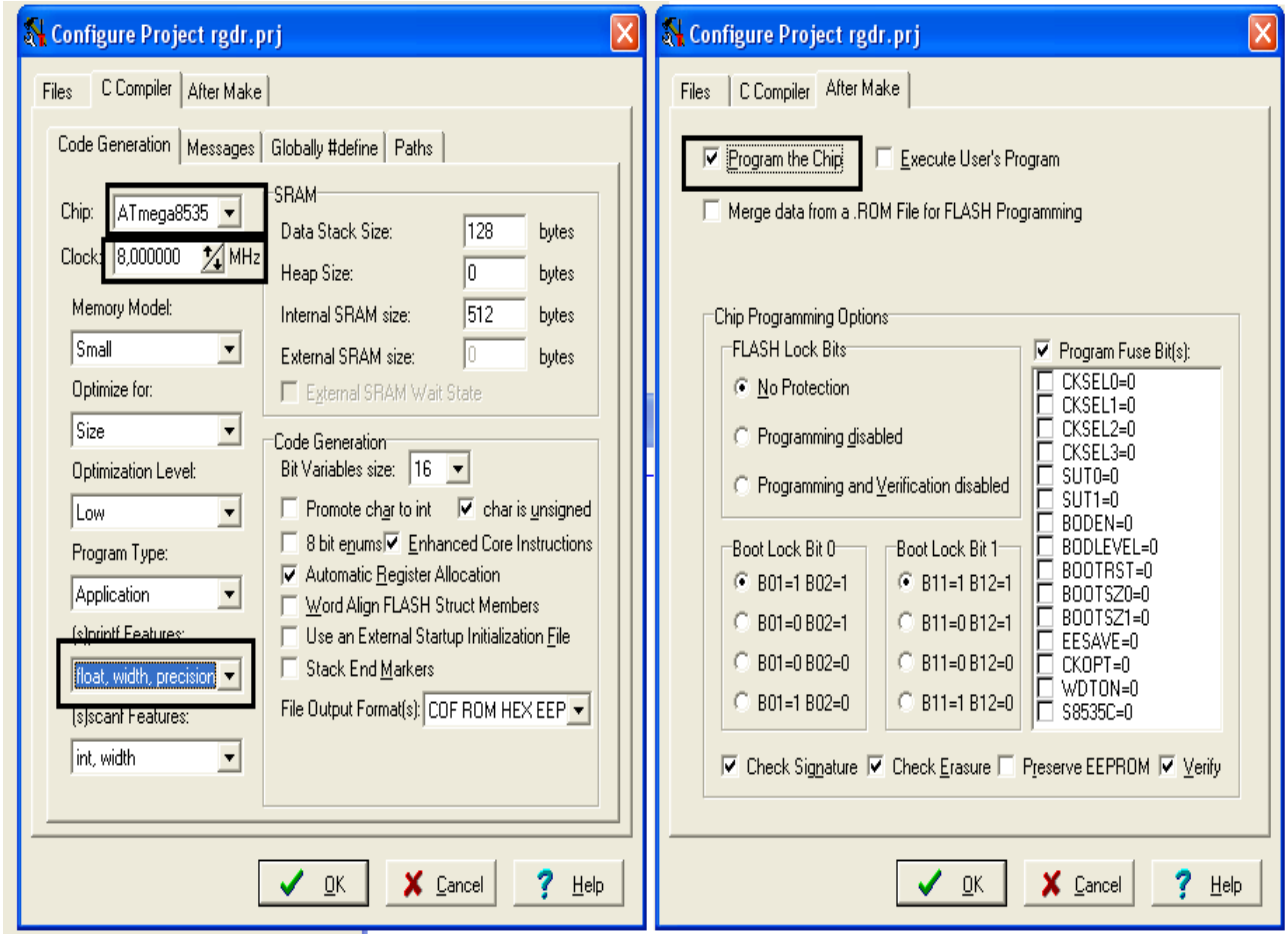


Illustration 16: Configuration de CodeVisionAVR

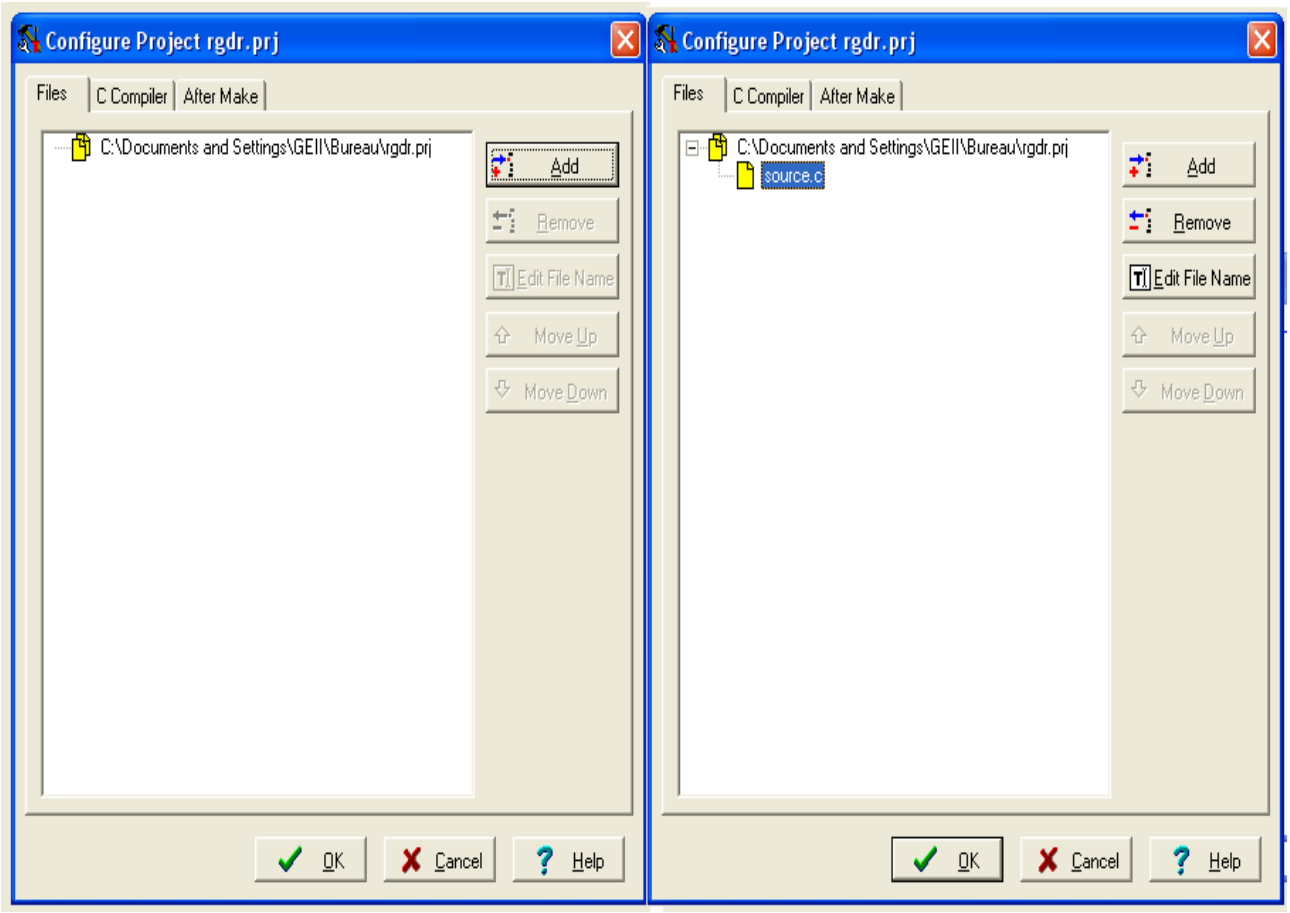


Illustration 17: Configuration de CodeVisioAVR 2

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 repris un programme simple.

```
while(1)
{
        delay(1000) ;
        if (Lampe= =1)
        Lampe=0 ;
        else Lampe=1 ;
}
```

Ce programme permet de faire clignoter toutes les lampes quel que soit l'état des entrées avec un délai de 1 sec. Grâce à ce programme, nous allons voir si toute les sorties sont correctement connectées.

4.3. Programmation

4.3.1. Ordinogramme

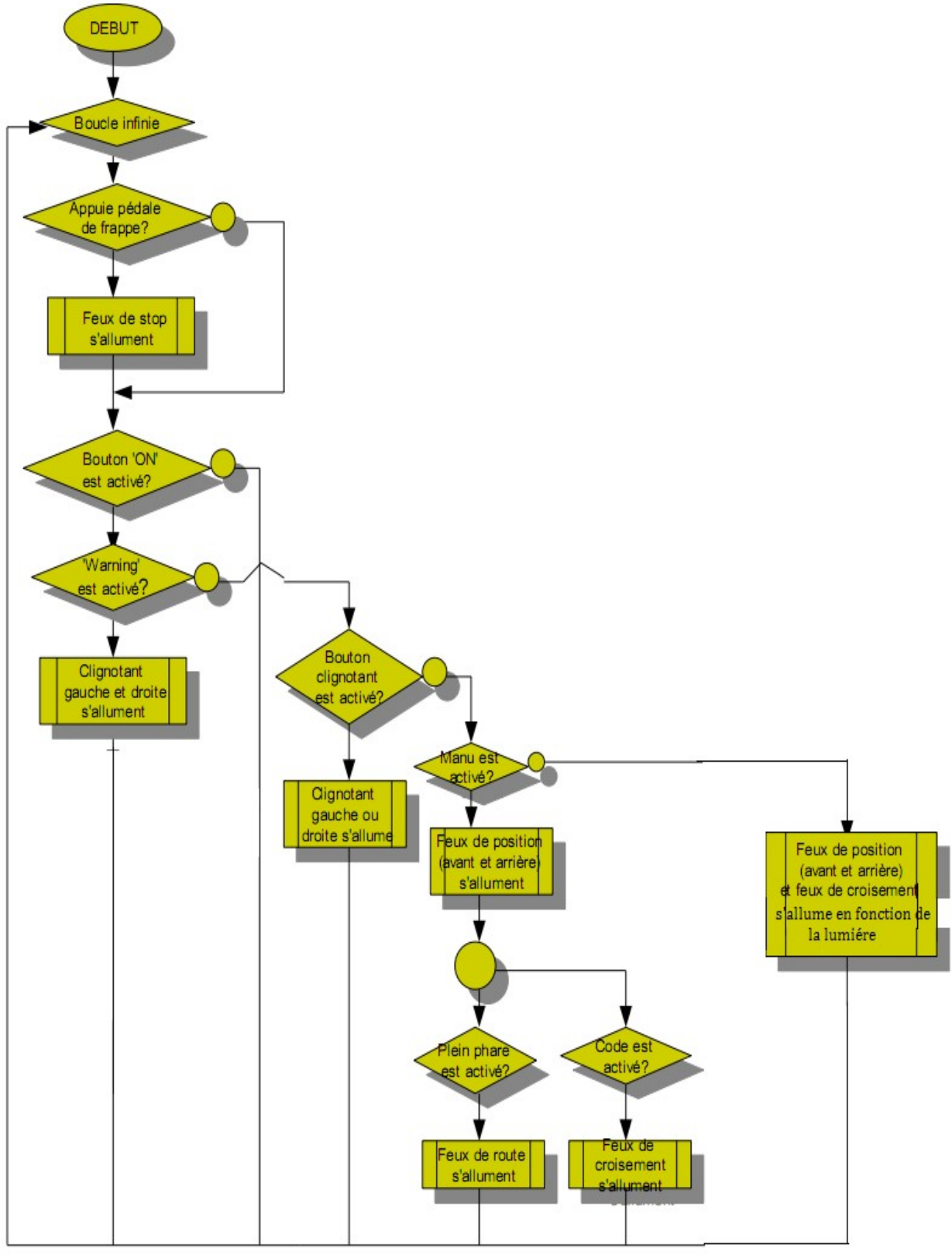


Illustration 18: Ordinogramme du programme

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

```
//Déclaration des Sorties
#include<delay.h>
#define feux_recul PORTD.0
#define feux_arriere PORTD.1
#define feux_avant PORTD.2
#define cligno_g PORTD.3
#define cligno_d PORTD.4
#define feux_stop PORTB.3
#define feux_route PORTD.7
#define croisement PORTD.6

//Déclaration des Entrées
#define BP_cligno_g PORTC.0
#define BP_cligno_d PORTC.1
#define BP_warning PORTC.2
#define BP_mode PORTC.3       //  AUTO/MANU
#define BP_feux_route PORTC.4
#define BP_feux_crois PORTC.5
#define BP_eclairage PORTC.6   //  ON/OFF
```

Il faut ensuite configurer les 4 Ports de l'Atmega 8535 afin de définir les entrés et les sorties.

Chaque port est configurer sur 8 Bits. En mettant 1 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 entré
PORTA=0x00;
DDRA=0x00;

//Initialisation du port B → PortB3 en sortie, le reste en entrés
PORTB=0x00;
DDRB=0x08;


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

//Initialisation du port D → Port D en sortie
PORTD=0x00;
DDRD=0xFF;
```

Il fait ensuite déclarer et configurer nos compteurs, le MLI et les différents registres de l'Atmega 8535

```
//Compteur 0
TCCR0=0x00;
TCNT0=0x00;
```

```
OCR0=0x00;

//Compteur 1
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BH=0x00;

//Compteur 2
ASSR=0x00;
TCCR2=0x64;
TCNT2=0x00;
OCR2=0x00;

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

TIMSK=0x00;  //initialisation des compteurs

ACSR=0x80;
SFIOR=0x00;
```

Puis on rentre le programme voulu.

Exemple:

```
if(BP_warning==0)
                {
                        cligno_d=!cligno_d;      //On fait clignoter le clignotant droit
                        cligno_g=!cligno_g;      //On fait clignoter le clignotant gauche
                        delay_ms(500);
                }
```

4.3.2. Programmation des feux avant progressif

La photorésistante nous permet d'obtenir un tension en fonction de la lumière ambiante. Les données issues de cette photorésistante sont des données analogiques. C'est pourquoi nous devons initialiser le Convertisseur Analogique Numérique(CAN) du microcontrôleur.

```
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
```

Grâce aux tests des projets précédents nous avons pu avoir un tableau de valeurs correspondant aux valeurs données par la photorésistante.

Valeur Analogique	Valeur Numérique
3,4V	173
4,2V	214
5V	255

Grâce à ce tableau nous pouvons programmer l'Atmega 8535 afin d'avoir un éclairage plus fort lorsque la luminosité est faible.

```
Ne=read_adc(2);
                if(Ne>125)
                {
                        PORTD.7=0;
                        PORTD.6=0;
                        OCR2=173;
                        feux_arriere=1;
                        feux_avant=1;
                }
                if(Ne>173)
                {
                        PORTD.7=0;
                        PORTD.6=1;
                        OCR2=214;
                        feux_arriere=1;
                        feux_avant=1;
                }
                if(Ne>220)
                {
                        PORTD.7=1;
                        PORTD.6=1;
                        OCR2=255;
                        feux_arriere=1;
                        feux_avant=1;
                }
                else
                {
                        PORTD.7=0;
                        PORTD.6=0;
                        OCR2=0;
                        feux_arriere=0;
                }
```

La technique est presque la même pour le potentiomètre. Grâce aux tests des précédents projets nous avons pu obtenir un pente.

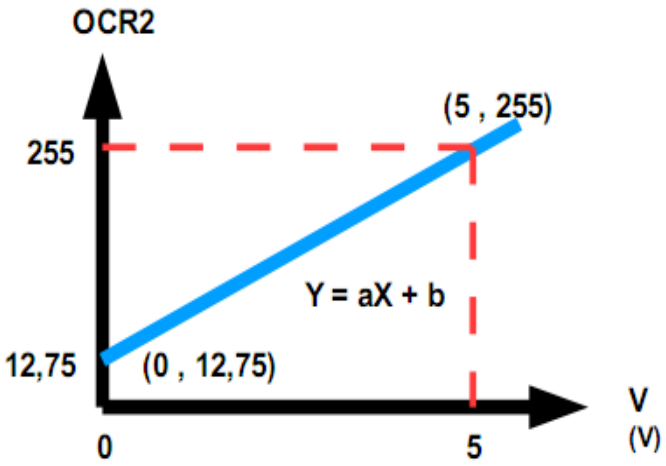


Illustration 19: Graphique correspondant à la varation de la luminosité

On obtient l'équation suivante: OCR2 = 48,5*V + 12,5 avec a = (255-12,75)/(5-0) = 48,5.

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

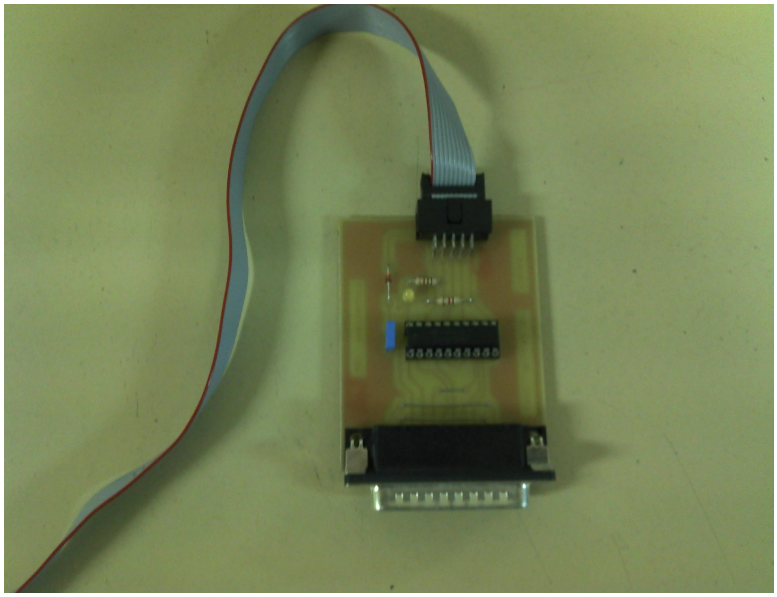


Illustration 20: Connecteur de programmation

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 (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 feux de reculs qui devront s'allumer lorsque le moteur tourne dans le sens de rotation anti-horaire. Enfin il est possible de créer un boîtier permettant d'y insérer la carte et par conséquent éviter toute détérioration de celle-ci lors de l'installation.

5. Liste des composants

NOM	Référence	Empreinte	Valeur	Quantité	Prix Unitaire	Prix Total
Résistances	R9, R11, R12, R13, R14, R15	RC04L	1,5 KΩ	6	0,05 €	0,30 €
	R7, R8	RC04L	10 KΩ	2	0,05 €	0,10 €
	R4, R5	RC04L	820 Ω	2	0,05 €	0,10 €
	R1, R2, R6, R10	RC04L	1 KΩ	4	0,05 €	0,20 €
Condensateurs	C7, C1	RADIAL 06	10 uF	2	0,45 €	0,90 €
	C2, C8	CK 06	100 nF	2	0,51 €	1,02 €
	C4, C3	CK 06	22 pF	2	0,51 €	1,02 €
	C5	RADIAL 08	100 uF	1	0,45 €	0,45 €
	C6	RADIAL O6L	470 uF	1	0,45 €	0,45 €
LED	D1	LED3	3mm	1	0,09 €	0,09 €
Diodes	D2 , D3	D041	1N5819	2	2,00 €	2,00 €
Inductances	L1	10 uH	RADIAL06 L	1	0,73 €	0,73 €
	L2	47 uH	RADIAL06 L	1	0,73 €	0,73 €
Transistors	Q1, Q2, Q3 Q4, Q5, Q6, Q7, Q8,Q10, Q11	TO220	IRL2203N	10	1,73 €	10,73 €
Quartz	Q9	HC18UV	16 MHz	1	0,39 €	0,39 €
Réseau de résistance	R3	09PL1	8 x 4,7 KΩ	1	0,28 €	0,28 €
Micro-	U1	40DIP600L	ATMega	1	5,93 €	5,93 €

Contrôleur			8535			
Régulateur	U2	08DIP300L	LM2574N	1	1,80 €	1,80 €
Lampe LED Rouge			1 W	4	3,50 €	14,00 €
Lampe LED Jaune			1 W	4	4,30 €	17,20 €
Lampe LED Vert			1 W	2	6,60 €	13,20 €
Connecteurs	J2, J3, J4, J5	8 broches		4	1,36 €	5,44 €
	JP	5 broches		1	1,12 €	1,12 €
	JP1	CON ISP		1	2,40 €	2,40 €
	JP2	2 broches		1	2,30 €	2,30 €
	JP3	DB12 femelle		1	2,53 €	2,53 €
TOTAL				59		85,41 €

6. Planning final

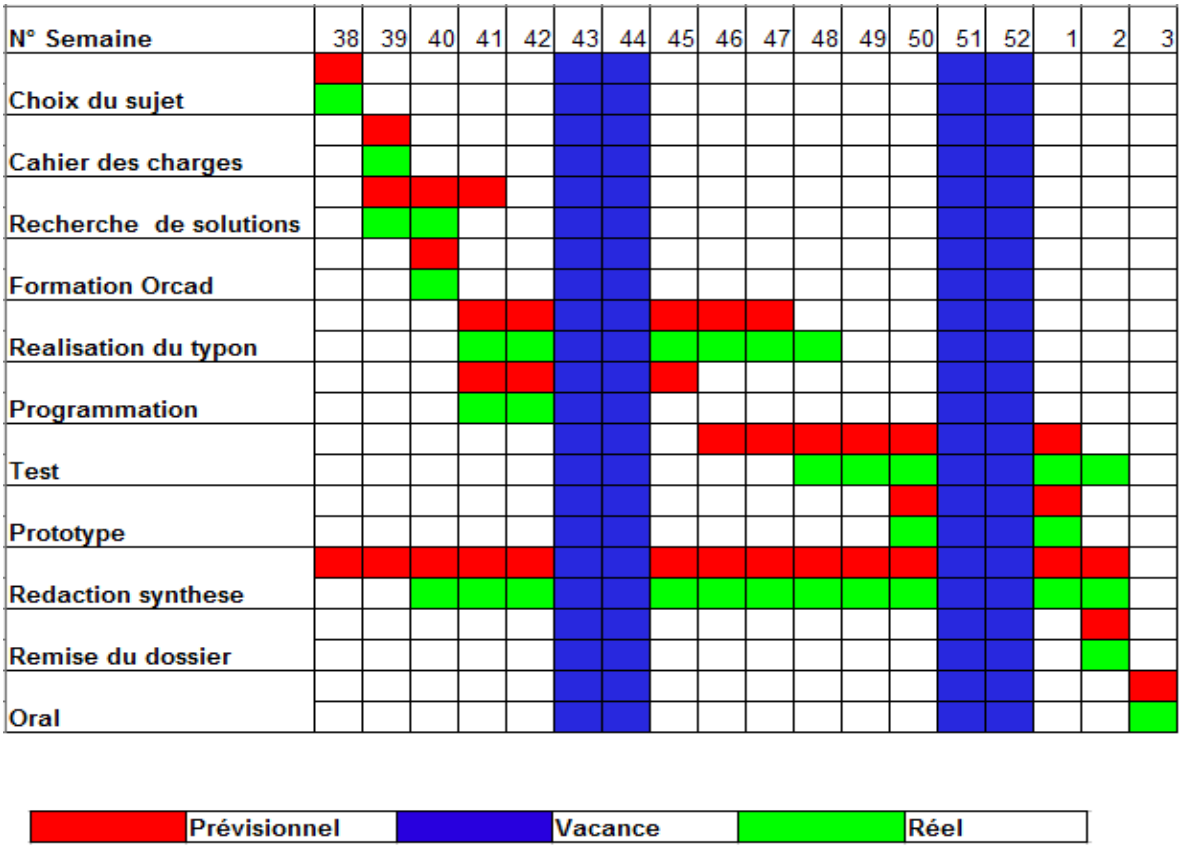


Illustration 21: Planning final

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 notamment le transistor MOSFET dans le cours EN1.

Nous remercions sincèrement notre professeur M Thierry LEQUEU, qui nous a aidé pendant la réalisation de notre carte électronique. Grâce à lui, nous avons découvert quelques nouveaux logiciels comme Code Vision AVR pour la partie programmation, Orcad Capture et Layout qui nous ont servi pour la réalisation du typon plus facilement.

A ce jour, nous n'avons pas eu le temps de réaliser le test de la programmation mais nous espérons pouvoir le faire avant le passage à l'oral. Ce test nous permettrait de voir si le programme correspond à nos attentes.

Pour conclure nous pouvons dire que ce projet nous a permis de mettre en oeuvre 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'a la suite de nos études.

Résumé

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. L'une des entrées est le photorésistance qui permet l'acquisition de tension selon la luminosité ambiante afin de faire varier l'intensité lumineuse des feux avants du kart. Puis, nous avons utilisé un potentiomètre sur la pédale de freinage qui nous a permis de commander l'intensité lumineuse des feux de stops en détectant la pression appliquée à la pédale. Ainsi plusieurs bouttons poussoirs à deux positions ou trois positions sont placés sur les entrées pour commander des clignotants et phares. Nos sorties constituées par les lampes à LED qui sont activées dépendent des états des transistors MOSFET qui fonctionnent comme des interrupteurs. De plus, l'alimentation à découpage de type BUCK est utilisée afin 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 décidé d'utilisé le logiciel Code Vision AVR qui est basé principalement sur le langage C.

Index des illustrations

Bibliographie

[1] http://www.thierry-lequeu.fr

[2] http://www.thierry-lequeu.fr/data/RAP-ROSLI.pdf

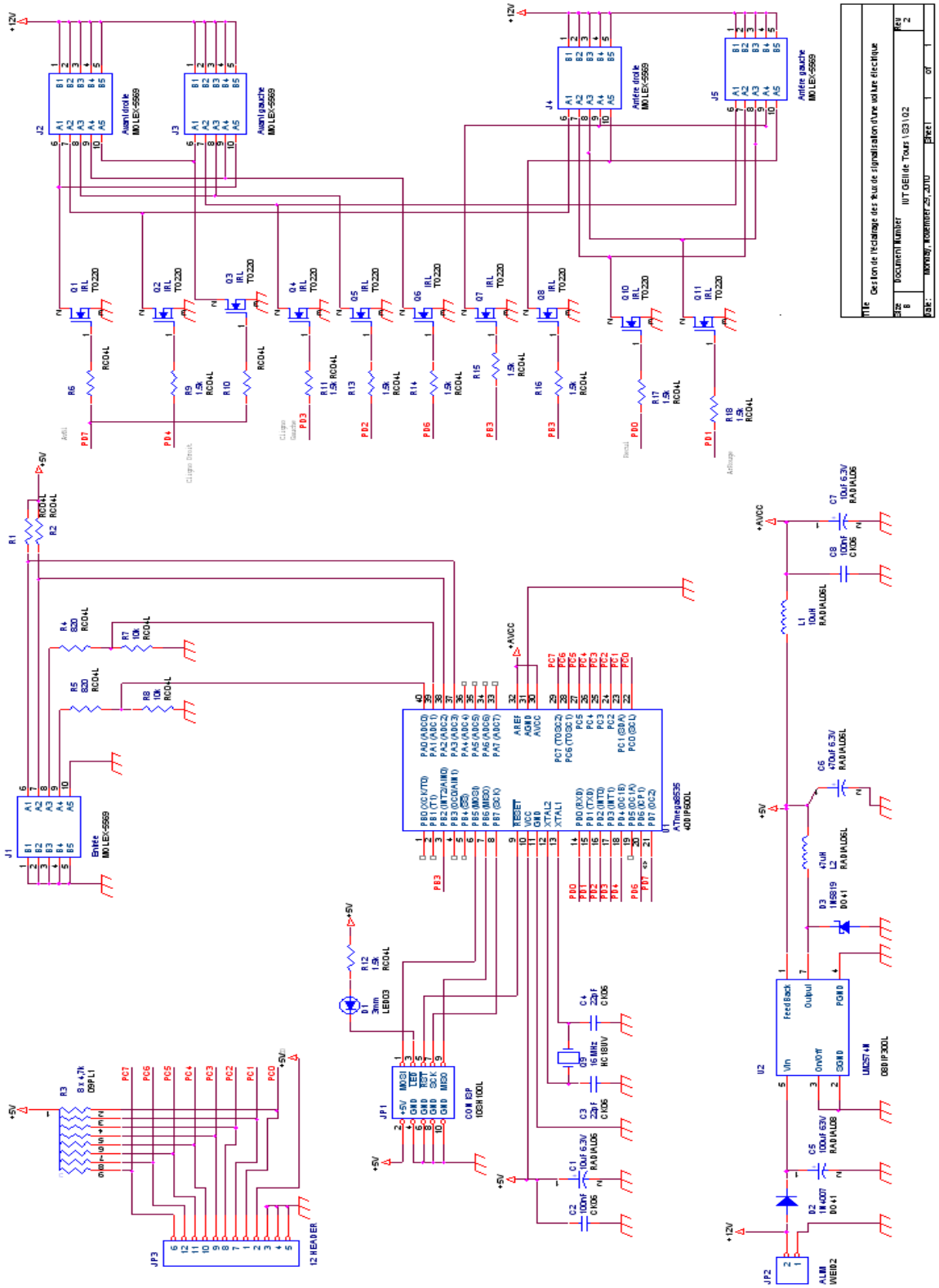
[3] http://www.thierry-lequeu.fr/data/RAP-DIEME.pdf

[4] Cours de MC-ET2

[5] Module complémentaire microprocesseur

Les annexes

Annexe 1: Schéma capture de la carte centrale du système



Annexe 2: Programmation complète

```
#include <mega8535.h>
#include <stdio.h>
#define ADC_VREF_TYPE 0x20
unsigned char read_adc(unsigned char adc_input)
{
ADCSRA|=0x40;
while((ADCSRA&0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
//variables globales

//Déclaration des Sorties
#include<delay.h>
#define feux_recul PORTD.0
#define feux_arriere PORTD.1
#define feux_avant PORTD.2
#define cligno_g PORTD.3
#define cligno_d PORTD.4
#define feux_stop PORTB.3
#define feux_route PORTD.7
#define croisement PORTD.6

//Déclaration des Entrées
#define BP_cligno_g PORTC.0
#define BP_cligno_d PORTC.1
#define BP_warning PORTC.2
#define BP_mode PORTC.3        //  AUTO/MANU
#define BP_feux_route PORTC.4
#define BP_feux_crois PORTC.5
#define BP_eclairage PORTC.6   //  ON/OFF
```

```
//Programme principale
void main(void)
{
float Nf,Ne;
//Initialisation du port A
PORTA=0x00;
DDRA=0x00;

//Initialisation du port B
PORTB=0x00;
DDRB=0x08;


//Initialisation du port C
PORTC=0x00;
DDRC=0x00;

//Initialisation du port D
PORTD=0x00;
DDRD=0xFF;

//Initialisation des compteurs

//Compteur 0
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

//Compteur 1
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
```

```
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BH=0x00;

//Compteur 2
ASSR=0x00;
TCCR2=0x64;
TCNT2=0x00;
OCR2=0x00;


MCUCR=0x00;
MCUCSR=0x00;

TIMSK=0x00;

ACSR=0x80;
SFIOR=0x00;

ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;


while (1)
{
    if(BP_eclairage==0)         //Mettre ON
    {
            if(BP_warning==0)
            {
                    cligno_d=!cligno_d;       //On fait clignoter le clignotant droit
                    cligno_g=!cligno_g;       //On fait clignoter le clignotant gauche
```

```
        delay_ms(500);
}
else
{
        if(BP_cligno_d==0)
        {
                cligno_d=!cligno_d;
                delay_ms(500);
        }
        else
                cligno_d=0;

        if(BP_cligno_g==0)
        {
                cligno_g=!cligno_g;
                delay_ms(500);
        }
        else
                cligno_g=0;
}
if(BP_mode==0)   // Mode manuel
{
        feux_arriere=1;
        feux_avant=1;
        feux_route=!BP_feux_route;
        croisement=!BP_feux_crois;
}
if(BP_mode==1)  // Mode automatique
{
        Nf=read_adc(3);
        OCR0=(48.5*Nf+12.75);

        Ne=read_adc(2);
        if(Ne>125)
```

```
                {
                        PORTD.7=0;
                        PORTD.6=0;
                        OCR2=173;
                        feux_arriere=1;
                        feux_avant=1;
                }
                if(Ne>173)
                {
                        PORTD.7=0;
                        PORTD.6=1;
                        OCR2=214;
                        feux_arriere=1;
                        feux_avant=1;
                }
                if(Ne>220)
                {
                        PORTD.7=1;
                        PORTD.6=1;
                        OCR2=255;
                        feux_arriere=1;
                        feux_avant=1;
                }
                else
                {
                        PORTD.7=0;
                        PORTD.6=0;
                        OCR2=0;
                        feux_arriere=0;
                }
        }
    }
}
}
```