Université François Rabelais de Tours

Institut Universitaire de Technologie de Tours

Département Génie Electrique et Informatique Industrielle

E-Kart'Lighting Project

Moyrand Julien
Beaugrand Stephane
2ième Année – P2
Promotion 2006/2008

Enseignant: Thierry LEQUEU

Université François Rabelais de Tours

Institut Universitaire de Technologie de Tours

Département Génie Electrique et Informatique Industrielle

E-Kart'Lighting Project

Moyrand Julien
Beaugrand Stephane
2ième Année – P2
Promotion 2006/2008

Enseignant: Thierry LEQUEU

Sommaire

PROJET Étude & Réalisation
E'Kart'Lighting Project

Descriptif :

L'étude proposée, consiste à des feux de signalisations à partir d'un microcontrôleur afin de pouvoir équiper un kart des feux de signalisations.

Cahier des charges :

- Alimentation continu 0 / 48v
- ATmega8535
- Commande Feux de stop avant et arrière
- Commande Feux de route
- Commande Feux de croisement avant et arrière
- Commande Clignotants
- Coût et encombrement réduit
- Capteur de position
- Logiciel Codevisionavr

Étude :

Le but de ce projet est d'étudier et de mettre en œuvre un montage conformément au cahier des charges. Le rapport doit comporter une explication sur la programmation du microcontrôleur, ainsi qu'une explication sur la technologie de l'ATméga8535 ainsi que des composants choisis.

Introduction

Dans le cadre des cours, il nous a été demandé de d'éffectuer un projet. Nous nous sommes porté vers la réalisation de la signalisation des karts.

Pour réaliser se projet nous avons utilisé le logiciel codevisionAVR mis à notre disposition à l'IUT lors des séances de projet.

Nous verrons les differents systèmes à programmer, par la suite nous étudierons la programmation sous codevisionAVR et enfin la partie réalisation du système.

1. Etude Blocs des différentes parties du système à programmer ou à piloter

1.1. Système de commande des phares avant

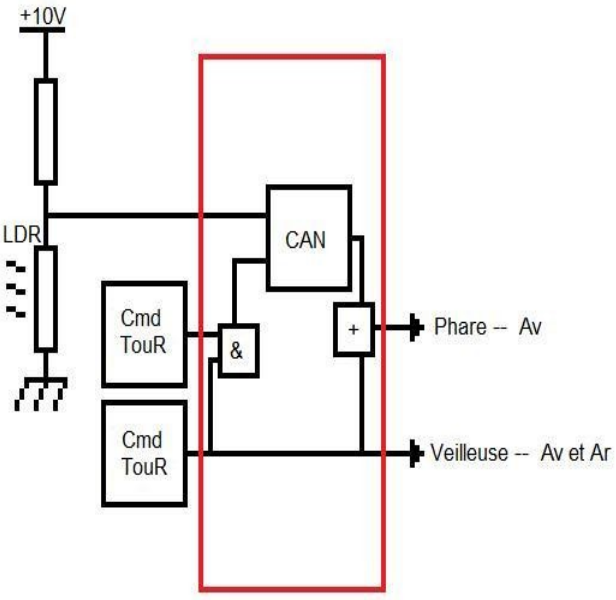

Figure 1: Schéma Bloc de gestion des Phares et Veilleuses

La commande des phares par une première commande Tout-ou-Rien pilotant les veilleuses Avant et Arrière (en bas du schéma).
On a une seconde commande Tout-ou-Rien pilotant les phares.

L'activation des phares se fait par l'activation préalable des veilleuses.
C'est donc la complémentarité des deux commandes qui activera la conversion du CAN.

Ce CAN permet de faire varier la tension de commande des phares en fonction de la luminosité ambiante (par la LDR (à gauche sur le schéma)).

Il fournit une tension de commande que l'on superposera (à l'aide d'un sommateur de tension) à celle des veilleuses afin de piloter l'intensité lumineuse des phares.

1.2. Système de commande des feux STOP

Les feux STOP seront activés par l'action sur le frein.
Le pilotage des feux STOP sera géré par le potentiomètre présenté ci-dessous. Il présente une commande. Tout-ou-Rien, permettant la conversion du CAN,et un cordon de sortie de potentiomètre qui fera transiter le signal analogique jusqu'au CAN.

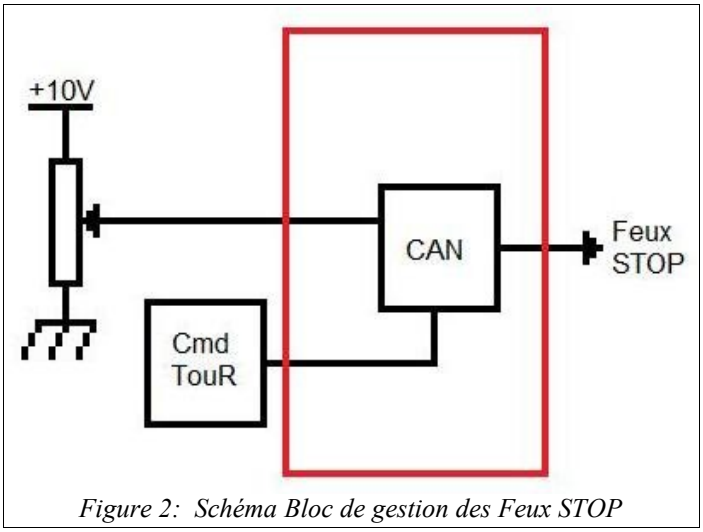

Figure 2: Schéma Bloc de gestion des Feux STOP

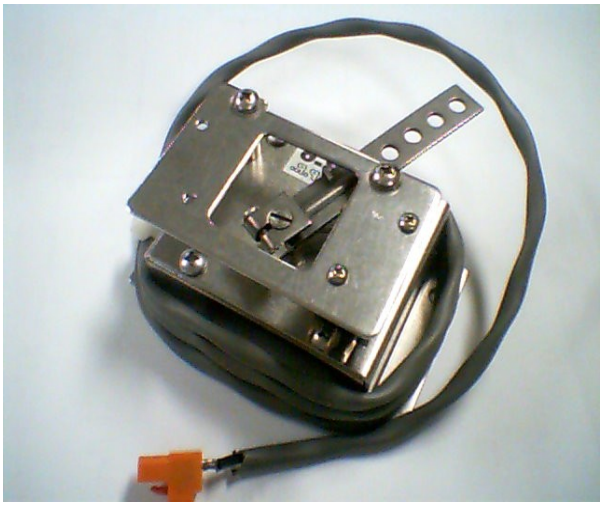

Figure 3: Photographie du potentiomètre de commande des feux STOP

1.3. Systéme de commande de feux de Recul

On a une simple commande Tout-ou-Rien actionnée par l'enclenchement de la marche arrière. Celle-ci sera reliée au levier d'actionnement de la marche arrière de l'engin.

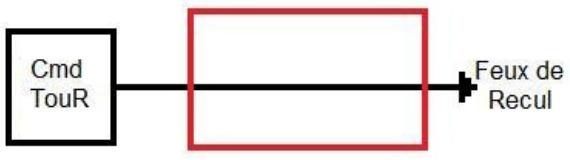

Figure 4: Schéma Bloc de gestion des Feux de Recul

1.4. Système de commande des clignotants

La fermeture d'un des deux commutateurs du comodo (Droite ou Gauche) générera l'activation d'un signal d'horloge.

Le positionnement sur la position repos du comodo désactivera les signaux d'horloge.

Ce signal d'horloge pilotera la conduction d'un transistor et donc le clignotement des optiques de sortie.

Il faudra simplement trouver un système de comodo type celui qui équipe les voitures actuelles.

Les blocs rouges des Schémas Bloc représentent les fonctions qui devront être développées lors de la programmation

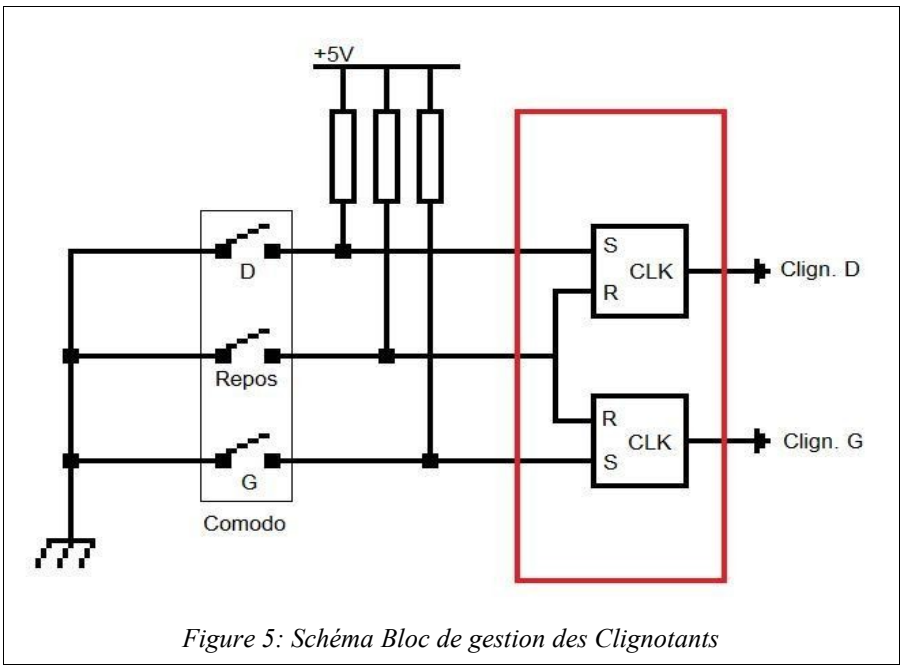

Figure 5: Schéma Bloc de gestion des Clignotants

2. Programmation en language C du ATmega8535

2.1. Présentation du module de programmation

Pour la programmation de notre projet nous avons utilisé la platine de programmation présentée ci-dessous recommandée par notre enseignant tuteur.
Cette platine nous permet de programmer le microcontrôleur et pouvoir le tester sur des modules additionnels.

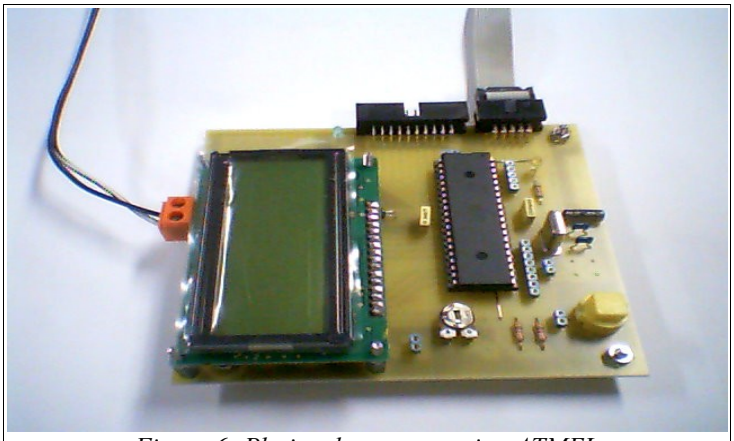

Figure 6: Platine de programation ATMEL

2.2. Présentation

Pour la programmation de notre projet, nous avons choisi l'ATmega8535, un microcontrôleur qui possède 3 timers et quatre ports dont un possédant des entrées analogiques.

L'ATMega8535 est doté de 2 Timers 8 bit et 1 16 bit.

Le Timer 0 (8 bit) et le Timer 1 (2x16 bit) ont en commun un même prédiviseur qui permet des divisions par 1, 8, 64, 128 et 1024, ceci afin d'ajuster au mieux la base de temps désirée. Ils disposent aussi chacun d'une entrée d'horloge indépendante. Le timer 2 quant à lui est à 8 bits et permet le branchement direct d'un quartz (32768 Hz). Il dispose aussi d'un prédiviseur personnel qui permet des divisions par 1, 8, 16, 32, 64, 128, 256 et 1024. Le timer est un outil pour pouvoir compter. Ces timers peuvent avoir des fonctionnalités différentes puisqu'ils peuvent, selon leurs entrées, compter du temps ou des événements. Ainsi ils servent de base de temps, de compteur ou de générateur MLI (Modulation de largeur d'impulsion = PWM).

Pour programmer, on utilise le logiciel CodeVisionAvr qui nous permet de configurer les timers avec précision, de régler en entrées ou en sorties les quatre ports, et de pouvoir utiliser des Convertisseurs Analogiques Numériques (CAN) ou Convertisseurs Numériques Analogiques (CNA) et de pouvoir compiler le programme en langage C.

Notre programme se décompose en 3 parties : tout d'abord la réalisation des feux stop, ensuite des feux de croisement et de route avant et arrière, et enfin la réalisation des clignotants.

2.3. Réalisation des Feux Stop

Tout d'abord, nous recevons en entrée une tension variable de 0V à 10V, et une commande tout ou rien envoyant un 0 logique ou un 1 logique.

On utilise un Convertisseur Analogique Numérique pour convertir la tension variable, pour cela cette tension est envoyée dans une entrée analogique qui s'appelle ici ADC1. Ceci permet de générer à partir d'une valeur analogique, une valeur numérique (codée sur plusieurs bits), proportionnelle à la valeur analogique entrée. La configuration du CAN :

```
// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
SFIOR&=0xEF;
```

Le convertisseur possède une horloge interne à une fréquence de 1 MHz, alimenté sous la tension du microcontrôleur.

Avec la fonction *read_adc(1)* permet de lire la tension et de la convertir automatiquement. Si la tension en entrée du CAN est de 0V alors en sortie du CAN nous aurons 0 et lorsque la tension est de 5V nous aurons 255 puisque la conversion est faite sur 8 bits.

```
//FEUX STOP
if(cmdstop==1)
{
        i=read_adc(1);                  //lecture de la valeur du potar
        sprintf(tampon,"%4d",i);
        lcd_gotoxy(0,1);
        lcd_puts(tampon);
        OCR0=i;
}else
        OCR0=0;
```

A la sortie du CAN, nous mettons une MLI pour faire varier le temps d'impulsion ainsi donc la valeur moyenne du signal. Lorsque l'entrée de la MLI est proche de 0, on obtient alors un signal carré avec un temps d'impulsion très étroit ainsi la valeur moyenne est faible. Et inversement si l'entrée de la MLI est proche de 255 alors le temps d'impulsion est large et donc la valeur moyenne du signal est grande.

Pour contrôler ce système nous utilisons dans le logiciel un timer 0 :

```
// 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;
```

Ce timer est configuré à une fréquence de 16 MHz, et en mode PWM c'est-à-dire MLI.
Dans le programme, après avoir converti la tension, la valeur obtenue est stockée dans une variable J. Celle-ci est alors implantée dans la MLI : `OCR0=i;`. Ainsi le temps d'impulsion de la MLI sera conforme la tension analogique.
Cette conversion n'est alors possible que si la commande du feu stop est mise à 1 : `if(cmdstop==1)`.

2.4. Réalisation des Feux de Croisement et de Route

La réalisation des feux de croisement et de route est le même principe que pour les feux de stop. Les feux de route ne peuvent être activés que lorsque les feux de croisement sont actionner. Dans le programme on convertit une tension analogique venant d'une photodiode qui est ensuite stockée dans une variable J : `j=read_adc(6);` que si la commande des feux de croisement est mise à 1. Cette tension est envoyée à l'entrée analogique ADC6.

On utilise le timer2 configuré de la même façon que le timer0 :

```
// 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;
```

Ce timer a une fréquence de 16MHz, et il est utilisé comme une MLI.

Programmation des Feux de croisement et de route :

```
// PHARE & CODE
if(cmdcode!=0)                  //cmd code
 {
    j=0;
    j=read_adc(6);              //lecture de la valeur du capteur de
lumière
    sprintf(tampon,"%4d",j);
    lcd_gotoxy(0,2);
    lcd_puts(tampon);

    if(cmdphare!=0)             //commande phare
                OCR2=j;
        else {
                j=10;
                OCR2 = j        //cst de code
                }
}
 else
        OCR2=0;
}
```

Lorsque la commande des feux de route est mise à 1, alors la MLI est activée et le temps d'impulsion est proportionnel à la tension convertit *OCR2 = j;*. Si cette commande est à 0 alors on affecte une constante à la MLI : *j=10;*

OCR2 = j;

2.5. Réalisation des Clignotants

Pour la commande des clignotants nous utilisons une interruption à partir du timer 1. Lors de l'activation du comodo, l'instruction courante est exécutée. L'interruption nous permet d'avoir un clignotement des leds de 0,5 seconde. C'est pour cela que nous avons choisi comme fréquence 62,5 Khz *Clock value: 62,500 kHz* sachant que le timer compte jusqu'à 65535. Pour avoir une demi seconde on calcul :

la periode : T = 1/F = 1/62500 = 16us

sachant que le timer compte jusqu'à 65535 ca nous fait une base de temps de :

16us x 65535 = 1,04 seconde

donc il faut diviser par 2 le compteur du timer soit :

35535 / 2 = 31250 soit 7A12 en hexadécimal

on a pour finir une base de temps de :

16us x 31250 = 0,5 seconde

Pour configurer le compteur du timer, il sufit d'écrire :

```
OCR1AH=0x7A;
OCR1AL=0x12;
```

Configuration du timer 1:

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 62,500 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0C;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x7A;
OCR1AL=0x12;
OCR1BH=0x00;
OCR1BL=0x00;
```

Programmation de l'interruption:

```
// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
        if(comodoD==1)
                clignoD = !clignoD;
        if(comodoG==1)
                clignoG = !clignoG;
        if(reset==1)
        {       clignoG = 0;
                clignoD = 0;
        }
}
```

La commande des clignotants se fait avec un comodo qui envoie en entrée de l'ATmega8535 0V ou 5V. Il y a trois positions: comodo Droit, comodo Gauche, et le reset. Quand l'une des trois positions est activée, on complemente la sortie `clignoD = !clignoD;` sauf le reset car on met les sorties à zéro lorsqu'il est activé `clignoD = 0;`.

3. Réalisation

3.1. Réalisation des blocs optiques

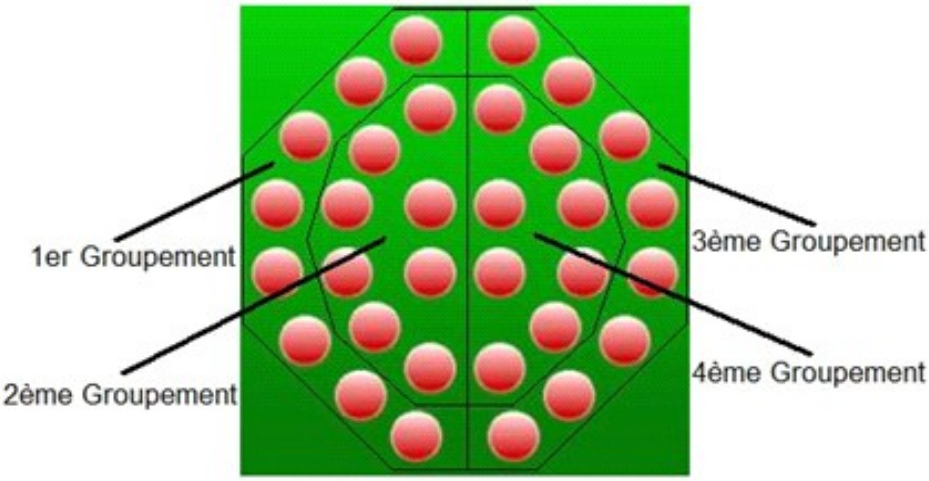

Figure 7: Schéma de répartition secteur de LED

Nous avons repris un modèle de bloc optique proposé par Thierry LEQUEU, notre enseignant tuteur et organisateur du projet. Ce modèle présente un bloc rond de 32 LED repartit en 4 groupements 8 LED.

Ces groupements sont pilotables 2 à 2 par commandes Cathodiques et sont câblées en Anodes communes. Ils possèdent donc 2 connecteurs de commandes et un connecteur de masse. Ce qui nous permettra de pouvoir obtenir des effets au sein des 4 groupements de LED, dont nous nous servirons pour notamment les feux stop et les phares avant...On observe également, sur le Typon (capturé ci-contre), 4 résistances de tirage répartie sur les 4 groupements de LED afin de réguler le courant traversant le système.
Ce système sera traversé par une tension de 48V (tension batterie des engins à équiper).

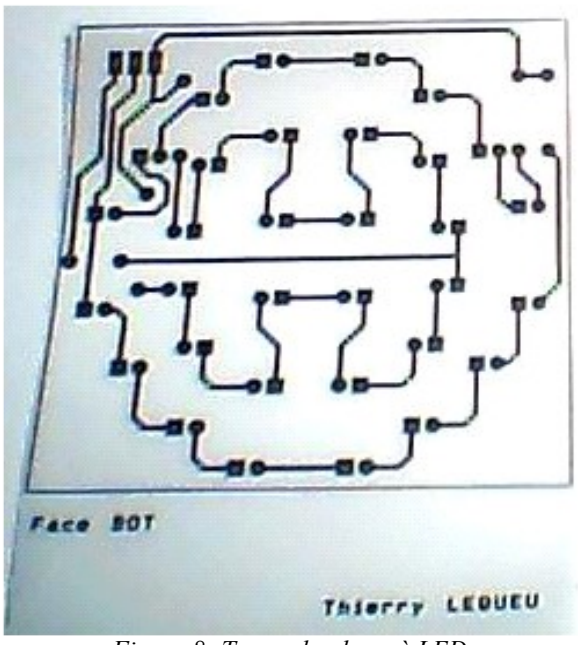

Figure 8: Typon du phare à LED

3.2. Réalisation de la carte de commande

3.2.1. Gestion de l'alimentation

Figure 9: Convertisseur DC / DC 48V - 15V

Pour pouvoir alimenter la carte supportant le microcontrôleur, il nous faut une tension d'alimentation de 5V, or la tension des batteries du kart nous impose une tension de 48V.

Pour abaisser la tension, nous utilisons un boitier spécifique TRACO POWER modèle TEN 5 (ci-contre), convertisseur DC / DC 48V - 15V.

Ce boitier nous fournit une tension de sortie de 15V, or nous avons besoin d'une alimentation 5V pour alimenter le microcontroleur. On réalise cette atténuation à l'aide d'un montage abaisseur de tension grace à un montage intégré LM2574N.

3.2.2. Schéma capture

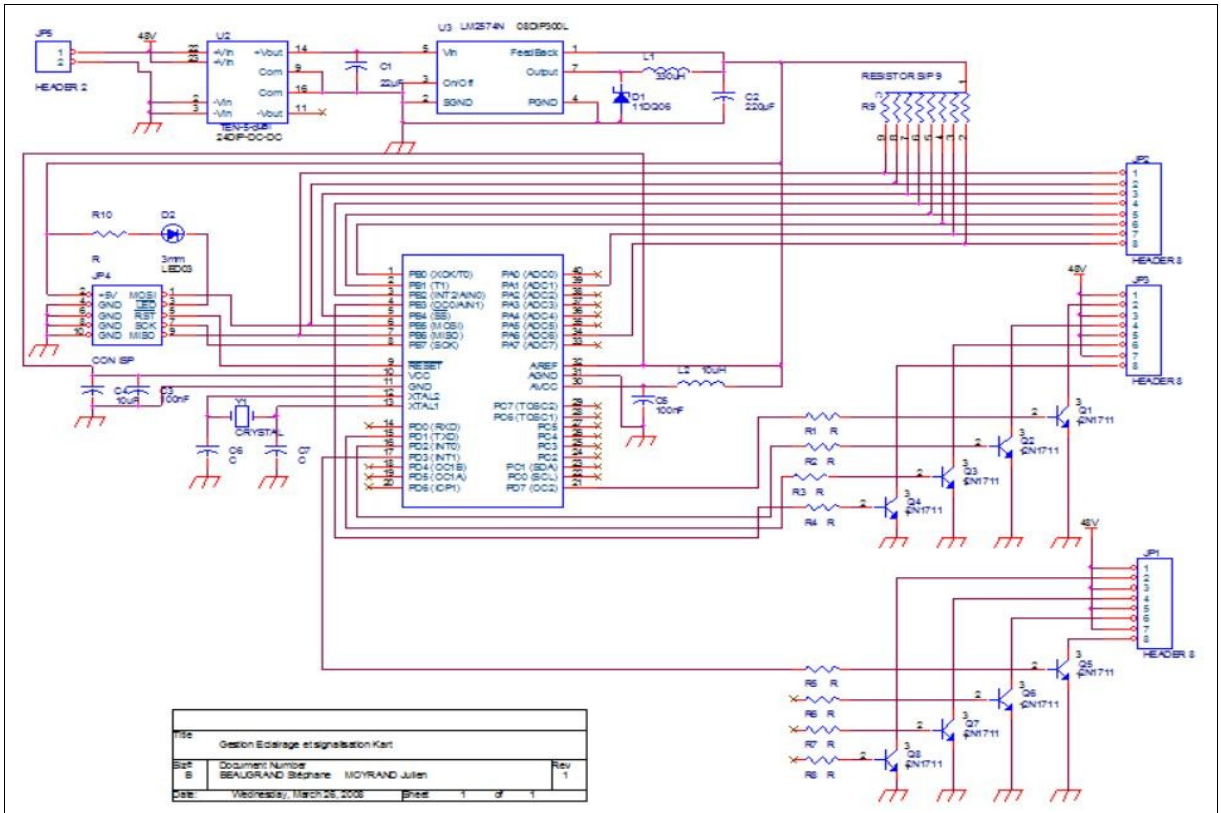

Figure 10: Saisie Capture de la carte centrale du système

3.2.3. Routage

Nous avons routé le typon à l'aide du logiciel Orcad, voici le routage capturé.

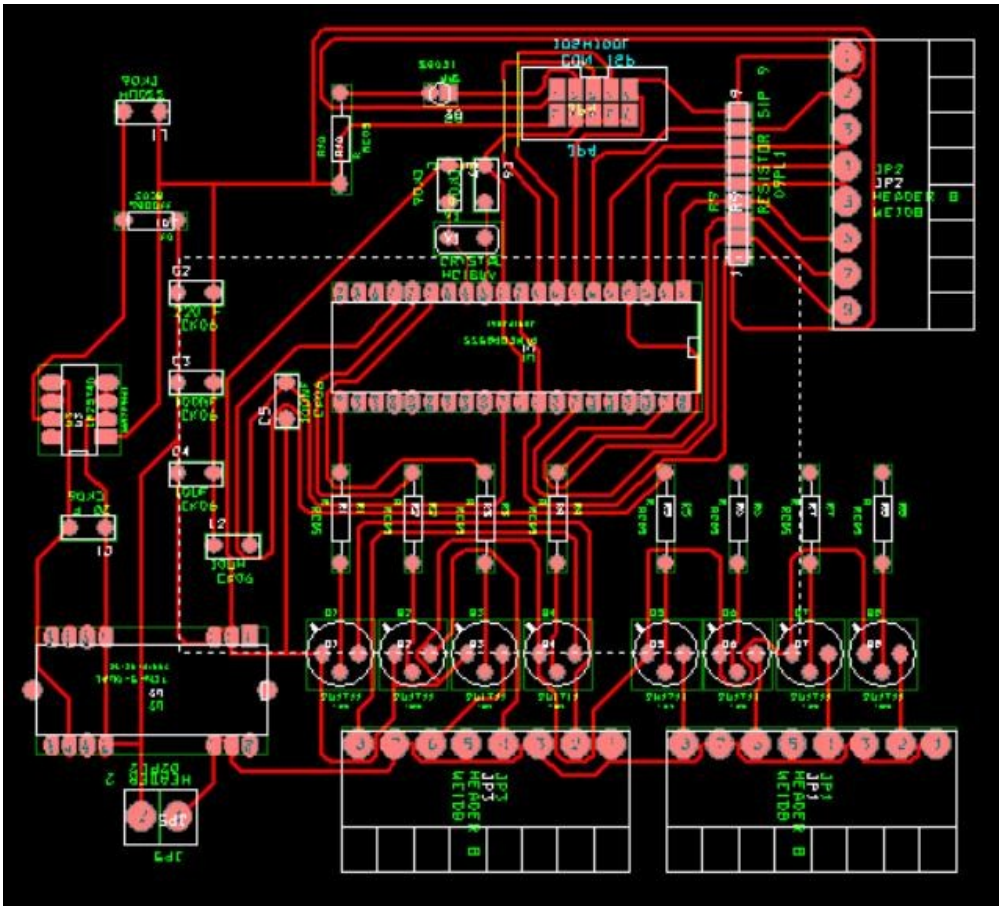

Figure 11: Saisie Layout de la carte centrale du système

Le routage ne laisse que deux pistes déconnectées au niveau du bornier de programmation. Nous devrons y placer deux straps afin de rétablir la continuité du circuit.

On remarque le placement logique des entrées sur le bas de la carte et le connecteur de sortie sur le bord droit de la carte.

Pour des contraintes de temps, nous n'avons pu réaliser la carte mère.

Conclusion

En conclusion, nous avons pu tirer profit de ce projet d'un point de vue innovation de procédé notamment avec la mise en oeuvre de la programmation d'un microcontrôleur et de son implantation dans un système électronique.
L'utilisation du logiciel Orcad nous a permis de se remettre en tête les commandes et fonctionalités de ce logiciel.
Le fait de mener un projet qui a une application sur un véhicule nous a bien motivé.

Le projet est bien avancé mais, manque de temps, il est décevant de ne pas pouvoir le finir. C'est un projet fort intéressant qui peut être optimisé par l'ajout de warning ou par des réflécteurs sur le montage à LED ici utilisé et un montage total en boitier afin d'avoir un produit fini.

En espérant que ce projet verra son application mise en place sur les véhicules de l'IUT et qu'il soit durable dans le temps.

ANNEXE

Table des illustrations

Bibliographie

Programmation:

http://www.thierry-lequeu.fr/data/DIV517.HTM

http://eva.unice.fr/portail_geii/files/Enseignants/Salvat/atmega/TD4_uc_AVR_assembleur.pdf

http://anyedit.free.fr/telechargement/ATMEGA%208535%20-%20Port.pdf

logiciel CodeVisionAvr : http://www.hpinfotech.ro/

Réalisation :

Schémas personnels de BEAUGRAND Stéphane et MOYRAND Julien

Logiciel ORCAD Capture et Layout

Datasheet :

Atmega8535 : http://www.thierry-lequeu.fr/data/AT-MEGA-8535L.pdf

Planning

Tâches \ Semaine	6	7	8	9	10	11	12	13	14
Recherche technologique et Cahier des charges									
Étude du projet									
Etude Codevision avr et programmation8									
Test du programme									
Réalisationsur orcade									
Test									
Redaction du projet									

LEGENDE

PREVISIONNEL

REEL

Schéma de capture