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

Feux de signalisation pour kart électrique

Paul Tallet-Pinet
Patrice Vieyra
Groupe : Q1
Année : 2010/2011

Enseignants :
Thierry LEQUEU
Bernard GLIKSOHN

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

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Feux de signalisation pour kart électrique

Paul Tallet-Pinet
Patrice Vieyra
Groupe : Q1
Année : 2010/2011

Enseignants :
Thierry LEQUEU
Bernard GLIKSOHN

Table des matières

INTRODUCTION

Au semestre 3 un groupe a réalisé une carte électronique permettant de contrôler les feux d'un kart électrique. Un programme avait été écrit. Le but de notre projet est de contrôler les feux du kart électrique. Pour ce faire nous allons étudier la carte électronique et le programme précédent afin de comprendre son fonctionnement . Nous avons remarqué de nombreux problèmes et oublis dans les projets précédents, la première partie de notre rapport sera consacrée à la description des réparations et améliorations d'ordre électronique et dans une seconde partie nous traiterons de la réalisation du programme de commande des feux du kart électrique.

1. cahier des charges

1.1. Contraintes[1]

- Utilisation de la carte électronique réalisée par un autre groupe au S3
- Utilisation du micro-contrôleur ATmega 8535
- Utilisation du boitier de commande
- Utilisation du boitier de test
- Programmation du micro-contrôleur
    - Feux de position
    - Feux de croisement
    - Éclairage variable (mode automatique)
    - Clignotants
    - Warnings
    - Feux de recul
    - Feux stop
- Programmation à l'aide du logiciel Code vision AVR

1.2. Cahier des charges

Contrôle des feux de signalisation du kart électrique

Il faut contrôler :

- les feux de STOP progressif en fonction de la position de la pédale de frein,
- les feux avant progressif en fonction de la lumière ambiante,
- les clignotants et les feux de détresses ;

Il faut utiliser la carte électronique disposant d'un ATmega 8535.

Étudier le programme existant puis l'améliorer, le compléter et le modifier pour contrôler les feux.

Planning prévisionnel

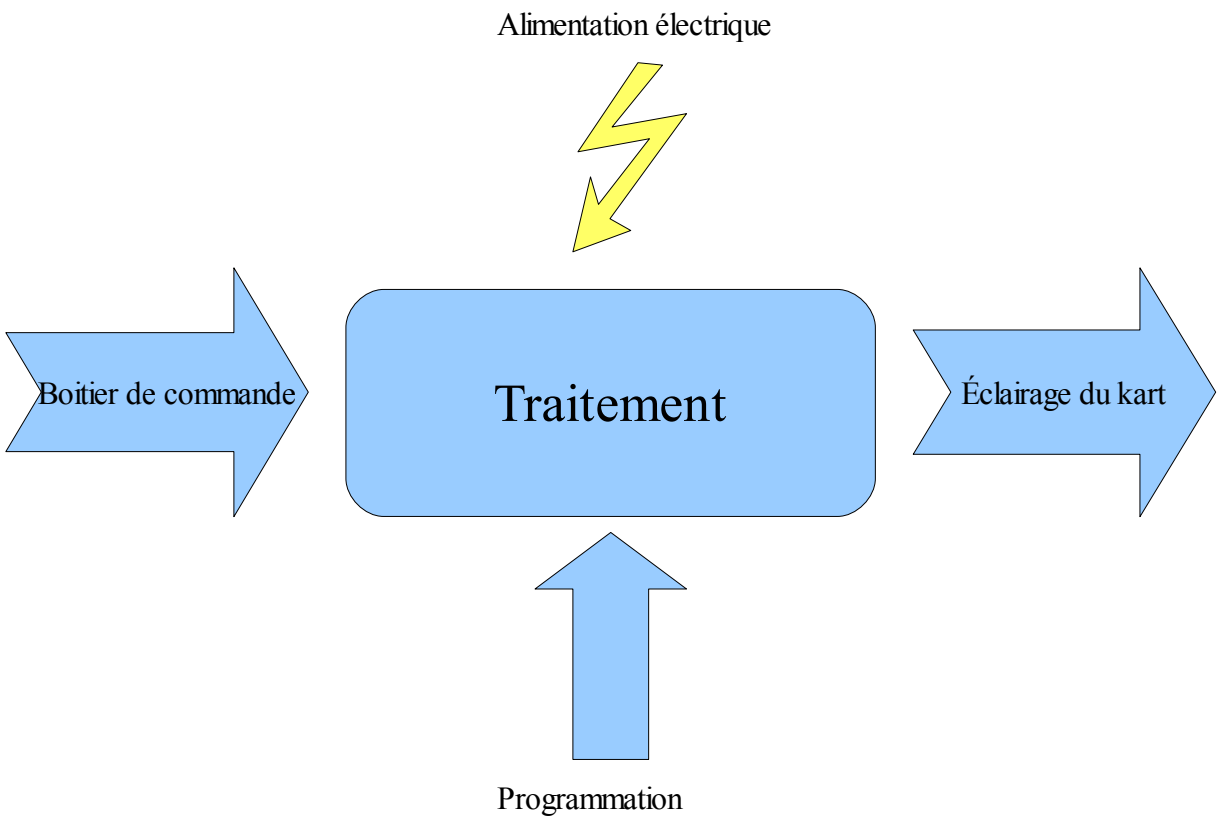
Semaine	05	06	07	08	09	10	11	12	13	14
Réalisation du cahier des charges										
Étude de la carte et du programme existant										
Conception du programme										
Test										
Réalisation du dossier										
Préparation et soutenance orale										

Nous avons rencontrer de nombreux problèmes de conception qui seront détaillés plus loin dans le rapport, ce qui nous a conduits à modifier complétement le planning.

Planning Réel

Semaine	05	06	07	08	09	10	11	12	13	14
Réalisation du cahier des charge										
Étude de la carte et du programme existant										
Réparation de la carte										
Test sur le kart										
Réalisation du dossier										

1.3. Synoptique général



2. Présentation du projet précédent

2.1. Présentation

Le projet précédent consistait en la réalisation d'une carte électronique contrôlant l'éclairage et la signalisation d'un kart électrique de l'IUT GEII de Tours.

Pour l'éclairage, la carte gère deux modes :

- Mode manuel : Gestion des feux de position et des feux de croisement
- Mode automatique : Éclairage variable en fonction de la luminosité et de la position de la pédale.

Pour la signalisation, la carte gère les clignotants, les feux de stop et les warnings.

2.2. Carte électronique[2]

Dans cette sous-partie, nous allons expliquer de manière succincte le

fonctionnement de la carte électronique.

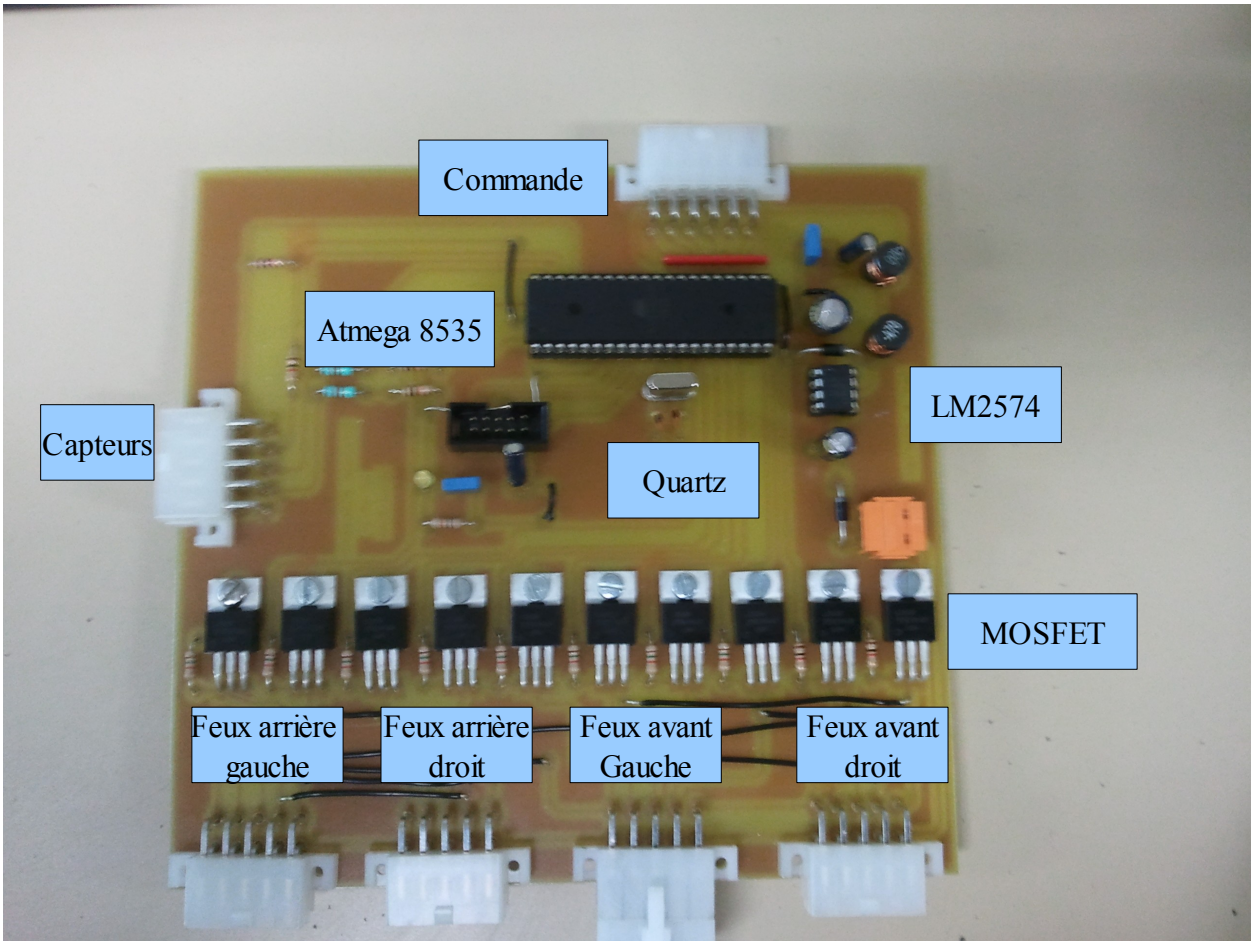


Illustration 1: carte électronique: vue de haut

- LM2574 : Le micro-contrôleur AtMega8535 doit être alimenté en +5V, il est donc nécessaire d'abaisser la tension de 12V à 5V. Pour ce faire un montage abaisseur de tension de type buck a été réalisé grâce à l'utilisation d'un régulateur de tension à découpage, le LM2574N. Ce type de montage réduit les pertes de puissances qu'il y aurait eu si un composant linéaire tel qu'un régulateur linéaire avait été utilisé.
- L'ATmega8535 : C'est un micro-contrôleur 8 bits qui se présente sous la forme d'un circuit intégré de 40 broches. Il possède de nombreuses entrées/sorties ainsi que plusieurs fonctions utiles pour la suite du projet (explication plus bas). C'est à l'intérieur de celui-ci que nous implanterons le programme.
- Quartz : C'est un composant qui possède la propriété d'osciller à une fréquence stable lorsqu'il est électriquement stimulé. Ce sont les propriétés piézoélectriques du minéral de quartz qui lui permettent d'obtenir une fréquence d'oscillation très précise. Le quartz utilisé est cadencé à 16 MHz, il permet de remplacer l'horloge interne de l'ATmega qui est cadencée à une fréquence moins élevée.
- Transistors MOSFET : Ils servent d'interrupteur de sortie, en effet ils possèdent 3 pattes (Grille(1), Source(3), Drain(2)). L a grille est reliée directement à l'ATmega et recevra des impulsions, le drain est relié à une des pattes d'une lampe (l'autre patte étant reliée au +12V) et la source rejoint la masse. Lorsque la grille reçoit une impulsion, le transistor se ferme et le drain rejoint la masse, ce qui permet à l'ampoule d'avoir une tension de 12V à ses bornes (lampe : une patte au +12V et une patte à la masse). Dans le cas contraire la lampe est éteinte puisque une de ses patte est en l'air (lampe : une patte au +12V et une patte en l'air).

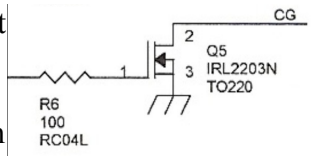


Illustration 2: Mosfet

- Connecteurs de sorties : Ces connecteurs seront directement branchés aux

LED et permettront la liaison entre les LED et la carte.

- Connecteur de commande : Connecteur branché au boitier de commandes et aux capteurs, il permet la liaison entre les interrupteurs, les capteurs et la carte.

2.3. Le micro-contrôleur ATmega 8535

L'ATmega 8535 est un micro-contrôleur 8bit fabriqué par la société Atmel, qui se présente sous la forme d'un circuit intégré 40 pattes.

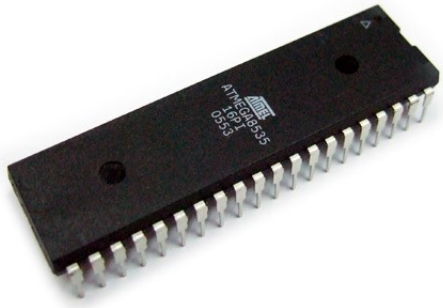


Illustration 3: ATmega 8535

2.3.1. Description des broches

L'ATmega 8535 possède 4 ports A, B, C, D ayant chacun des fonctions particulières et contenant 8 bits directionnels numérotés de 0 à 7. Pour chacun de ces ports les pattes peuvent être configurées en entrée ou sortie. En plus de ces ports l'ATmega possède plusieurs autres pattes que nous allons lister ci-après.[3]

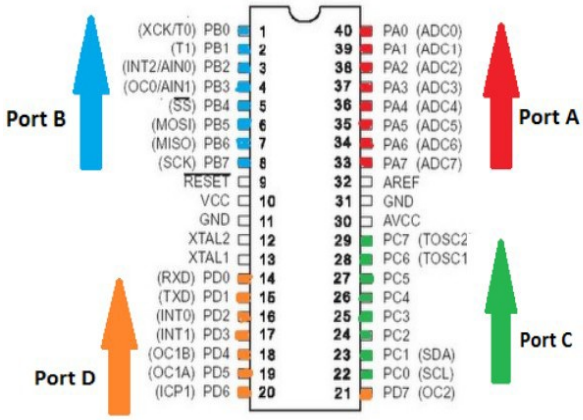


Illustration 4: brochage ATmega 8535

- RESET: cette patte permet de réinitialiser le micro-contrôleur, il faut appliquer une tension de VCC pour inhiber le reset;
- AVCC: C'est une patte d'alimentation pour le CAN1, elle doit être reliée au VCC via un filtre passe-bas qui élimine les parasites;
- AREF: C'est l'entrée de référence analogique du CAN;
- AGND: C'est une masse analogique;
- MISO, MOSI, SCK: Pattes de programmation du micro-contrôleur;
- INT0, INT1:Interruptions externes;
- VCC: C'est la broche d'alimentation du micro-contrôleur, le nôtre est alimenté en 5V;
- GND: C'est la masse de l'alimentation.

2.3.2. Description de deux fonctions importantes

2.3.2.1. Le CAN[1]

Le CAN ou convertisseur analogique numérique, permet d'obtenir l'image d'une tension analogique sur un certain nombre de bit, dans notre cas 10. Les deux sont reliés par l'équation suivante: Alphanumérique $= (Tensiomètre/Référencement)*1024$

Nous allons utiliser le CAN pour l'acquisition de la tension retournée par les deux capteurs.

En effet la photo-résistance nous renvoie une tension proportionnelle à l'éclairage ambiant, et le potentiomètre mécanique nous renvoie une tension proportionnelle à la position de la pédale.

2.3.2.2. La MLI[2]

La MLI ou modulation en largeur d'impulsion est un moyen d'obtenir une tension moyenne variable à partir d'une alimentation continue. Elle repose sur la modulation du rapport cyclique α

Le rapport cyclique est calculé par la formule suivante:

$$\alpha = T/Période$$

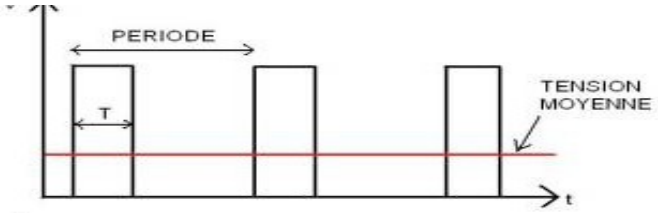


Illustration 5: MLI

Ainsi à 100% la tension moyenne est maximale tandis qu'à 0% la tension moyenne est nulle. Nous allons utiliser la MLI pour moduler l'éclairage des feux de stop en fonction de la position de la pédale de frein.

1 Convertisseur analogique numérique
2 Modulation de largeur d'impulsion

2.4. Étude et mise en place des capteurs

Un capteur est un dispositif qui transforme une grandeur physique en une grandeur étudiable. Dans notre projet, nous allons mettre en œuvre deux capteurs passif : une thermistance et un potentiomètre mécanique. Ces deux capteurs sont modélisables par une impédance : une variation du phénomène physique à savoir la luminosité pour la photo-résistance et la position de la pédale pour le potentiomètre, entraîne une variation de l'impédance du capteur.

2.4.1. La photo-résistance

C'est un composant électronique ayant la propriété d'avoir une résistance qui varie en fonction de la luminosité. Nous allons l'utiliser pour faire varier la luminosité des feux en fonction de la lumière ambiante.

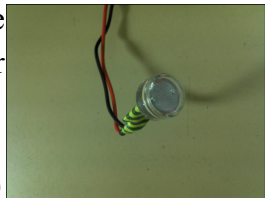


Illustration 6: photorésistance

2.4.2. Le potentiomètre mécanique

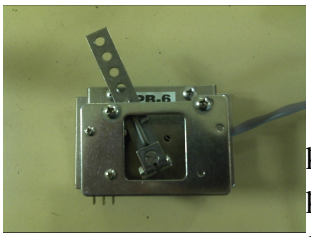


Illustration 7: potentiomètre mécanique

C'est un composant mécanique ayant la propriété d'avoir une résistance qui varie en fonction de la position de la pédale. Nous allons l'utiliser pour faire varier la luminosité des feux de stop en fonction de la position de la pédale de frein.

3. Partie électronique

Dans cette partie nous allons expliquer les différentes réparation et modifications qu'il a fallut apporter à la carte électronique et au boîtier de commande.

3.1. Le boîtier de commande

La maquette est commandée par un boîtier de commande conçu en 2009 par un groupe d'étudiant. Le boîtier contient des interrupteurs correspondants aux différents feux de signalisation du kart électrique.

A l'intérieur le câblage a été réalisé avec des fils rigides soudés sur un connecteur DB15 afin de communiquer avec la carte électronique. Cependant plusieurs étant dessoudés, il a fallut ouvrir le boîtier pour le réparer. Mais nous avons constater qu'aucune table de routage n'avait été mis en place. Nous avons testé à l'ohmmètre les contacts du DB15 ainsi que les pins de la carte pour trouver la place de chaque fil. Nous avons décidé de changer le connecteur DB15 afin de repartir sur de bonnes bases.

Illustration 8: boîtier de commande

1	Marron	9	Rouge
2	Jaune	10	Vert
3	Blanc	11	Vert
4	Violet	12	Gris
5	Noir	13	Gris
6	Orange/blanc	14	Rose
7	Orange	15	Rose
8	Bleu		

Illustration 10: table de routage du DB15

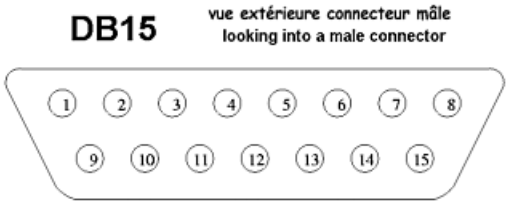


Illustration 9: brochage DB15

3.2. La maquette de test

Pendant la phase de développement nous avons utilisé une maquette pour simuler les feux de signalisations du kart électrique par soucis de commodité. Là aussi nous avons rencontré un problème, en effet les fils des lampes étaient trop court pour être tous raccordés à la carte de commande. Nous avons donc dû rallonger les fils afin d'être en mesure de tester le programme sur une maquette complète. Cette maquette nous a permis de tester le programme de commande en nous affranchissant des éventuels problèmes liés au kart (câblage, alimentation, encombrement,...)

3.3. La carte électronique

Notre première mission a été de couper une piste et de faire un strap entre deux pins de l'ATmega. En effet, suite à une erreur de routage, il y avait eu une confusion entre ces deux pins. Lorsque nous avons commencer à nous servir du CAN[3], nous nous sommes rendu compte que la tension de référence était très parasitée. Nous avons donc soudé des condensateurs de découplages sur les entrées des CAN ainsi que sur son alimentation afin de supprimer le bruit.

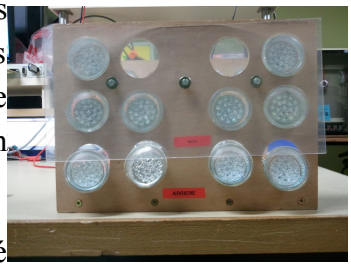


Illustration 11: maquette

Lors de la réalisation de la carte électronique, il a été décidé d'utiliser des connecteurs de type molex 2*5 pour les

3 Convertisseur Analogique Numérique

sorties des feux. Aucune table de routage n'existait pour connaître l'assignation de chaque pin, nous l'avons donc réalisée.

Feux avant droit	Feux avant gauche	Feux arrière droit	Feux arrière gauche
. . . . . F Cd Av C F	. . . . . . F Cg Av C F	. . . . . . Cd R Fa Fs Fs	. . . . . . Cd R Fa Fs Fs
F=Feux de route, Cd/Cg=Clignotants gauche/droit, Av=Feux avant, C=Feux de croisement, R=Feux de recul, Fa=Feux arrière, Fs=Feux de stop			

Illustration 12: table de routage des feux

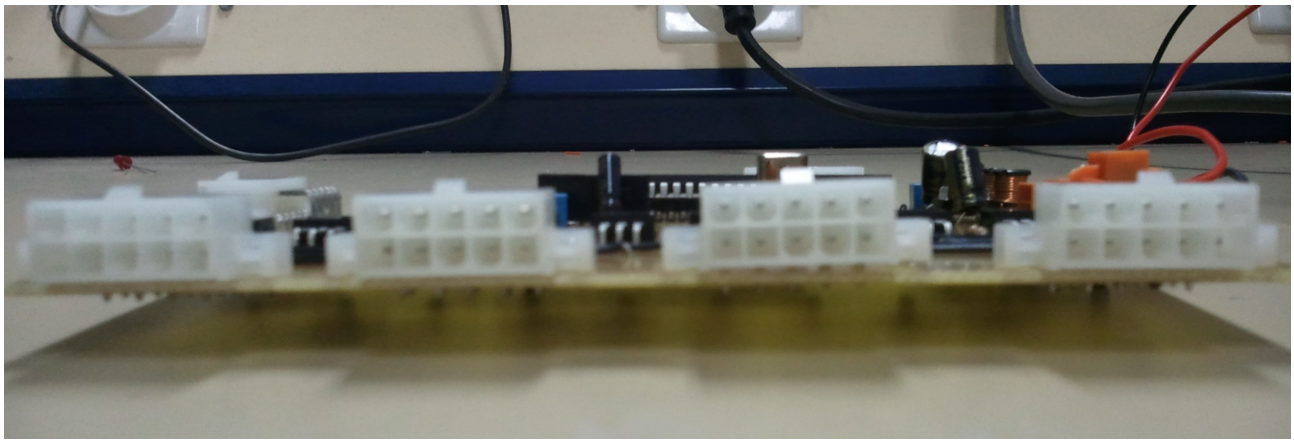


Illustration 13: carte électronique: vue de face

Cette partie du projet n'avait pas été prévue dans le planning. Nous avons perdu du temps à réparer la carte et le boîtier électronique. Cependant cette a été bénéfique puisqu'elle nous a permis de mieux comprendre le fonctionnement de la carte électronique du groupe précédent.

4. Partie informatique

4.1. Organigramme principal du programme

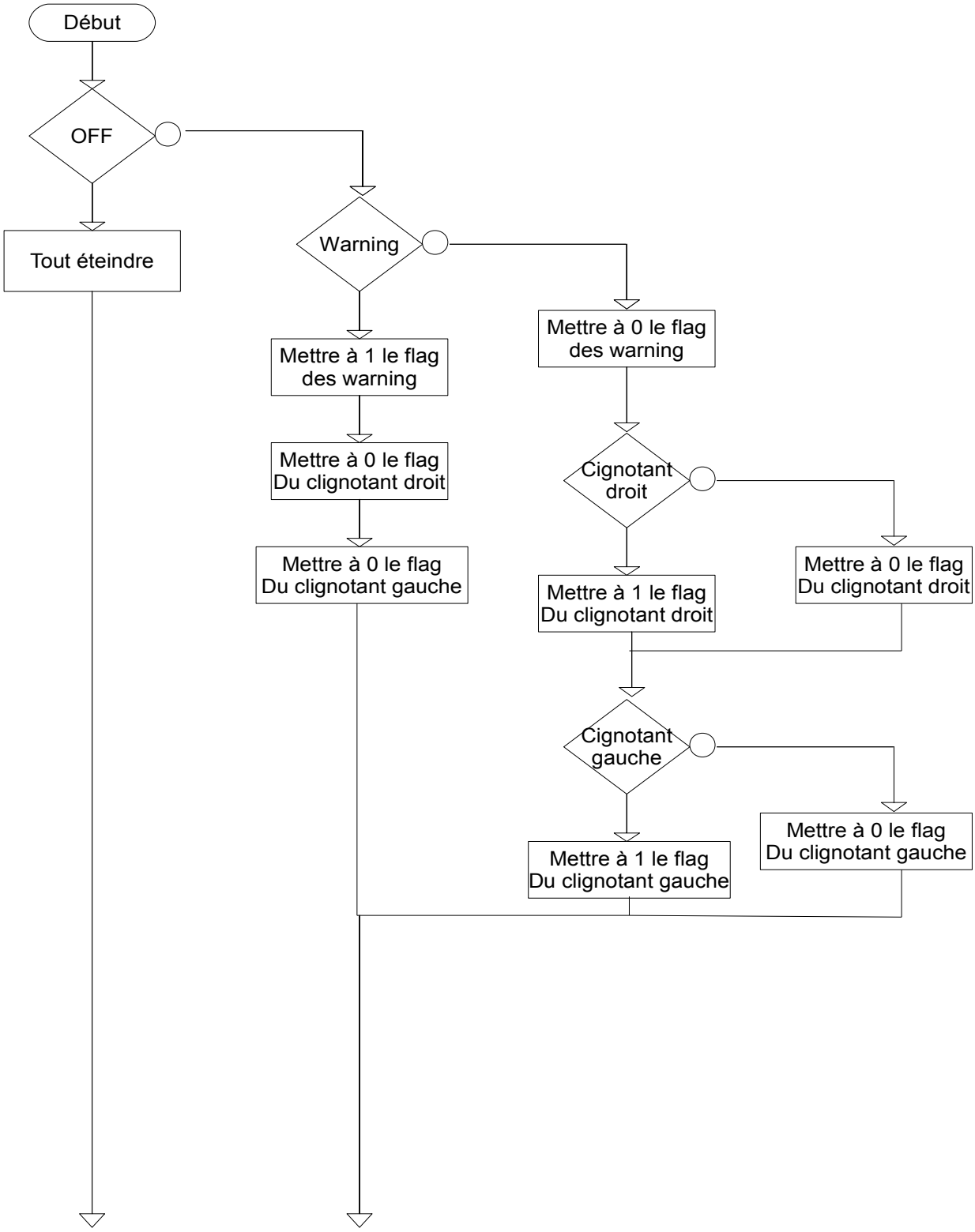


Illustration 14: ordinogramme principal début

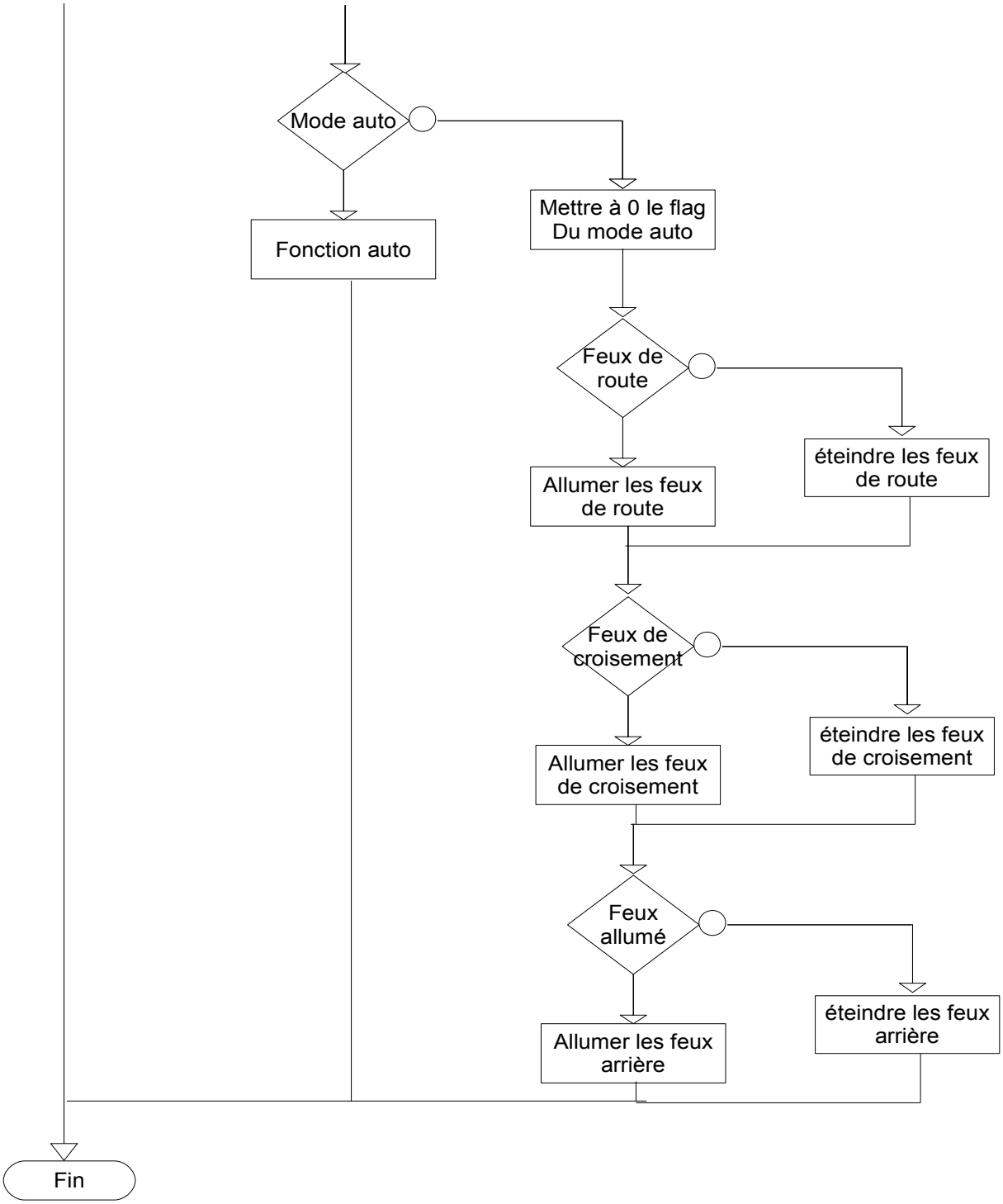


Illustration 15: ordinogramme principal fin

Le programme principal est implanté dans une boucle infinie, les instructions contenues dans cette boucle s'exécutent en permanence de façon cyclique. Les flags des warnings et des clignotants sont utilisés dans une fonction d'interruption interne qui permet de les faire clignoter.

4.2. Gestion des clignotants et des warnings

Nous avons choisi d'utiliser une fonction d'interruption pour gérer les clignotants et les warnings de façon à les rendre indépendants des autres feux de signalisation du kart. La gestion des clignotants et des warnings se fait à l intérieur d'une fonction d'interruption qui se lance toute les 0.5

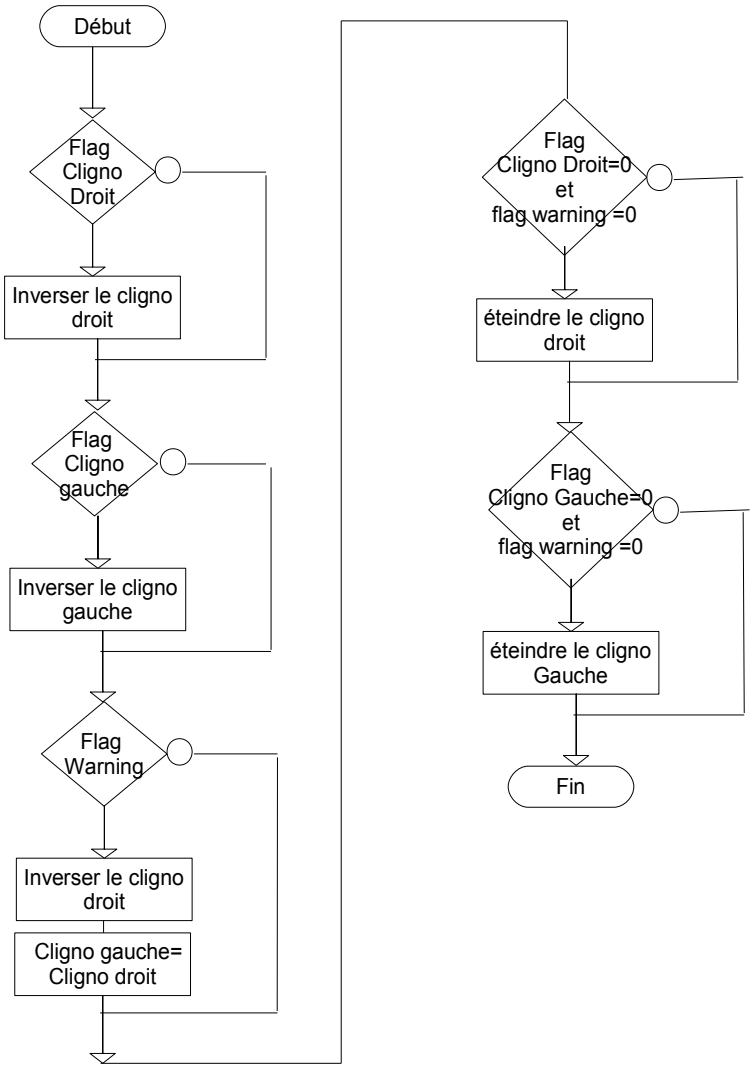


Illustration 16: ordinogramme de la fonction d'interruption

seconde.

Pour configurer cette interruption toute les 0.5 seconde nous avons utiliser l'outil CodeWizardAVR

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 3,906 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x0f;
OCR1AL=0x3c;
OCR1BH=0x00;
OCR1BL=0x00;
```

4.3. Gestion du mode AUTO

Le mode AUTO a deux objectifs : adapter la puissance des feux avant en fonction de la luminosité et créer des feux de stop à éclairage progressif en fonction de l'appui sur la pédale de frein. De plus en mode automatique, tout les interrupteurs gérant les feux hormis les clignotants et warnings gérés par la fonction d'interruption, sont désactivés Nous avons donc créé une fonction qui recevra un numéro de voie en paramètre et qui devra lire la tension analogique de cette voie et la convertir en valeur numérique:

```
// ADC interrupt service routine
// 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;
      // Start the AD conversion
      ADCSRA|=0x40;
      // Wait for the AD conversion to complete
      while ((ADCSRA & 0x10)==0);
      ADCSRA|=0x10;
      return ADCH;
}
```

Pour configurer le CAN nous avons utiliser l'outil CodeWizardAVR

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
```

Pour éviter les parasites qui peuvent provenir de la pédale de frein nous avons réaliser un filtrage numérique. Pour faire ce filtrage nous avons fait la moyenne de 100 relevés de la valeur numérique du CNA, et c'est cette moyenne que nous utilisons par la suite.

```
Moyenne=Moyenne+read_adc(3);
N++;
if(N>=100)
{
       OCR0=(Moyenne/100)/2.15;
       Moyenne=0;
       N=0;
}
```

La modification du registre OCR0 permet de modifier la valeur du rapport cyclique du timer0 et ainsi faire varier la valeur moyenne de la tension de sortie. Nous avons du corriger la valeur en divisant par 2.15 pour adapter la caractéristique des diodes. En effet la variation de luminosité n'était visible que sur une petite plage de variation de la pédale de frein. En divisant les valeurs par 2.15 nous avons dilater cette zone de variation de façon à avoir une variation plus douce sur l'ensemble de la plage de mouvement de la pédale de frein.

Nous avons utiliser la même technique pour filtrer la valeur de la luminosité.

```
luminosite=luminosite+read_adc(2);
J++;
if(J>=100)
{
       luminosite=(luminosite/100);
       OCR2=(((luminosite-70)/2) / (10*exp(-(luminosite-70)/4)+1));
       luminosite=0;
       J=0;
}
```

Là aussi nous avons dû adapter la fonction. En effet nous avions deux problèmes principaux. En plein jour la photo-résistance ne recevait pas assez de lumière donc les feux étaient faiblement allumés. De plus, dans l'obscurité, la pièce était faiblement éclairée donc nous n'avions pas un éclairage maximal. Notre correction permet de corriger ces deux problèmes, de plus elle permet de dilater la courbe de façon à avoir une variation linéaire de l'éclairage des feux en fonction de la luminosité comme demandé dans le cahier des charges.

CONCLUSION

Le but de notre projet était de réaliser la programmation de l'ATmega chargé de gérer les feux de signalisations du kart électrique. Cet objectif a été atteint. Ce projet a été réellement enrichissant car nous avons rencontré des problèmes imprévus du coup nous n'avons pas pu respecter le planning prévisionnel, nous avons été obligé de régler ces problèmes le plus vite possible afin de finir le projet à temps. Nous avons donc appris en condition réelle, à gérer notre temps, à se répartir les tâches.

RESUME

Pour ce projet tutoré de S4, nous avons choisi de réaliser ce projet dont le cahier des charges nous a été donné par M. Lequeu Thierry. Notre projet s'appuie sur des projets déjà réalisés lors des années précédentes. Durant toutes les séances, nous avons donc pu nous appuyer sur les dossiers des groupes précédents, et nous avons utilisé la carte électronique réaliser par un autre groupe lors du S3.

La première partie de notre projet a été d'étudier la carte électronique, de la tester , et de retrouver les erreurs faites lors de la réalisation. Ensuite nous avons réalisé le programme que nous avons pu tester sur la maquette de test qui était à notre disposition. Une fois le cahier des charges remplis, nous avons rajouter à notre programme un mode tuning avec deux choix de motif.

Index des illustrations

Bibliographie

1: Thierry Lequeu, La documentation de Thierry LEQUEU sur OVH, 2011, http://www.thierry-lequeu.fr
2: K. TOUBLANC, F. THYPHONNET, Eclairage 2008, 2008, http://www.thierry-lequeu.fr/data/DATA393.HTM
3: F. LAMBERT, M.N. BIN ROSLI, Eclairage 2010, 2010, http://www.thierry-lequeu.fr/data/RAP-LAMBERT-BIN-ROSLI.pdf

ANNEXE

Programme principal:

```
#include <mega8535.h>
#include <stdio.h>
#include<delay.h>
#include <math.h>
//variables globales
//Déclaration des Sorties

#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_d PINC.3 //INVERSE
#define BP_cligno_g PINC.2 //INVERSE
#define BP_warning PINC.1 //INVERSE
#define BP_mode PINC.0 // AUTO/MANU
#define BP_feux_route PINC.4 //INVERSE
#define BP_feux_crois PINC.5 //INVERSE
#define BP_eclairage PINC.6 // ON/OFF

#define ADC_VREF_TYPE 0x20
```

```
unsigned char Cligno_D;
unsigned char Cligno_G;
unsigned char Warning;
float Moyenne=0;
float luminosite=0;
int N=0, J=0;
int FlagTunnig1;
int FlagTunnig2;

// ADC interrupt service routine


// 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;
        // Start the AD conversion
        ADCSRA|=0x40;
        // Wait for the AD conversion to complete
        while ((ADCSRA & 0x10)==0);
        ADCSRA|=0x10;
        return ADCH;
}

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
```

```
if(Cligno_D)
{
        cligno_d=!cligno_d;
}
if(Cligno_G)
{
        cligno_g=!cligno_g;
}

if(Warning)
{
        cligno_d=!cligno_d;
        cligno_g=cligno_d;
}
if((Warning==0)&&(Cligno_D==0))
{
        cligno_d=0;
}
if((Warning==0)&&(Cligno_G==0))
{
        cligno_g=0;
}
if(FlagTunnig1==1)
{

        if(OCR2  == 255)
        {
```

```
        OCR2=0;
}
if(OCR0  == 255)
{
        OCR0=0;
        OCR2=255;
}
if(croisement == 1)
{
        croisement = 0;
        OCR0=255;
}
if(feux_avant == 1)
{
        feux_avant = 0;
        croisement  = 1;
}
if(feux_arriere == 1)
{
        feux_arriere = 0;
        feux_avant  = 1;
}
if(feux_recul == 1)
{
        feux_recul = 0;
        feux_arriere  = 1;
}
```

```
                if((feux_recul == 0)&&(feux_arriere == 0)&&(feux_avant == 0)&&(croisement == 0)&&(OCR0  == 0)&&(OCR0  == 0))
                {
                        feux_recul = 1;


                }

        }
        if(FlagTunnig2==1)
        {
            if(OCR2  == 0)
                {
                        OCR2=255;
                }
                if(OCR0  == 0)
                {
                        OCR0=255;
                        OCR2=0;
                }
                if(croisement == 0)
                {
                        croisement = 1;
                        OCR0=0;
                }
                if(feux_avant == 0)
                {
                        feux_avant = 1;
                        croisement  = 0;
```

```
                    }
                    if(feux_arriere == 0)
                    {
                            feux_arriere = 1;
                            feux_avant  = 0;
                    }
                    if(feux_recul == 0)
                    {
                            feux_recul = 1;
                            feux_arriere  = 0;
                    }
                    if((feux_recul == 1)&&(feux_arriere == 1)&&(feux_avant == 1)&&(croisement == 1)&&(OCR0  == 255)&&(OCR0  == 255))
                    {
                            feux_recul = 0;

                    }
            }

    }

    //variables globales
    //Déclaration des Sorties

    //Programme principale
    void main(void)
    {
```

```
//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;
// 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: 3,906 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x0f;
OCR1AL=0x3c;
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;


MCUCR=0x00;
MCUCSR=0x00;

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

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;



// Global enable interrupts
#asm("sei")

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;
#asm("sei")
while (1)
{



   if(BP_eclairage==0) //Mettre ON
   {

      if(BP_warning==0)
      {
         Cligno_G=0;
               Cligno_D=0;
               Warning=1;
```

```
}
else
{
        Warning=0;

        if(BP_cligno_g==0)
        {
                Cligno_G=1;
        }
        else
        {
                Cligno_G=0;
        }
        if(BP_cligno_d==0)
        {
                Cligno_D=1;
        }
        else
        {
                Cligno_D=0;
        }
}
if(BP_mode) //MODE MANUEL
{
        if(BP_feux_route==0)
        {
                OCR2=255;
```

```
        }
        else
        {
                OCR2=0;
        }

        if(BP_feux_crois==0)
        {
                croisement =1;

        }
        else
        {
                croisement =0;

        }
        if((BP_feux_route==0)||(BP_feux_crois==0))
        {
                feux_arriere =1;
        }
        if((BP_feux_route==1)&&(BP_feux_crois==1))
        {
                feux_arriere =0;
        }
        FlagTunnig1=0;
        FlagTunnig2=0;
}
```

```
else
{
        if(BP_feux_route==0)
        {

                FlagTunnig1=1;
        }
        if(BP_feux_crois==0)
        {

                FlagTunnig2=1;
        }
        if((BP_feux_route==1)&&(BP_feux_crois==1))
        {
                FlagTunnig1=0;
                FlagTunnig2=0;
                Moyenne=Moyenne+read_adc(3);
                N++;
                if(N>=100)
                {
                        OCR0=(Moyenne/100)/2.15;
                        Moyenne=0;
                        N=0;
                }
                feux_recul =0;
                feux_arriere  =0;
                feux_avant =0;
                croisement =0;
```

```
                luminosite=luminosite+read_adc(2);
                J++;
                if(J>=100)
                {
                        luminosite=(luminosite/100);
                        OCR2=(((luminosite-70)/2) / (10*exp(-(luminosite-70)/4)+1));
                        luminosite=0;
                        J=0;
                }
        }


    }




}
else
{

    feux_recul =0;
    feux_arriere  =0;
    feux_avant =0;
    cligno_g =0;
    cligno_d =0;
    OCR0=0;
    OCR2=0;
```

```
            croisement =0;


        }
    }
    }
```