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

SYSTEME D'ECLAIRAGE ET SIGNALISATION D'UN KARTING : Programmation

Rafizzi ROSLI
2e Année – Groupe P2
2008 - 2010

Enseignants
M. Thierry LEQUEU
Mme. Véronique AUGER

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

SYSTEME D'ECLAIRAGE ET SIGNALISATION D'UN KARTING :

Programmation

Rafizzi ROSLI
2e Année – Groupe P2
2008 - 2010

Enseignants
M. Thierry LEQUEU
Mme. Véronique AUGER

Sommaire

Introduction

Au cours de ce quatrième semestre au sein du département GEII, j'ai eu à choisir un projet à réaliser parmi plusieurs sujets liés au karting électrique. Mon choix s'est porté sur l'éclairage du kart électrique.

L'année précédente, un groupe d'étudiants a réalisé une carte électronique permettant de contrôler l'éclairage et la signalisation d'un karting en utilisant un micro-contrôleur programmable. Cependant, ce travail n'a pas été achevé et la programmation réalisée avec l'aide du logiciel « Code Vision AVR » n'a pas été remise au professeur. J'ai donc souhaité rependre cette expérience en programmant le micro-contrôleur AtMega8535.

Dans ce dossier du projet tutoré, mon plan est le suivant : nous allons tout d'abord analyser, puis comprendre le fonctionnement de la carte précédemment créée. Nous pourrons ainsi tester l'éclairage et la signalisation en utilisant le boîtier de commande de la carte. Enfin, nous pourrons, si nécessaire, apporter des améliorations.

1 Cahier des charges

Projet : Programmation pour l'éclairage d'un karting

Comme évoqué dans l'introduction, ce projet consiste en la programmation d'un micro-contrôleur AtMega8535 qui a déjà implanté sur la carte de signalisation et de l'éclairage d'un karting. Cette programmation va permettre de contrôler et améliorer l'éclairage pour un karting.

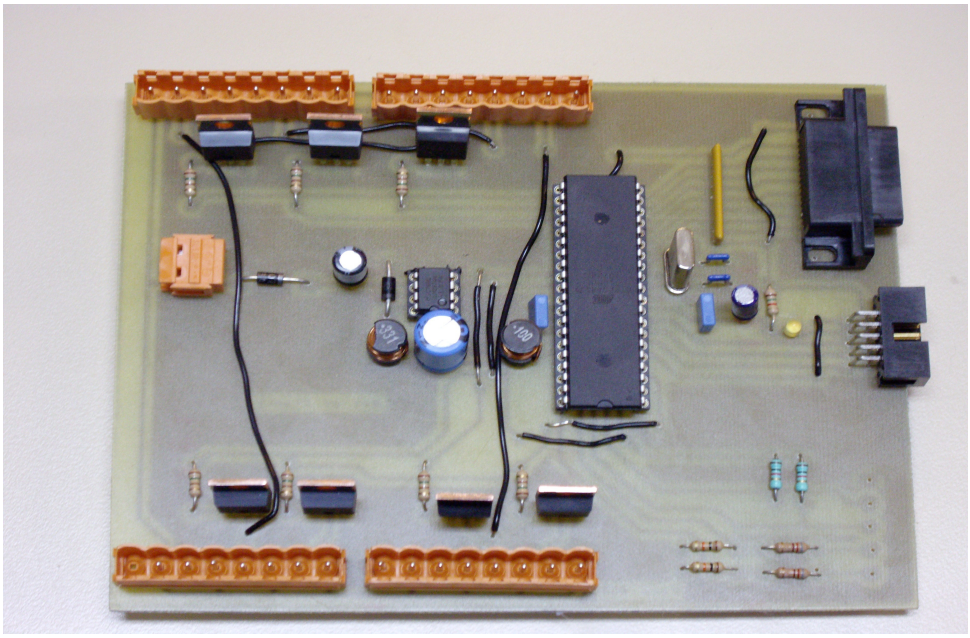


Figure 1 : carte électronique avec un microcontrôleur AtMega8535 [1]

1.1 Contraintes :

–Étude du projet réalisé précédemment
–Étude du micro-contrôleur AtMega8535
–Programmation du micro-contrôleur pour :

- Feux de position
- Clignotants
- 'Warning'
- Feux de recul
- Feux de stop
- Feux de croisement

–Utilisation du logiciel « Code Vision AVR »
–Respect des normes de sécurité

1.2 Synoptique général

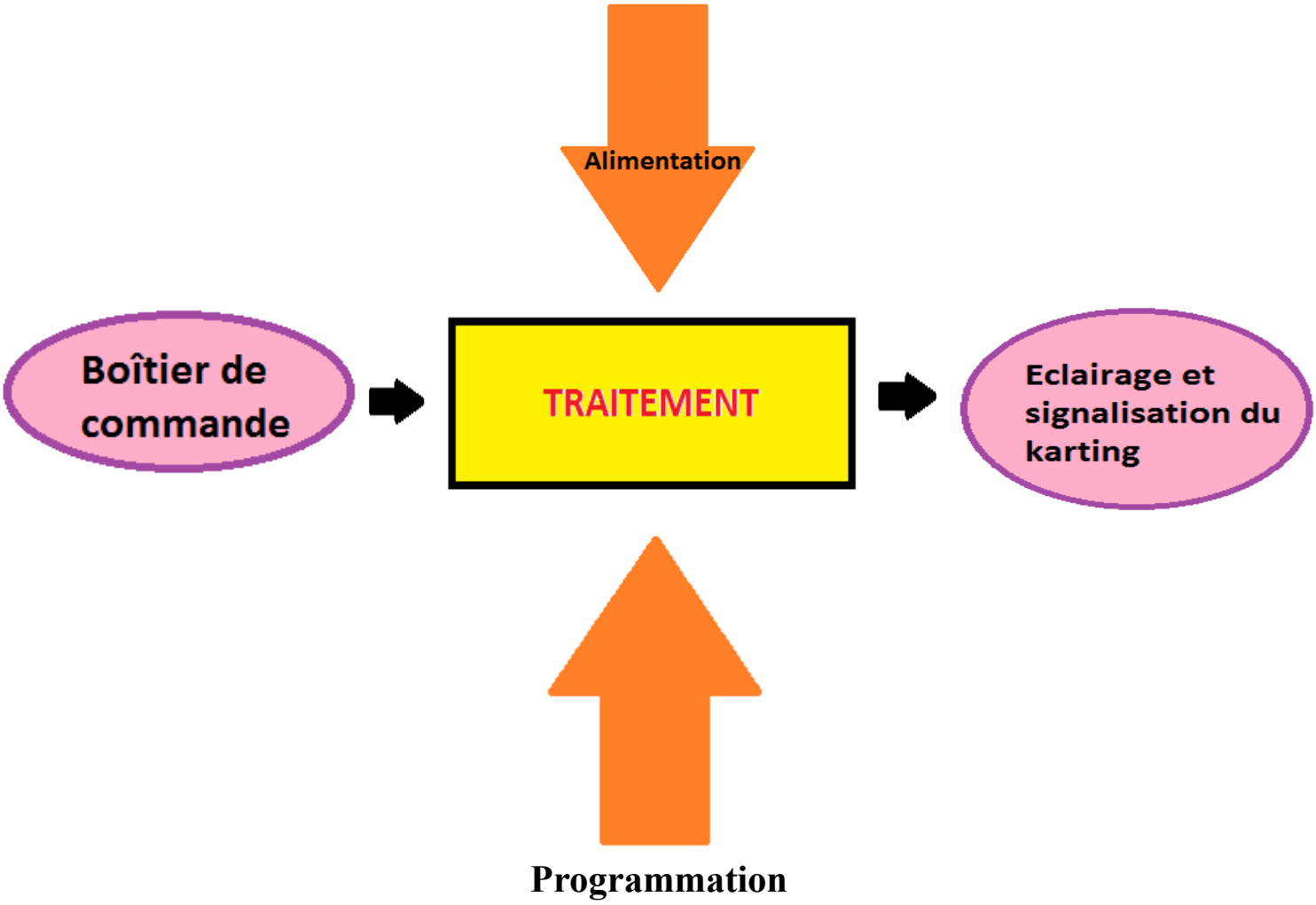


Figure 2 : Schéma synoptique général du projet [2]

1.3 Planning prévisionnel

Afin de garder une bonne organisation, je prévois la répartition des tâches dans un planning.

N° Semaine / Tâche	3	4	5	6	7	8	9	10	11	12	13
Choisir le sujet											
Prise de connaissance du projet précédent et du logiciel Code Vision AVR											
Programmation											
Test et validation											
Rédaction du rapport											
Séance PT											
Préparation de la présentation orale											
Oral											

Vacances

Planning prévisionnel

2 Étude du projet précédemment réalisé

2.1 Présentation

Les étudiants avaient décidé de choisir un équipement standard (c'est-à-dire réglementaire au code de la route) pour l'éclairage et la signalisation du karting électrique. Le système comporte l'équipement suivant :

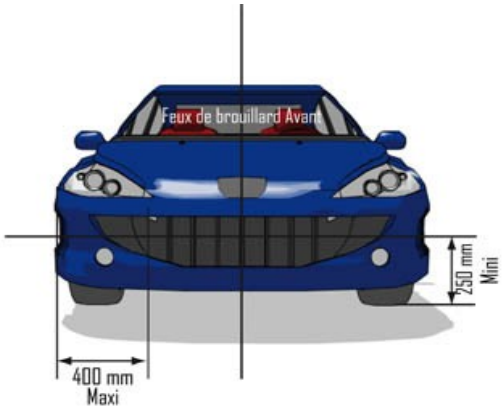
Éclairage	Signalisation
- Feux de position - Feux de recul - Feux de croisement	- Feux stop - Clignotants - 'Warning'

Tableau 1 : partie d'éclairage et partie de signalisation dans un karting

Ce principe est identique à une voiture, comme le montrent des images ci-dessous:

Feux stop

Feux de route + Feux de croisement



Feux de brouillard avant

Figure 3 : exemple de l'éclairage et de la signalisation pour une voiture [3]

2.2 La maquette et boîtier de commande des entrées

Pour leur expérience, ils avaient réalisé une maquette comportant les différentes parties étudiées : les différentes lampes, la carte électronique incluant un micro-contrôleur programmable, ainsi qu'un boîtier de commande afin de contrôler les différentes fonctions apportées par le système.

Figure 4 : photo de la maquette [4]

De plus, ils ont conçu un boîtier de commande des entrées afin de commander l'éclairage ainsi que les autres fonctions de la carte électrique. Il comporte les éléments suivants :

- Alimentation générale ON/OFF
- Un bouton trois positions commandant les clignotants gauche et droit (avec une position repos).
- Une commande pour des 'warnings'.
- La commande de marche arrière sera simulée par un interrupteur
- Le choix du mode d'éclairage : automatique/manuel.
- La mise en marche des feux de position.
- La mise en marche des feux de routes.

Figure 5 : photo du boîtier de commande [5]

2.3 *Le programme de test*

Pendant les trois première séances, je me suis concentré sur les entrées et les sorties du projet. J'ai alimenté la maquette sous une tension de 12V. Ensuite, j'ai relié la carte électronique (détaillée dans la partie suivante) à un ordinateur grâce à un connecteur de programmation.

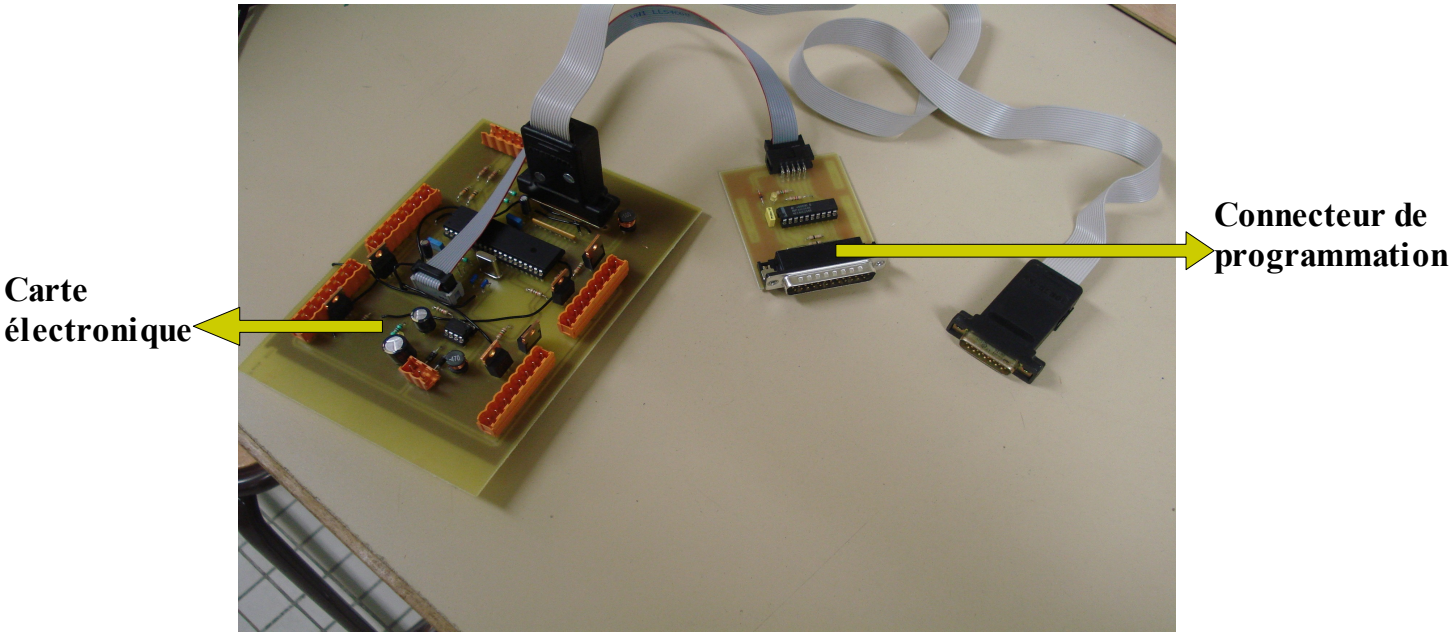


Figure 6 : carte électronique avec le connecteur de programmation[4]

Une fois cette étape faite, je lance le logiciel « Code Vision AVR » dans lequel j'écris des lignes de programmation. (voir les annexes pour les lignes de programmation complètes). Afin de tester le fonctionnement de la carte, je réalise le petit programme simple ci-dessous.

```
while (1)                    // Le programme tourne en boucle ici
{
      delay(1000);           // Temporisation de 1s
      if(Lampe) Lampe = 0;   // Lampe ON
      else Lampe = 1;        // Lampe OFF
}
```

Ce code me permet de faire clignoter l'ensemble des lampes quelque soit l'état des entrées. Je peux vérifier que toutes les sorties sont correctement reliées grâce à ce programme.

2.4 La carte électronique

Avant de commencer le projet, il nous faut étudier la carte électronique. Cette étape est importante afin de bien comprendre son fonctionnement, et de suivre le 'datasheet' correctement (surtout pour l'alimentation de la carte).

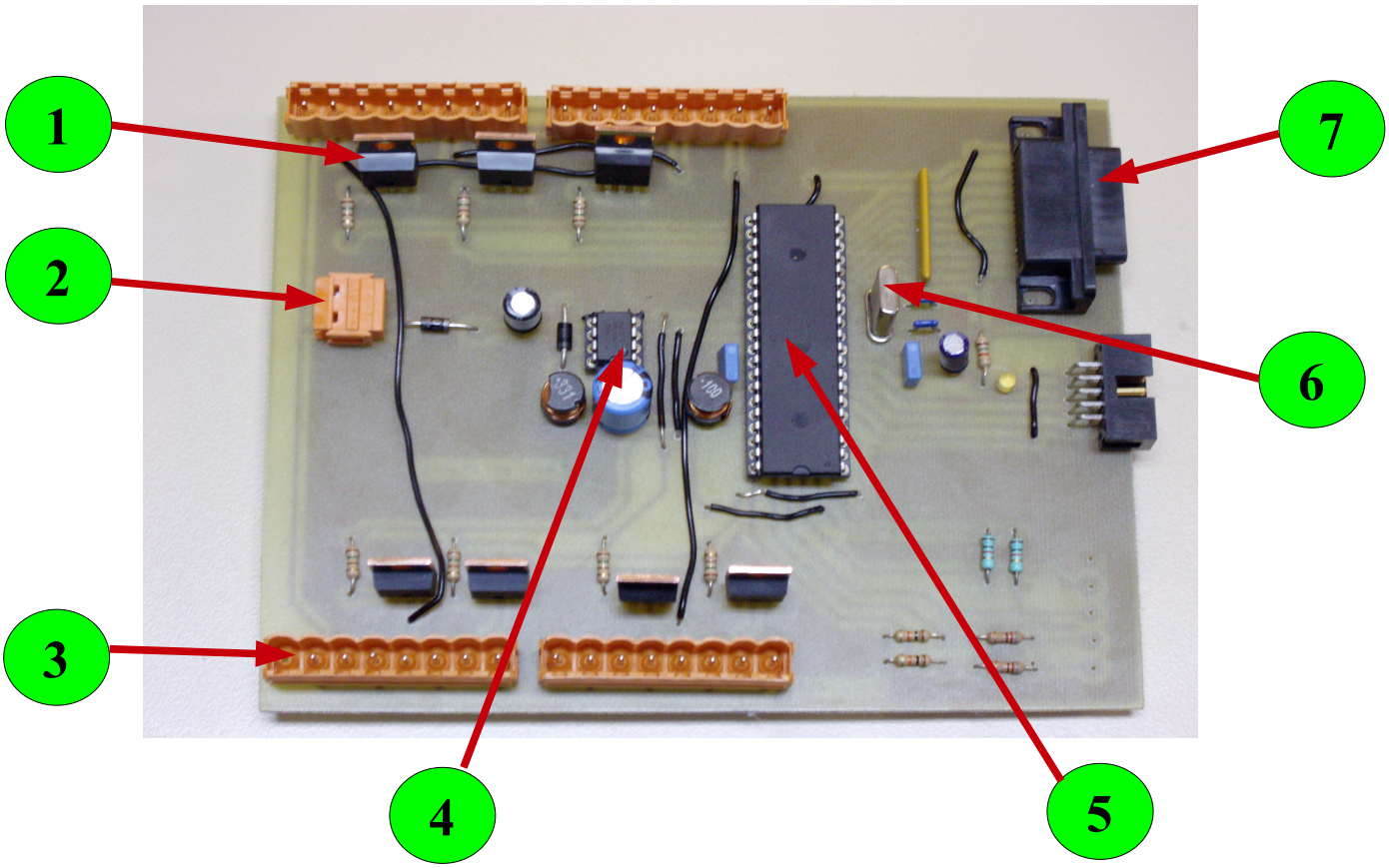
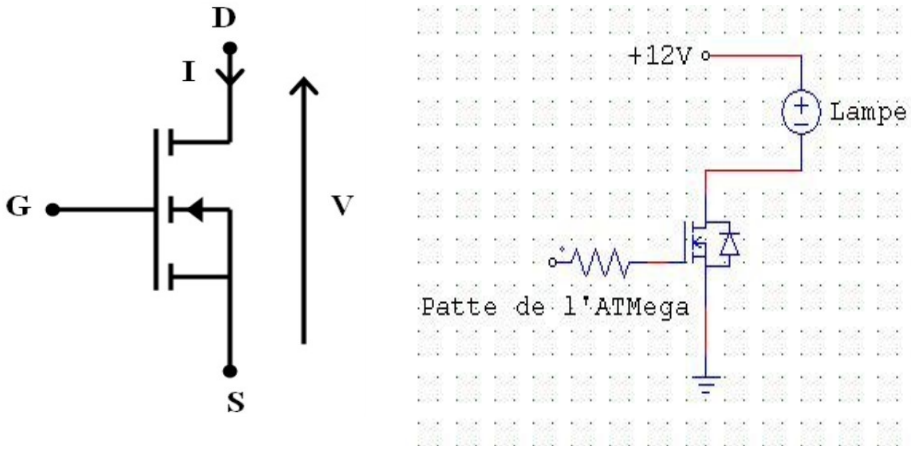


Figure 7 : carte électronique [1]

Indication :

	Composant	Description
1	Transistors MOSFET [1]	-D'après le cours d'électronique de première année, ce composant est un transistor à effet de champ (TEC) ou « Field Effect Transistor » (FET) en anglais. -Il sert d'interrupteur de sortie et se compose comme le bipolaire de trois broches – D (Drain), S (Source) et G (Grille). Le drain est relié à l'une des broches une lampe (l'autre étant reliée à +12V), la source rejoint la masse, et la grille est reliée directement à l'AtMega8535 et recevra des impulsions. -Lorsque la sortie du micro-contrôleur est activée, la tension appliquée à la grille du transistor ferme le contact entre le drain et la source. Ainsi, la lampe est reliée entre la masse et le 12V, elle s'allume donc. Au contraire, quand la tension de sortie du micro-contrôleur est à 0V, le contact dans le transistor est ouvert, la lampe n'est plus reliée à la masse, elle ne peut donc pas fonctionner et reste éteinte.  Figure 8 : schéma du transistor MOSFET [4]
2	Alimentation	Sur ce connecteur, je vais récupérer une tension d'alimentation de 12V (le kart possède aussi deux batteries de 12V).

1 MOSFET : Metal Oxide Semiconductor Field Effect Transistor

3	Connecteurs lampes (pour les sorties)	Ces connecteurs permettent la liaison entre les LEDs[2] et la carte. Ils sont directement branchés aux LEDs aussi.
4	Régulateur de tension à découpage LM2574N	-L'utilisation d'une alimentation à découpage de type BUCK permet d'abaisser une tension continue en entrée. -Ainsi, grâce au LM2574, il est capable de fournir une tension continue de 5 Volts à l'ATMega853 à partir d'une tension continue de 12 Volts délivrées par les batteries du kart électrique  Figure 9 : photo du régulateur à découpage LM2574N [6]
5	Micro-contrôleur l'AtMega8535	-C'est un micro-contrôleur 8 bits et se présente sous la forme d'un circuit intégré de 40 branches. -C'est à l'intérieur de ce composant que j'implanterai le programme. Il possède de nombreuses entrées et sorties que je vais expliquer dans la prochaine partie.
6	Quartz	-Le quartz est de la silice (SiO^2) cristallisée. C'est un transducteur, qui convertit l'énergie électrique en énergie mécanique et inversement. -Dans les oscillateurs à quartz, le résonateur est toujours placé de manière à osciller dans sa zone inductive (sous l'influence d'un champ électrostatique alternatif). Il en résulte une grande précision de la fréquence d'oscillation obtenue. -Le quartz utilisé dans ce projet est cadencé à 16 MHz, il permet de remplacer l'horloge interne de l'AtMega8535 qui est cadencée à une fréquence moins élevée.
7	Connecteurs d'entrées	-Connecteurs branchés au boitier de commandes permettront la liaison entre les interrupteurs, les capteurs et la carte.

Tableau 2 : descriptions générales des composants principaux utilisés

2 LED : Light Emitting Diod (en anglais) ou DEL : Diode Electroluminexente (en français)

2.5 Le micro-contrôleur AtMega8535

Pour réaliser la commande globale de ce projet, il est nécessaire de définir les actions à affecter aux différentes entrées. Pour cela, l'utilisation d'un micro-contrôleur AtMega8535 est utile car il offre 4 ports (A,B,C et D) ayant chacun des fonctions particulières et contenant 8 bits directionnels numérotés de 0 à 7.

En général, il y a 4 types de micro-contrôleur « Atmel » :

- AtTiny
- AtMega
- At90Sxx
- At86RFxx

Il existe quelques différences entre ces 4 types : au niveau d'application, mémoire et périphérie. L'AtMega8535 peut fonctionner jusqu'à 16MHz et il est efficace.

Figure 10 : illustration de l'AtMega8535 [7]

2.5.1 Description des broches

Pendant la séance d'étude et réalisation, je définis les pattes de l'AtMega8535 avec M.Lequeu. Chaque port peut être configuré comme entrée ou bien comme sortie. Ce composant possède aussi plusieurs autres pattes que je vais lister dans la page suivante:

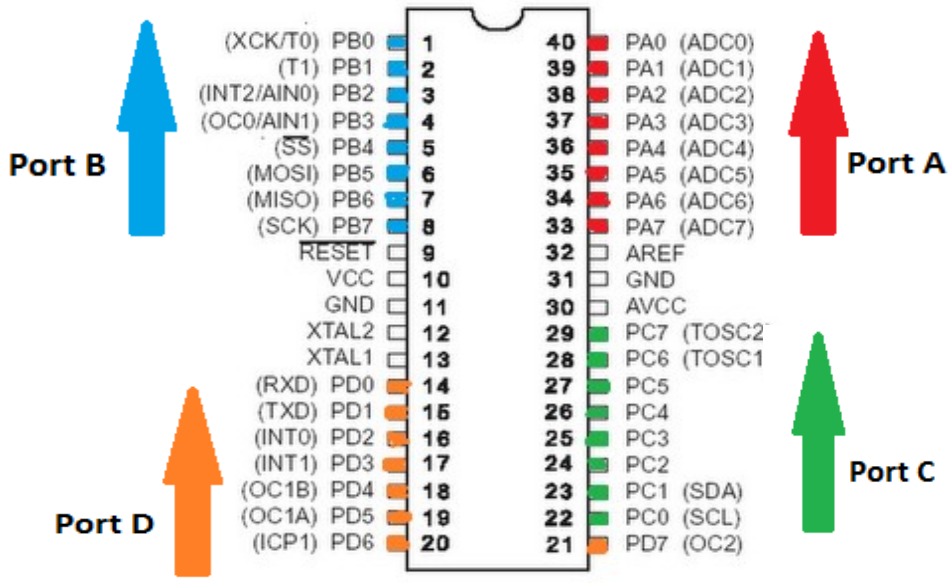


Figure 11 : affectation des pattes d'AtMega8535 [8]

Broche	Fonction
VCC	Broche d'alimentation du micro-contrôleur
GND	Masse de l'alimentation
RESET	Permettre la réinitialisation du micro-contrôleur
XTAL1	Broche pour l'oscillateur externe pour l'horloge interne (quartz)
XTAL2	Production de l'amplificateur d'oscillateur
AVCC	Broche d'alimentation pour le CAN (qui doit être reliée à VCC)
AREF	Entrée de référence analogue pour le CAN
AGND	Masse analogique

Tableau 3 : fonctions des broches

De plus, l'AtMega8535 possède trois sorties pouvant être programmées en modulation de largeur d'amplitude (MLI), ce qui permet de faire varier la valeur moyenne de la tension d'alimentation des feux avant et arrière (les feux de position) et l'intensité reçue par les lampes et leur intensité d'éclairage.

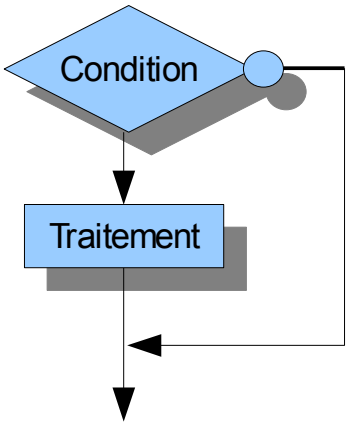
3 Réalisation du programme

3.1 Description d'un organigramme

Toutes les extrémités d'un symbole doivent posséder un chemin et chaque structure de base n'a qu'une seule entrée et qu'une seule sortie. De plus, les structures peuvent être imbriquées les unes dans les autres (c'est-à-dire, une opération de traitement peut être composée d'une ou de plusieurs structures). Voici des descriptions générales pour chaque organigramme que j'utilise pendant ce projet :

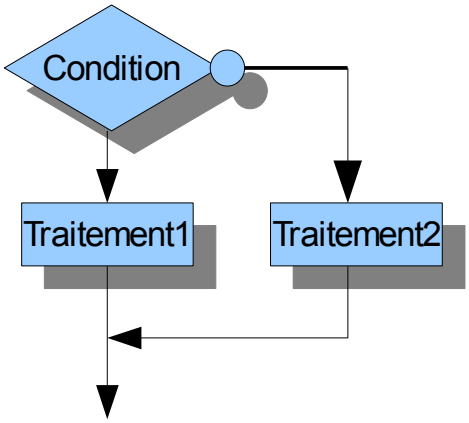
- 

  Cet élément est utilisé pour un appel de sous-programme.

- 

  Cette structure est une alternative simple. Elle permet d'effectuer un traitement seulement si la condition est vraie. Sa ligne de programmation est de la forme suivante :

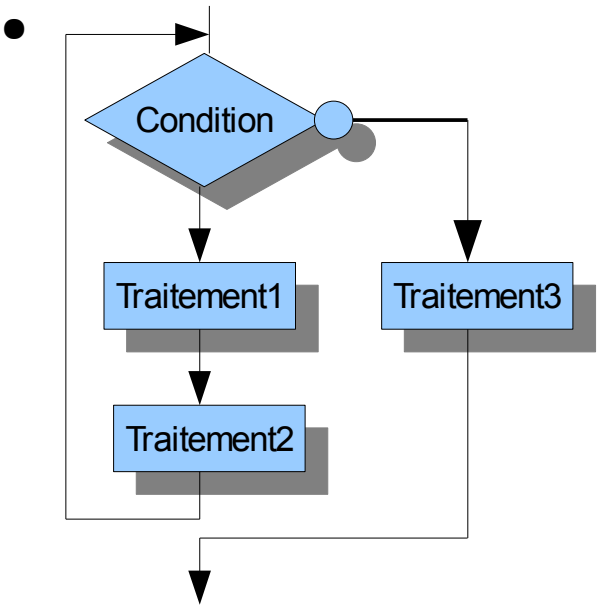
```
if ( Condition )
{
    // Traitement ;
}
```

- 

  Ceci est une alternative double. Elle permet d'effectuer un traitement seulement si la variable possède une condition.

  Si la condition est fausse, le second traitement s'effectuera. Sa ligne de programmation est de la forme suivante :

```
if ( Condition )
{
    // Traitement1 ;
}
else    //sinon
{
    // Traitement2 ;
}
```



Voici un exemple de structure répétitive. C'est une boucle consiste à refaire une ou plusieurs actions tant que la condition de boucle est fausse.

Dans ce projet, j'utilise une boucle infinie pour permettre le programme de ne jamais s'arrêter.

Exemple de code :

```
while ( Condition )  //tant que
{
    Traitement1;
    Traitement2;
}
else  //sinon
{
    Traitement3;
}
```

3.2 Organigramme de programmation

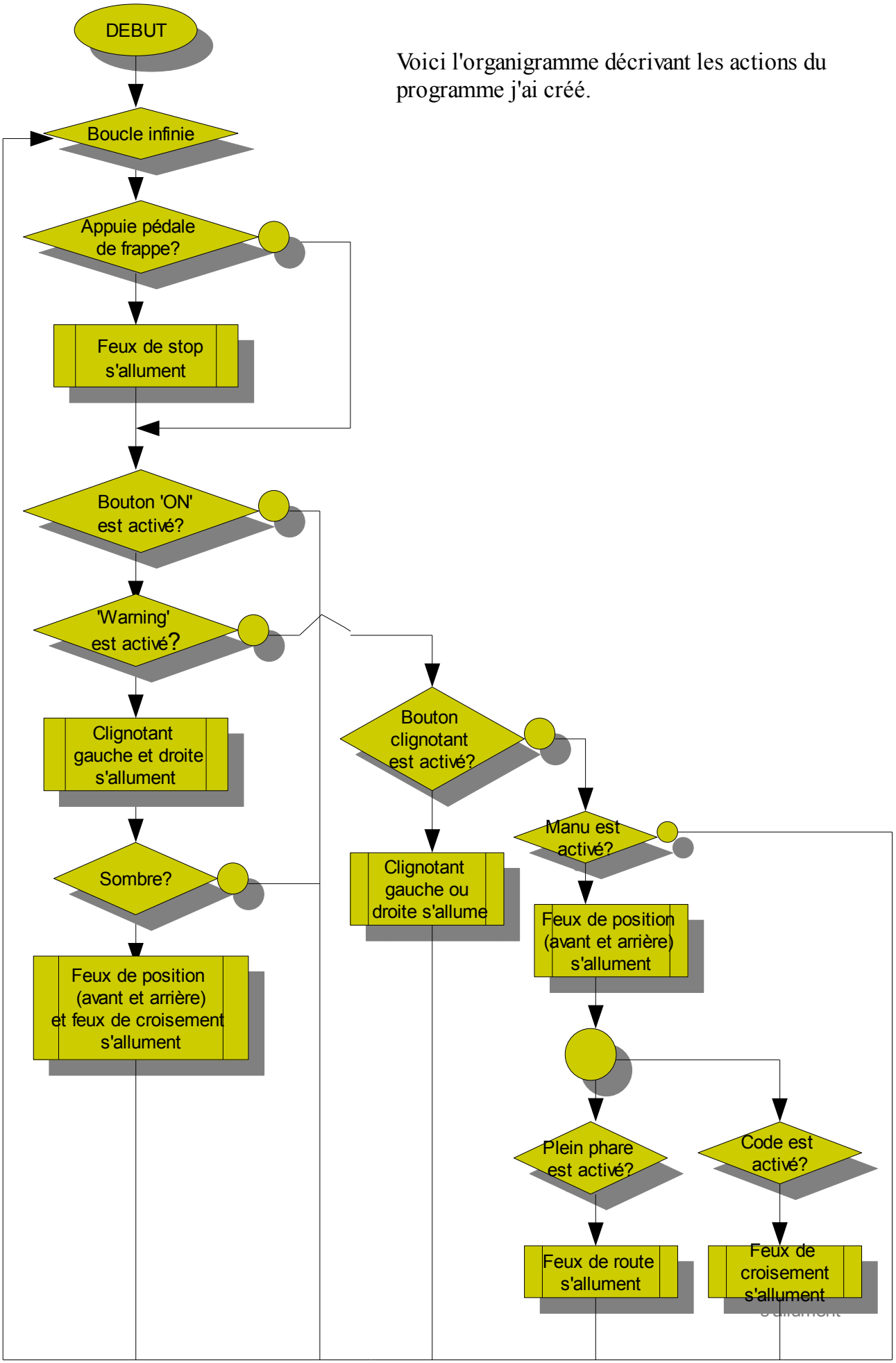
Avant que j'écrive des lignes de programmation, il va falloir de réfléchir un chemin de fonctionnement, ce qu'on appelle un organigramme de programmation. Dans cette partie, je vais décrire les actions de programmation que j'ai décidé et ensuite, les traduire sous la forme d'un organigramme.

Tout d'abord, le programme sera géré par une boucle infinie puisque ce projet nous impose d'assurer le fonctionnement de l'éclairage sans interruption. Quand on appuie sur une pédale de frein (qui est le rôle du potentiomètre de mécanique dans ce projet), les feux de stop vont s'allumer. Ensuite, le bouton 'ON' sur une boîte de commande active tous les éclairages et les signalisations.

Après, quand on met le bouton 'Warning' en actif, les clignotants gauches et droites vont clignoter en même temps. Si les environs sont sombres, les feux de position (avant et arrière) et les feux de croisement s'allument automatiquement à l'aide d'une photorésistance. Le bouton 'Warning' est prioritaire sur les clignotants.

Quand le bouton clignotant est activé, le clignotant gauche ou droite s'allume. Sinon, pour allumer les feux de position, il va falloir d'activer le bouton « manuel ». Les feux de route sont mis en fonction quand le plein phare est en actif, et les feux de croisement s'allument quand le code est activé.

Voici l'organigramme décrivant les actions du programme j'ai créé.



3.3 Programmation des ports d'entrées / sorties

Grâce au logiciel Code Vision AVR, j'ai pu modifier et paramétrer les ports du micro-contrôleur. Voici les informations sur les ports que j'ai appliqué dans ma programmation.

```
// 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=Out Func6=Out Func5=In Func4=Out Func3=In Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=T State4=0 State3=T State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xD7;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x80;
```

Tous les broches dans le port A sont paramétrées en entrée (IN). Également pour les broches 0,1,2,4,5,6,7 du port B, sauf pour la broche 3 de ce port qui est paramétrée en sortie (OUT). Pour le port C, les broches 3 et 5 sont en entrées, et les autres broches sont en sortie. Et le dernier port, le port D, les broches 1 jusqu'à 6 sont programmées comme les entrées, sauf pour la broche 7 qui est programmée comme la sortie.

3.4 Définition des symboles utilisés

Dans la partie de variables globales, je déclare plusieurs mnémoniques pour faciliter les écritures et aussi rendre le programme plus lisible. Les mnémoniques comportent les éléments d'entrées ainsi que les éléments de sorties.

ENTREES		
Pattes	**Mnémonique**	**Désignation**
PIND.0	BP_mode	Interrupteur mode : manuel/Automatique
PIND.1	BP_Warning	Interrupteur Warning
PIND.2	BP_clign_d	Interrupteur clignotant droit
PIND.3	BP_clign_g	Interrupteur clignotant gauche
PIND.4	BP_feux_croisement	Interrupteur feux de croisement
PIND.5	BP_feux_position	Interrupteur feux de position
PIND.6	BP_eclairage	Interrupteur mode : éclairage

SORTIE		
Pattes	**Mnémonique**	**Désignation**
PINC.0	feux_recul	Sortie feux de recul
PINC.1	eclairage_arriere	Sortie éclairage arrière
PINC.2	leds_bleu	Sortie LEDs bleues
PINC.4	leds_verte	Sortie LEDs vertes
PINC.6	clign_g	Sortie pour les clignotants gauches
PINC.7	clign_d	Sortie pour les clignotants droits
PINB.3	eclairage_avant	Sortie pour l'éclairage avant
PIND.7	feux_stop	Sortie pour les feux stops

Tableau 4 : définition des symboles des entrées et des sorties

3.5 Fonction MLI et sa programmation

Qu'est-ce que c'est la MLI? La MLI (Modulation de Largeur d'Impulsion) ou PWM[3] en anglais, est une façon de générer une tension moyenne variable à partir d'une tension continue fixe (qui est la tension d'alimentation), en modulant le rapport cyclique d'un signal périodique.

Pour la définition d'une impulsion, dans le domaine physique, est une variation brusque d'une grandeur physique suivie d'un retour rapide à sa valeur initiale. Les trois graphes ci-dessous montrent un même signal rectangulaire, mais avec les trois valeurs de rapport cyclique différentes.

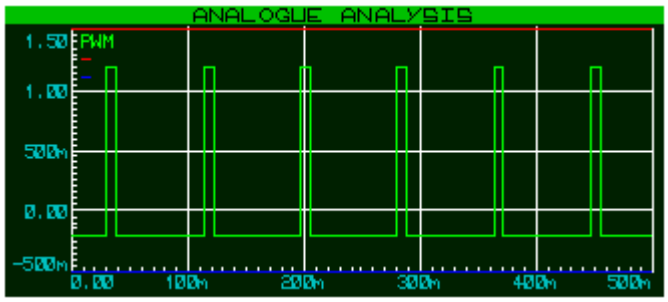
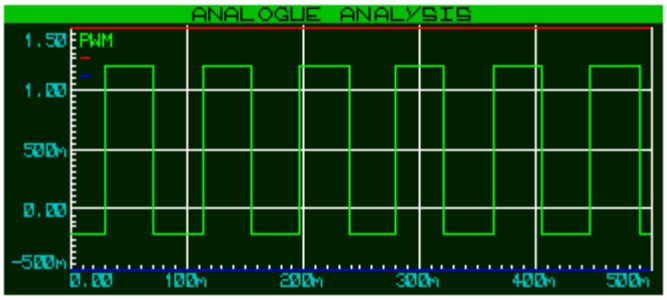
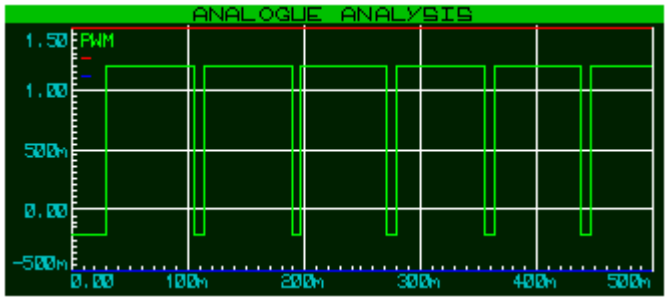
	Graphe	Commentaires
1)		Ce signal rectangulaire, avec la fréquence de 12 Hz, présente un rapport cyclique de 10%, ce qui signifie que le niveau électrique reste à l'état haut pendant 10% du temps total d'un cycle (un cycle est duré 1/12 secondes). Quand il est à l'état haut, la valeur moyenne est faible, par rapport à l'amplitude maximale qu'il possède.
2)		Le même signal rectangulaire avec la même fréquence présente ici un rapport cyclique de 50%, ce qui signifie que la durée pendant laquelle le signal reste à l'état haut, est identique à la durée pendant laquelle il reste à l'état bas. La valeur moyenne est égale à la moitié de son amplitude maximale.
3)		Le même signal rectangulaire avec la même fréquence présente cette fois un rapport cyclique de 90%, il reste plus longtemps à l'état haut qu'à l'état bas. La valeur moyenne est élevée.

Tableau 5 : graphes avec les trois valeurs de rapport cyclique différentes

3 **P**ulse **W**idth **M**odulation

Grâce à la fonction MLI, je peux l'appliquera directement sur l'éclairage variable les feux de stop et aussi les feux de position et les feux de croisement, en faisant varier le rapport cyclique. Voici les lignes de codes générées par le logiciel Code Vision AVR.

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x64;
TCNT2=0x00;
OCR2=0x00;
```

D'après ces lignes, elles indiquent que le « Timer[4] 2 » est programmé à l'horloge interne du micro-contrôleur qui est cadencée à 250 MHz. Elles montrent aussi que le « Timer 2 » sera en mode MLI. De plus, le rapport cyclique du signal de sortie peut être varié en changeant la valeur du registre OCR2. Ce registre est codé sur 8 bits (1 Octet) ainsi il peut varier de 0 (qui correspond à 0V) à 255 (qui correspond à 5V).

Figure 12 : application de MLI sur les feux de stop [2]

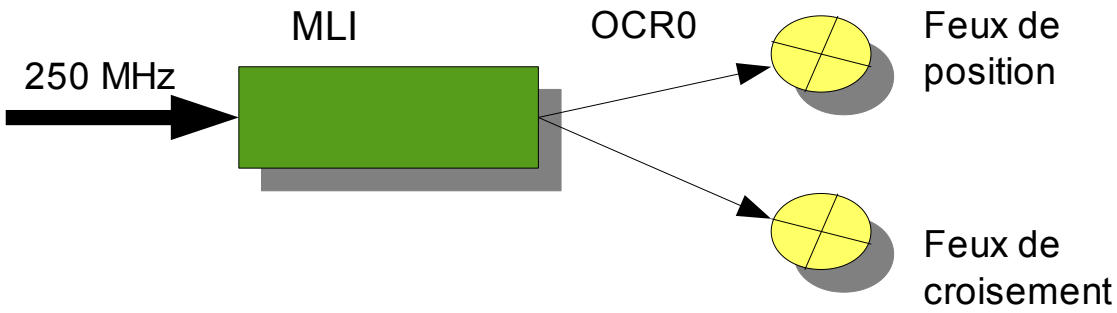


Figure 13 : application de MLI sur les feux de position et de croisement [2]

4 Timer signifie un compteur

3.6 ***Programmation du CAN***

CAN (Convertisseur Analogique Numérique) ou ADC[5] en anglais, est utilisé pour générer une valeur numérique à partir d'une valeur analogique. Il est situé sur le port D du micro-contrôleur AtMega8535. AVCC est une broche d'alimentation du CAN et AREF est sa broche de référence analogique.

Les données issues de la photorésistance et du potentiomètre mécanique sont des données analogiques, et c'est pourquoi l'application de CAN est nécessaire dans ce projet. Voici les lignes de codes qui sont générées par le logiciel Code Vision AVR.

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: On
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
```

Au début, elles initialisent la fonction convertisseur qui est cadencée à 125 MHz. Ensuite le CAN prend sa référence de tension sur la broche « AREF » (broche 32) et comme la fonction MLI, le CAN est codé sur 8 bits ou 1 Octet.

Pour utiliser le convertisseur, j'ai appliqué une fonction « read_adc(n) », ce qui signifie 'lire le CAN' avec 'n' est un numéro de broche. Par exemple, « read_adc(2) » est pour la photorésistance car elle est branchée sur l'entrée numéro 2 de micro-contrôleur, et « read_adc(3) » est pour le potentiomètre mécanique car il est connecté sur l'entrée numéro 3.

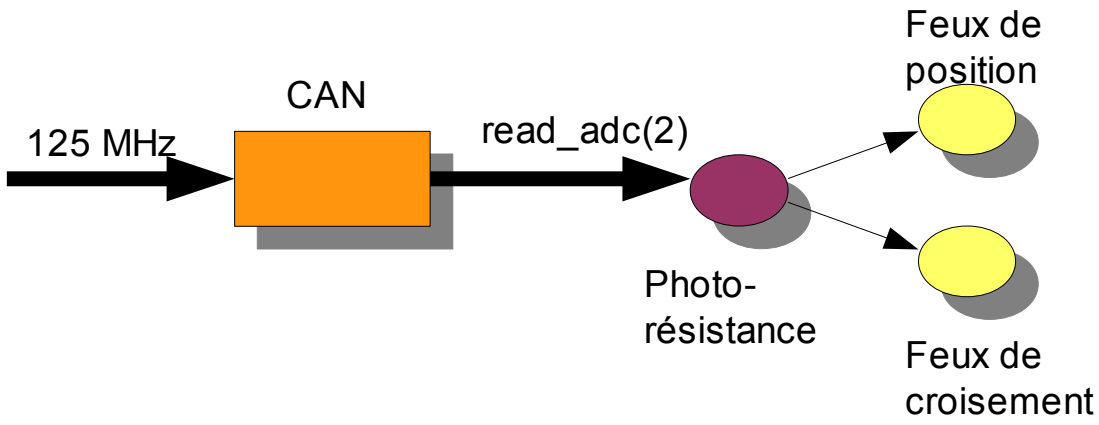


Figure 14 : application de CAN sur la photorésistance [2]

5 **A**nalogic **D**igital **C**onverter

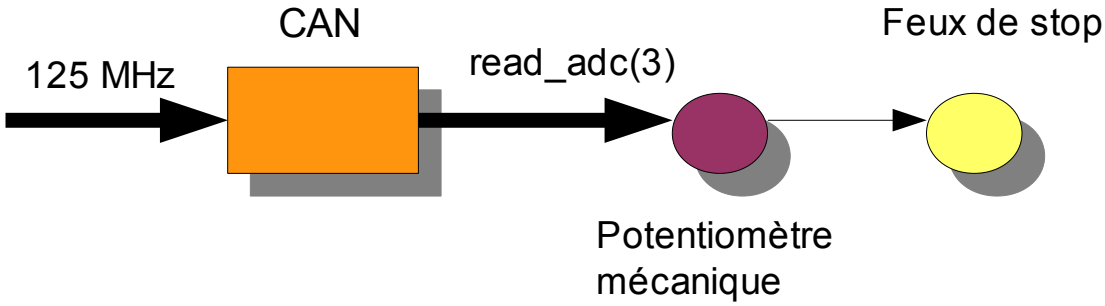


Figure 15 : application de CAN sur le potentiomètre mécanique [2]

3.7 Programmation des 'warnings' et des clignotants (droite et gauche)

Dans ce programme, j'ai utilisé la fonction delay_ms(t) qui permet de faire attendre le programme au « t » milisecondes. Voici extrait du programme que j'ai réalisé :

```
if(BP_Warning==0)                    //Si le bouton Warning est activé
{
        clign_d=!clign_d;            //Faire clignoter le clignotant droite
        clign_g=!clign_g;            //Faire clignoter le clignotant gauche
        delay_ms(500);               //Une durée de 0,5s
}
        else                         //sinon
{
                if(BP_clign_g==0)    //Si l'interrupteur clignotant gauche est activé
                {
                        clign_g=!clign_g;
                        delay_ms(500);
                }
                else
                        clign_g=0;

                if(BP_clign_d==0)    //Si l'interrupteur clignotant droite est activé
                {
                        clign_d=!clign_d;
                        delay_ms(500);
                }
                else
                        clign_d=0;
}
```

Pour répondre au cahier des charges, j'ai réalisé le programme de tel sorte à ce que les 'warnings' soient prioritaires sur les deux clignotants. Ça signifie que, quand j'active en premier le bouton warnings, les clignotants droites et gauches vont clignoter tout le temps, même que je désactive l'interrupteur clignotant droite ou gauche.

En plus, au lieu d'écrire ces lignes en dessous pour faire clignoter les clignotants;

```
{
        clign_g=0;
        clign_d=0;
        delay_ms(500);
        clign_g=1;
        clign_d=1;
        delay_ms(500);
}
```

Je peux les simplifier en écrivant :

```
{
        clign_d=!clign_d;
        clign_g=!clign_g;
        delay_ms(500);
}
```

4 Photorésistance

4.1 Fonctionnement de la photorésistance

Pour permettre la détection de la luminosité ambiante, on a besoin une photorésistance. C'est une résistance dont la valeur varie avec l'intensité de la lumière qui l'éclaire.

Si les environs sont sombres, la photorésistance va permettre les feux de position (avant et arrière) et les feux de croisement s'allument automatiquement. Dans ce cas, la résistance de ce capteur lumineux devient très grande. Sinon, sa résistance est petite lorsqu'elle est éclairée, donc les feux s'éteignent.

Figure 16 : photorésistance [9]

4.2 Étude théorique

Dans la partie précédente, on a vu que la fonction MLI va être utile pour avoir les différents niveaux de tension à la sortie du montage, car il suffit de modifier le rapport cyclique du signal.

Afin d'effectuer ces différents niveaux, tout d'abord, il faut acquérir la tension aux bornes de la photorésistance. A l'aide d'un wattmètre, j'ai mesuré la tension aux bornes de la photorésistance. Quand les environs sont sombres, j'ai trouvé que la tension est 4,96V (ce qui est proche la tension maximale, 5V), et quand les environs sont à jour, j'ai trouvé que la tension vaut 2,03V. Ensuite, j'ai appliqué le CAN du micro-contrôleur AtMega8535, et j'obtiens la tension codée sur 8 bits, ce qui signifie 0 pour le 0V et 255 pour le 5V.

La tension maximale qui est capable de fournir par l'AtMega8535 n'est que 5V. Pour cela, les étudiants précédents ont utilisé des montages Tout-Ou-Rien afin d'élever cette tension. Ils ont aussi fait des mesures sur les lampes et ils ont conçu qu'il fallait obtenir 3.4V, 4.2V et 5V (sans les montage Tout-Ou-Rien) pour avoir les différents niveaux de tension à la sortie.

Mon travail est donc de calculer le rapport de proportionnalité entre les valeurs analogiques et les valeurs numériques. Le tableau ci-dessous montre les valeurs que j'obtiens après mes calculs :

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

Tableau 6 : rapport de proportionnalité entre les valeurs analogiques et les valeurs numériques

Une fois tout est déterminé, je peux les utiliser dans la programmation.

4.3 Réalisation de la programmation

Sur le boîtier de commande, il y a de choix du mode d'éclairage : « automatique/manuel ». Pour le mode automatique, il est nécessaire de détecter l'intensité lumineux. Pour cela, j'ai effectué plusieurs tests avec la photorésistance pour déterminer les plages de luminosité :

Plage de luminosité	Action
0<Ne<125	Les phares sont éteints
125<Ne<173	Les feux de position sont allumés
173<Ne<220	Les feux de croisement sont allumés
N>220	Les feux de route sont allumés

Tableau 7 : Plage de luminosité avec les actions correspondantes

Voici l'extrait de programmation que j'ai réalisé pour le fonctionnement les feux de croisement et les feux de position à l'aide de la photorésistance.

```
float Ne ;
Ne=read_adc(2);              //lire le CAN pour l'entrée numéro 2 : photorésistance

if(Ne>125)                   //luminosité minimale pour feux de position à 2,5 V
  {
     PORTC.4=0;
     PORTC.2=0;
     OCR0=173;               //OCR0 = le registre qui fait varier le rapport cyclique
     eclairage_arriere=1;
  }
 if(Ne>173)                  //luminosité minimale pour feux de croisement à 3,4V
  {
      PORTC.4=0;
      PORTC.2=1;             //pour éclairer la sortie de LEDs bleues
      OCR0=214;
      eclairage_arriere=1;
  }
 if(Ne>220)                  //luminosité minimale pour feux de route à 4,3V
  {
      PORTC.2=1;
      PORTC.4=1;             //pour éclairer la sortie de LEDs vertes
      OCR0=255;
      eclairage_arriere=1;
  }
 else                        //sinon éteindre tous les feux
  {
      PORTC.2=0;
      PORTC.4=0;
      OCR0=0;
      eclairage_arriere=0;
  }
```

5 Potentiomètre mécanique

5.1 Fonctionnement du potentiomètre mécanique

Afin d'assurer la variation d'intensité lumineuse des feux de stop en fonction de la pression exercée sur la pédale de frein, il est utile d'utiliser un potentiomètre mécanique. Grâce à ce composant, je vais obtenir une tension variable qui sera mise en entrée de l'AtMega8535. Le but est de faire varier la valeur moyenne de la tension d'alimentation des feux de stop, et il est réalisable grâce à la fonction MLI ainsi que le CAN.

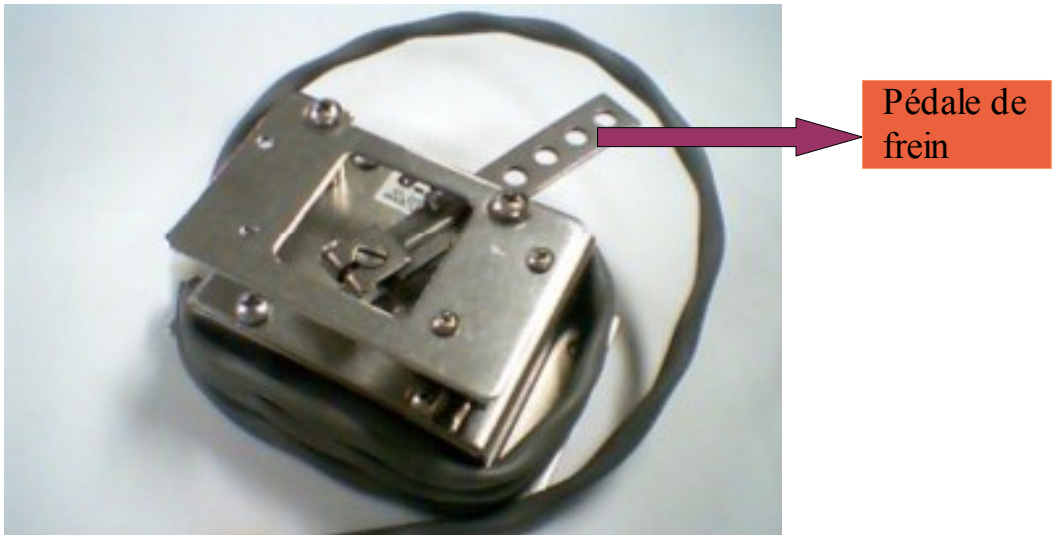


Figure 17 : potentiomètre mécanique [4]

5.2 Réalisation de programmation

Pour programmer les feux de stop, j'ai utilisé une équation de droite qui est sous la forme :

$$Y = aX + b$$

avec a = la pente de la droite
b = l'ordonnée à l'origine

Avec cette équation générale, j'ai pu calculer le rapport de proportionnalité afin de déterminer une variation continue de la luminosité. Pour les feux de stop, les ampoules utilisées sont constituées de LEDs. C'est pourquoi il est nécessaire de prendre en compte la tension de seuil de 0,6V. Pour cela, je dois considérer une valeur numérique non nulle lui correspond lorsque la tension d'entrée est nulle.

J'ai donc besoin de 0,6V minimum sur les lampes, ce qui correspond à 0,25V avant le transistor MOSFET, et 12,75 en valeur numérique. Ensuite, je souhaite aussi d'avoir une intensité lumineuse maximale pour les feux des stop lorsque la valeur numérique est en 255. Avec tous ces paramètres, je suis capable de tracer la droite, comme montre le graphique ci-dessous :

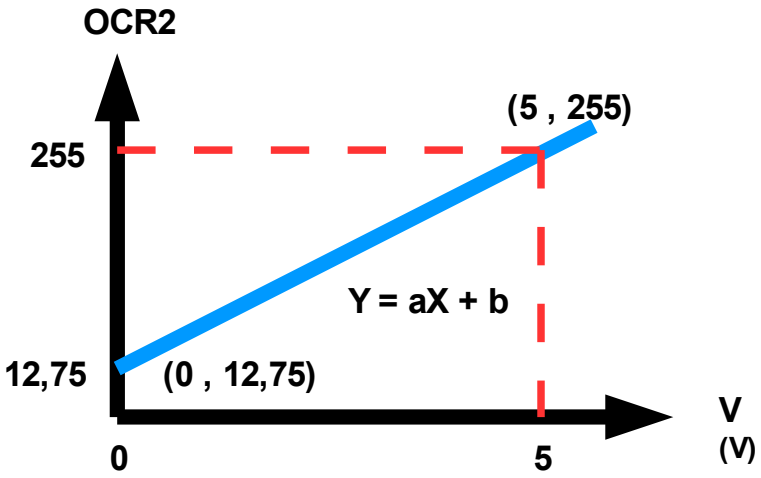


Figure 18 : graphique correspond à la variation continue de la luminosité [2]

J'obtiens l'équation de droite : OCR2 = 48,5*V + 12,75 avec a = (255-12,75)/(5-0) = 48,5. Une fois que tout est déterminé, je les applique dans la ligne de programmation.

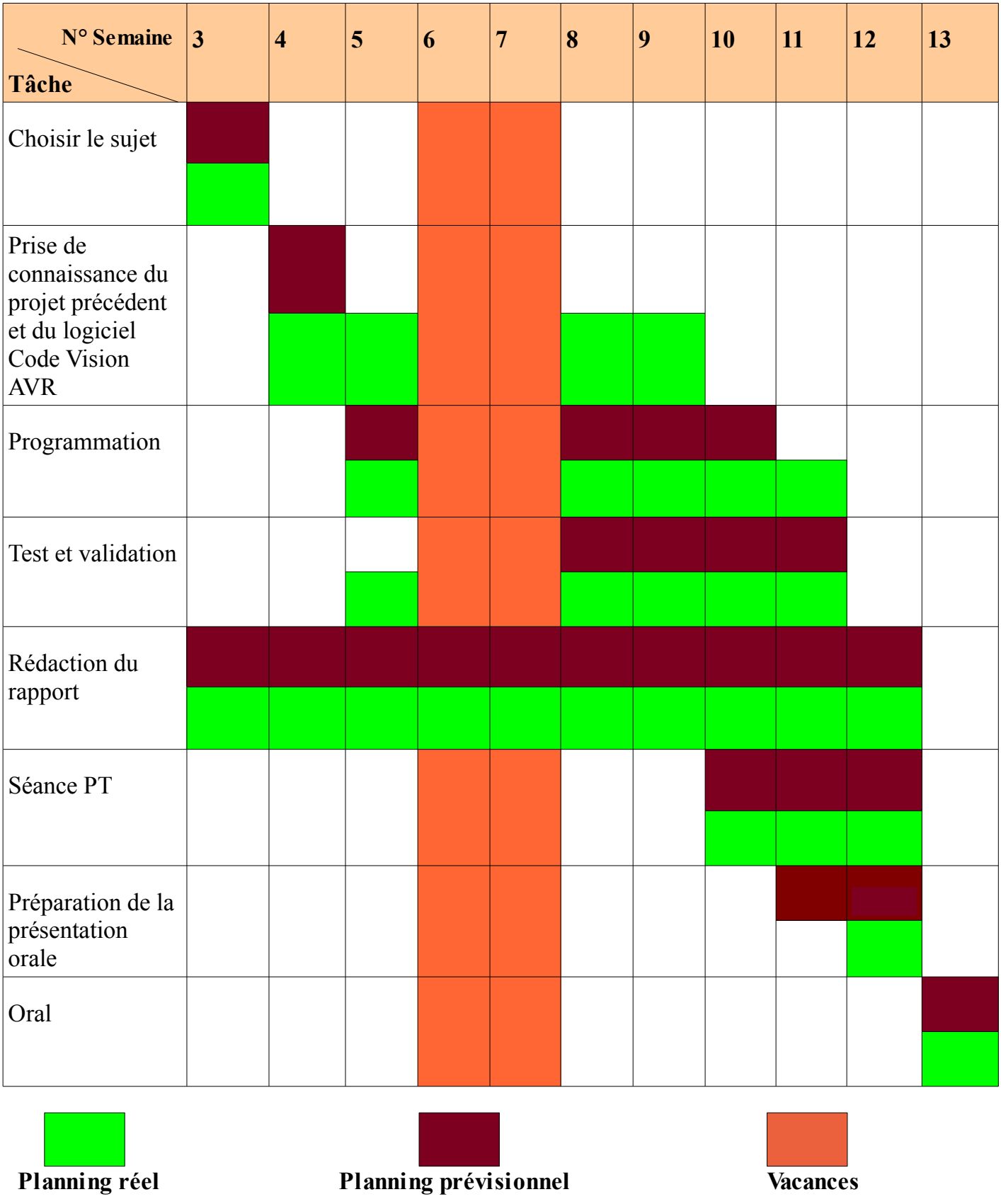
```
float Nf ;
Nf= read_adc(3);          //lire le CAN pour l'entrée numéro 3 : potentiomètre
                            mécanique
OCR2=(48.5*Nf + 12.75);   //allumer les feux de stop en fonction du potentiomètre
```

6 Coût de projet

Comme je continue le projet d'étudiants précédants, je n'ai pas fait la commande de composant. Je n'ai que utilisé les composants qui sont déjà disponibles à la salle d'étude et réalisation. Pourtant, j'ai vérifié le coût pour chaque composant utilisé, et en totale il coûte environs 90 Euros. Ce prix est un peu élevé à cause des lampes LED rouges, bleues, vertes et blanches qui sont chères.

Nom	Référence	Empreinte	Valeur	Quantité	Prix Unitaire	Prix total
Résistances	R9, R11, R12, R13, R14, R15	RCO4L	1,5 kΩ	6	0,05 €	0,30 €
	R4, R5	RCO4L	10 kΩ	2	0,05 €	0,10 €
	R6, R8	RCO4L	820 Ω	2	0,05 €	0,10 €
	R1, R2, R7, R10	RCO4L	1 kΩ	4	0,05 €	0,20 €
Condensateurs	C1,C7	RADIAL06	10uF	2	0,45 €	0,90 €
	C2, C8	CK06	100nF	2	0,51 €	1,02 €
	C3, C4	CK06	22pF	2	0,51 €	1,02 €
	C5	RADIAL08	100uF	1	0,45 €	0,45 €
	C6	RADIALO6L	470uF	1	0,45 €	0,45 €
LEDs	D1	LED3	3mm	1	0,09 €	0,09 €
	D4	LED3	2mA	1	0,09 €	0,09 €
Diodes	D2	D041	1N4007	1	1,11 €	1,11 €
	D3	DO41	1N5819	1	2,00 €	2,00 €
Connecteurs	JP2, JP3, JP5, JP6 4	8 broches		4	1,36 €	5,44 €
	JP1	5 broches		1	1,12 €	1,12 €
	JP4	CON ISP		1	2,40 €	2,40 €
	JP7			1	2,30 €	2,30 €
	P1	DB15 femelle		1	2,53 €	2,53 €
Inductances	L1	RADIAL06L	10uH	1	0,73 €	0,73 €
	L2	RADIAL06L	47uH	1	0,73 €	0,73 €
Transistors	Q1, Q2, Q3, Q4, Q5, Q7	TO220	IRL2203N	6	1,73 €	10,38 €
Quartz	Q6	HC18UV	16Mhz	1	0,39 €	0,39 €
Réseau de résistance	R3	09PL1	8 x 4,7k	1	0,28 €	0,28 €
Micro-contrôleur	U1	40DIP600L	ATMega8535	1	5,93 €	5,93 €
Régulateur	U2	08DIP300L	LM2574M	1	1,80 €	1,80 €
Lampe LED Rouge			1W	4	3,50 €	14,00 €
Lampe LED Jaune			1W	4	4,30 €	17,20 €
Lampe LED Blanche			1W	2	6,60 €	13,20 €
Lampe halogène			20W	2	1,37 €	2,74 €
TOTAL				**58**		**89 €**

7 Planning réel et planning prévisionnel



Commentaire sur le planning :

D'après la présentation de planning, on voit bien qu'il y a de retard et décalage entre le planning réel et le planning prévisionnel. Par exemple, au début, j'ai prévu dans le planning prévisionnel juste une séance pour la prise de connaissance du projet précédent. Par contre, dans le planning réel, j'ai pris environs quatre séances pour cette tâche. Je me suis rendu compte que, bien qu'on continue le projet des autres personnes, ce n'est pas toujours évident, surtout au point de vue de la compréhension et de la connaissance du projet.

Ensuite, je me suis concentré sur la tâche de programmation du micro-contrôleur AtMega8535 pendant cinq séances au lieu de quatre séances dans le planning prévisionnel. J'ai fait les tests et les validations en même temps pour vérifier les résultats de mon programme. Pendant chaque séance d'étude et réalisation, j'ai aussi rédigé le rapport afin de ne pas être stressé au dernier moment, et de ne rien oublier les choses essentielles depuis au début de mon projet.

Conclusion

Après plusieurs tentatives, le projet initial qui consistait à programmer le micro-contrôleur AtMega8535 est terminé. Avant cette expérience, je n'étais pas apriori attiré par la programmation. Par contre, à la fin de ce projet, je trouve que c'est très intéressant du fait de son application concrète sur le karting.

Pendant la séance d'étude et réalisation, j'ai dû réaliser ce travail seul ce qui ne n'a pas facilité la situation. Je remercie très sincèrement mon professeur, M. LEQUEU, qui m'a encouragé et suivi dès que j'en avais besoin. Grâce à ce travail, j'ai l'opportunité de découvrir un nouveau logiciel, le Code Vision AVR ainsi que le fonctionnement de micro-contrôleur AtMega8535, ce qui me permettent de prendre le goût et découvrir le plaisir de programmation.

Les avantages de ce projet est on peut commander l'éclairage et la signalisation du karting facilement en utilisant un boîtier de commande. En plus, ce système d'éclairage et de signalisation est identique à une voiture. Grâce à des plusieurs entrées et sorties du micro-contrôleur, je peux réaliser beaucoup d'applications au niveau de programmation. Par exemple, je peux faire varier l'éclairage des certaines lampes en appliquant la fonction MLI et CAN de l'AtMega8535. Par la suite, je peux toujours tester et vérifier le fonctionnement de la carte électronique grâce à une programmation simple.

Pendant les vacances scolaires, j'en ai profité pour réfléchir et créer l'organigramme de programmation. J'ai aussi fait quelques recherches sur internet à propos de composants utilisés dans ce projet afin de comprendre leur fonctionnements. Ainsi, j'ai gagné du temps pour mon projet et je n'ai pas eu la contrainte de temps. Mon projet avançait efficacement et je voudrais remercie encore une fois toutes les personnes qui m'ont beaucoup aidé.

Index des illustrations

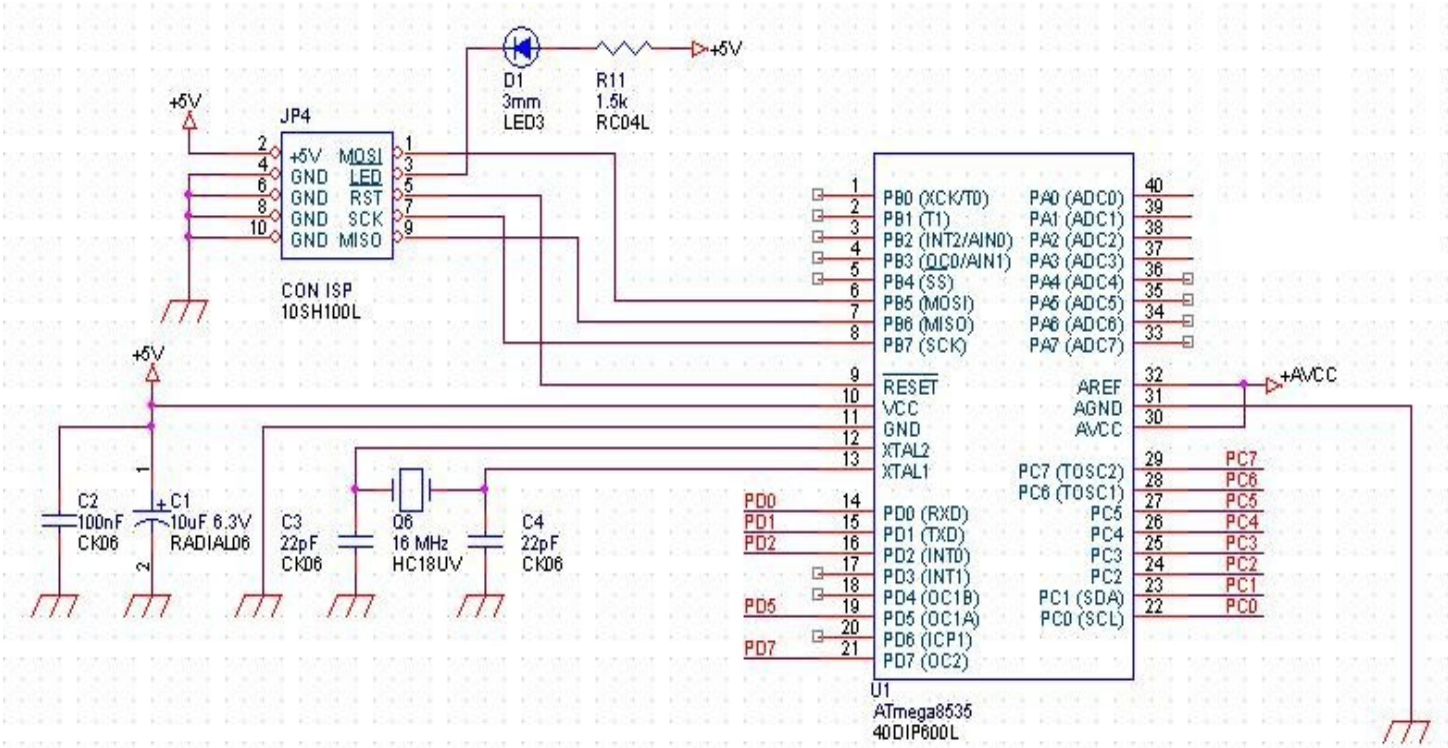
Index des tableaux

Bibliographie

[1] http://www.thierry-lequeu.fr/data/DATA393b.pdf
[2] Schéma personnel : Rafizzi ROSLI, TOURS, 37, FRANCE
[3] http://www.educauto.org/Documents/Tech/ANFA-ECLAIRAGE/Originaux/images/media17.jpg
[4] http://www.thierry-lequeu.fr/data/RAP-RICHARD-BRELIVET.pdf
[5] Photo personnel : Rafizzi ROSLI, TOURS, 37, FRANCE
[6] http://octopart.com/lm2940ct-12-national+semiconductor-9913
[7] http://chinaimportexport.wikispaces.com/
[8] http://www.mcu.hk/GIF/chips.php
[9] http://1wire.fr/dotclear/index.php/2008/02/26/6-realisation-d-un-detecteur-jour-nuit

Les annexes

Annexe 1 : Schéma capture du micro-contrôleur AtMega8535



Annexe 2 : Les lignes de programmation complètes

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Evaluation
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Test01
Version : 1
Date    : 15/02/2007
Author  : Freeware, for evaluation and non-commercial use only
Company : Thierry
Comments:

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

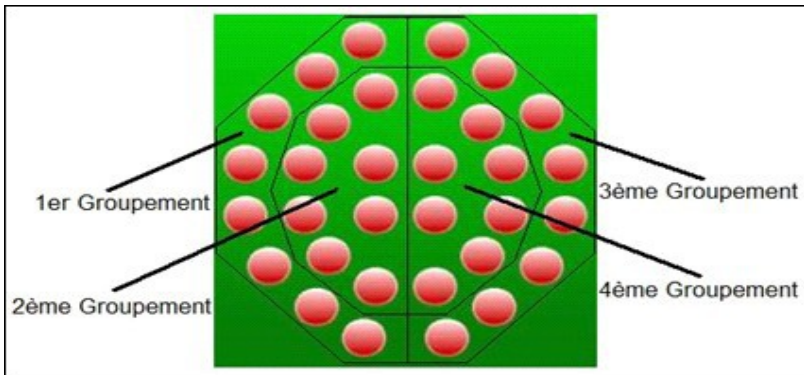
```
#include <mega8535.h>
#define ADC_VREF_TYPE 0x20
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
#include <delay.h>
#define feux_recul PORTC.0
#define eclairage_arriere PORTC.1
#define eclairage_avant PORTB.3
#define clign_g PORTC.6
#define clign_d PORTC.7
#define feux_stop PORTD.7
#define leds_verte PORTC.4
#define leds_bleu PORTC.2

#define BP_clign_g PIND.3
#define BP_clign_d PIND.2
#define BP_Warning PIND.1
#define BP_mode PIND.0
#define BP_feux_position PIND.5
#define BP_feux_croisement PIND.4
#define BP_eclairage PIND.6

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

float Nf,Ne ;

// 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=Out Func6=Out Func5=In Func4=Out Func3=In Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=T State4=0 State3=T State2=0 State1=0 State0=0
PORTC=0x00;
```

```
DDRC=0xD7;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x80;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// 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: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
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: 250,000 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x64;
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: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: On
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used

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

while (1)
    {
    if(BP_eclairage==0)
    {
        if(BP_Warning==0) //si activer l'interrupteur Warning
        {
                clign_d=!clign_d;  //faire clignoter clignotant droite
                clign_g=!clign_g;  //faire clignoter clignotant gauche
                delay_ms(500);     //durée 0,5s
        }
                else   //sinon
        {
                if(BP_clign_g==0) //si activer l'interrupteur clignotant gauche
                {
                        clign_g=!clign_g;
                        delay_ms(500);
                }
                else
                        clign_g=0;

                if(BP_clign_d==0) ///si activer l'interrupteur clignotant droite
                {
                        clign_d=!clign_d;
                        delay_ms(500);
                }
                else
                        clign_d=0;
                }

        if (BP_mode==0) //si activer l'interrupteur mode Manuel
        {
                eclairage_arriere =1;   //allumer feux de position arrière
                leds_verte=1;  //allumer feux de position avant
                        leds_bleu= !BP_feux_position ;
                eclairage_avant= !BP_feux_croisement ;
                }
```

```
                else
                {
                        leds_bleu=0;
                        feux_recul=0;
                        leds_verte=0;
                        eclairage_avant=0;
                        eclairage_arriere =0;
                }
    }
    else
    {
                leds_bleu=0;
                feux_recul=0;
                leds_verte=0;
                eclairage_avant=0;
                eclairage_arriere =0;
                clign_g=0;
                clign_d=0;
    }

        Nf= read_adc(3); // Acquisition de la tension (potentiomètre) entre 0 et 255
        OCR2=(48.5*Nf+12.75); //la pente a=48.5

        Ne=read_adc(2); //pour obtenir la luminosité et l'entrée 2 est branchée sur la photorésistance
        if(Ne>125)      //luminosité minimale pour feux de position
          {
             PORTC.4=0;
             PORTC.2=0;
             OCR0=173;   //OCR0 = le registre qui fera varier du rapport cyclique
             eclairage_arriere=1;
          }
         if(Ne>173)   //luminosité minimale pour feux de croisement
          {
              PORTC.4=0;
              PORTC.2=1; //eclairer sortie LEDs bleues
              OCR0=214;
              eclairage_arriere=1;
          }
         if(Ne>220)   //luminosité minimale pour feux de route
          {
              PORTC.2=1;
              PORTC.4=1; //eclairer sortie LEDs vertes
              OCR0=255;
              eclairage_arriere=1;
          }
         else   //sinon éteindre tous les feux
          {
              PORTC.2=0;
              PORTC.4=0;
              OCR0=0;
              eclairage_arriere=0;
          }

}

}
```

Annexe 3 : Schéma de répartition secteur de LED (pour les feux de stop)



Annexe 4 : Schéma de la carte centrale du système :