Université François-Rabelais de Tours

Institut Universitaire de Technologie de Tours

Département Génie Électrique et Informatique Industrielle

La Borne d'Arrivée pour le projet 50m départ-arrêté

GRANGER Geoffrey
ROUZIES Ywan
2éme Année – Groupe Q2
Promotion 2007/2009

Enseignants:
LEQUEU Thierry
GILKSHON Bernard

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

La Borne d'Arrivée pour le projet 50m départ-arrêté

GRANGER Geoffrey
ROUZIES Ywan
2éme Année – Groupe Q2
Promotion 2007/2009

Enseignants:
LEQUEU Thierry
GILKSHON Bernard

Sommaire

Introduction

Au 4ème semestre, en travaux d'étude et réalisation, nous devons réaliser un projet technologique mettant en œuvre nos connaissances et permettant dans acquérir de nouvelles.

Lors de la 1ére séance d'étude et réalisation, nous nous sommes dirigé vers les bornes de mesure de temps sur 50m départ-arrêté. Étant donné que nous étions 2 binômes à avoir choisi ce projet, on a décidé de diviser le projet. Nos camarades s'occupent de la borne de départ et nous celle de l'arrivée.

Dans ce rapport, nous allons vous présenter notre sujet basé sur la détection de la vitesse d'un karting électrique. Le projet 50m départ-arrêté avait déjà été traité auparavant, seule la réalisation d'un programme pouvant déterminer la vitesse de passage du véhicule devant la borne devait être créé.

Notre projet portera donc sur la mise en œuvre d'un programme.

Puisque le projet est en partie réalisé, nous présenterons d'abord le but de celui-ci. Ensuite nous étudierons les différentes parties du projet et nous expliquerons notre programme. Dans une dernière partie, nous finirons sur le déroulement du travail.

1. *Présentation du projet 50m départ-arrêté*

1.1. Présentation de l'épreuve pour le challenge e-kart

Tout d'abord, il faut savoir que le projet à pour but d'aider au déroulement du challenge e-kart 2009 (http://www.e-kart.fr). Dans ce challenge se déroule plusieurs épreuves dont celle qui permet de comparer les performances en accélération des différents kartings électriques. Cette dernière nommée 50m départ-arrêté et comme le sous-entend l'appellation une épreuve qui s'effectue sur une ligne droite de 50m.

Durant cette compétition, nous devions disposer de 2 commissaires. Le pilote s'arrête sur la ligne de départ et attend le signal du 1er commissaire. Au signal, le pilote démarre et le 2nd commissaire lance le chronomètre. Après 50m, sur la ligne d'arrivée, le 2nd commissaire arrête le chronomètre.

Afin d'éviter de solliciter plusieurs commissaires, la solution était de créer des bornes pouvant mesurer de manière fiable le temps tout en évitant les faux départs.

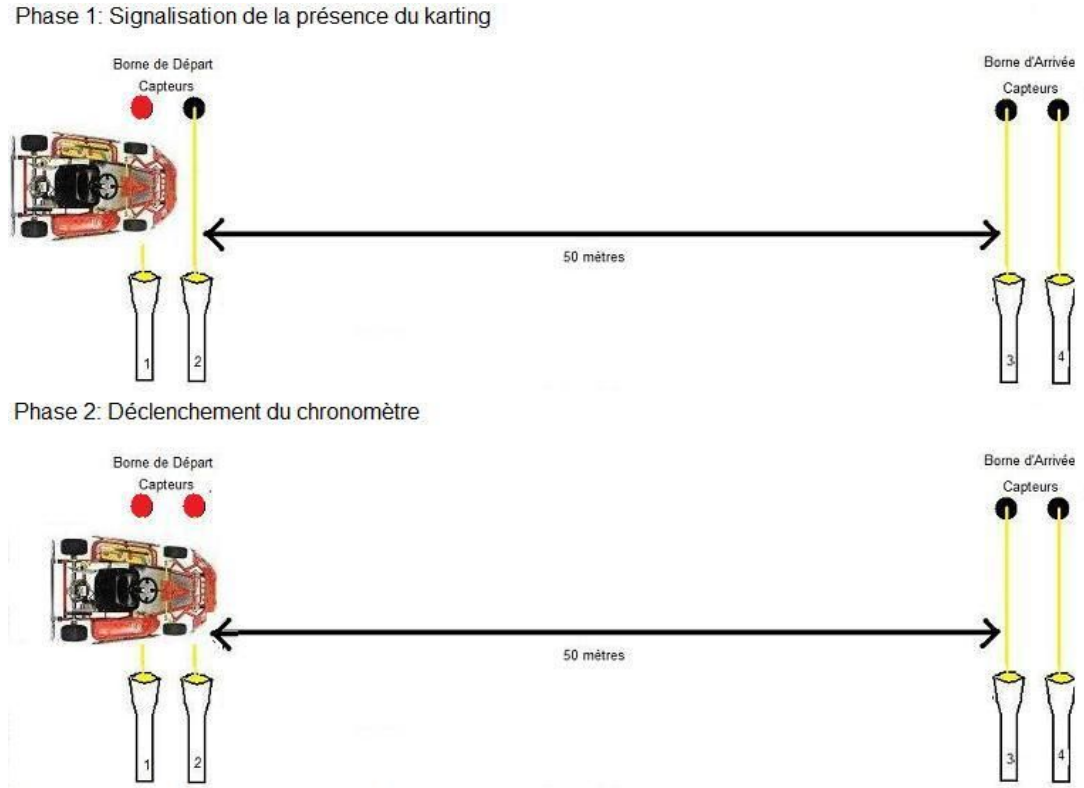


Illustration 1: Déroulement de l'épreuve 50m départ-arrêté (1)

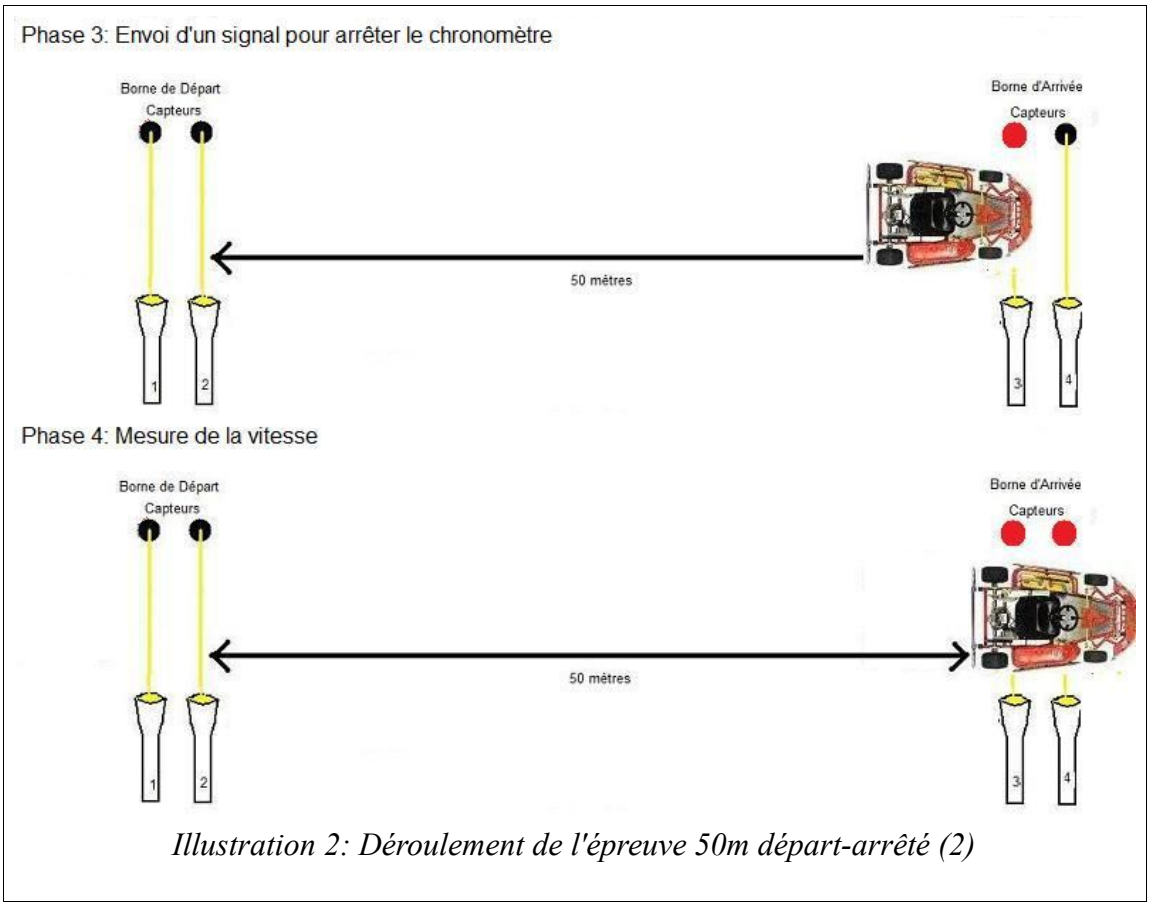


Illustration 2: Déroulement de l'épreuve 50m départ-arrêté (2)

Des lampes de poche émettent des faisceaux lumineux. Lorsque le karting franchit l'un des faisceaux, en face le capteur ne détecte plus la lumière et il le signal grâce à un témoin lumineux.

Au final grâce à ce système, la tâche d'un commissaire serait simplifiée et ceci permettra d'obtenir des résultats d'une plus grande précision.

1.2. Présentation du projet existant

En arrivant sur le projet, nous devons comprendre comment fonctionner ce dernier. Cela fait plusieurs années que le projet est en développement donc nous disposons d'un certain nombre de ressources pouvant nous être utile sur le site : www.thierry-lequeu.fr.

En décomposant le système, on a fini par obtenir différents éléments qui effectuer des fonctions :

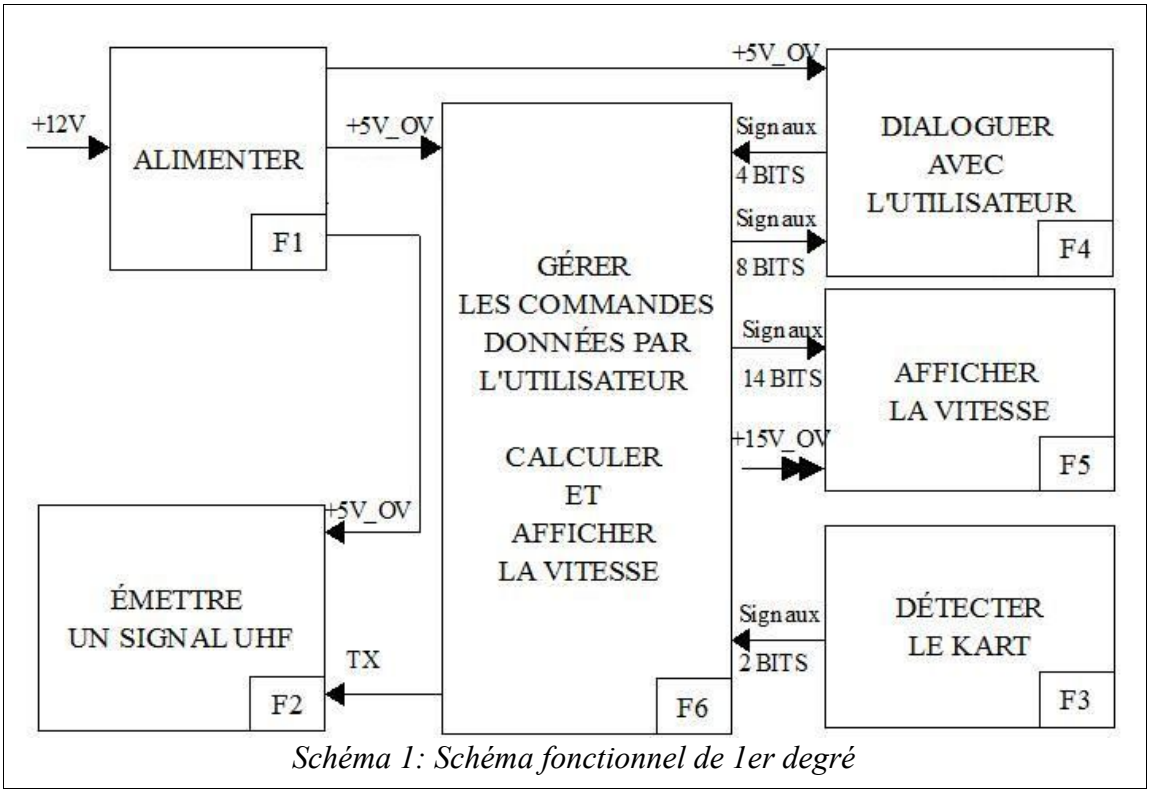


Schéma 1: Schéma fonctionnel de 1er degré

1.3. Cahier des charges

Notre but est de finaliser le projet en intégrant un programme à l'intérieur du microcontrôleur qui utilise l'ensemble des applications possibles pour mesurer la vitesse d'un karting qui passe devant la borne avec une précision à 0,1km/h et d'afficher le score sur un grand afficheur à LEDs.

2. *Étude des fonctions principales*

2.1. F1 : Alimenter

Illustration 3: Alimentation de la borne

L'alimentation de la borne est réalisée avec des régulateurs LM2577-ADJ. Chacun régule une tension en fonction d'un rapport de valeurs de résistances.

Des diodes de protection ont étaient disposées devant chaque sortie pour éviter d'endommager les composants environnants si l'on inverse les fils de branchement. Elle provoque une faible chute de tension et peut bloquer de fort courant inverse.

Des diodes électroluminescentes sont des témoins pour savoir si les 3 alimentations fonctionnent correctement.

Les condensateurs de forte capacité servent de réservoir d'énergie pour éviter les gros écarts de tension. Les autres condensateurs ont le rôle d'éliminer les perturbations hautes fréquences, pour obtenir un signal propre.

Quand nous avons procédé au premier test du projet, on s'est vite aperçu que les ampoules des lampes de poche grillés. Ce phénomène est provoqué par une surtension aux bornes de ampoules. Nous avons donc réaliser une nouvelle alimentation.

Illustration 4: Alimentation des lampes de poche

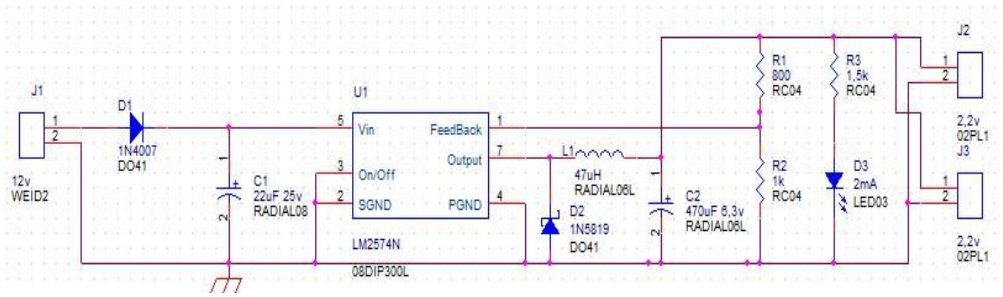


Schéma 2: Alimentation des lampes de poche

Comme précédemment la diode de protection D1 bloque le courant si l'on inverse les fils de branchement. Elle provoque une faible chute de tension mais ne change pas le fonctionnement du montage.

Le bornier JP1 est connecté à la batterie (+12 Volts) .

Le condensateur C1 sert à lisser à filtrer « les perturbations » de la batterie.

Le régulateur à découpage ajustable est le composant U1 (LM2574-ADJ). Une association de deux résistances R1 et R2, nous permet d'obtenir 2,2V en sortie.

$$R2 = R1 * \left(\frac{Vout}{Vref} - 1\right)$$ On prend R2=800Ω

$$R1 = \frac{R2}{\left(\frac{Vout}{Vref} - 1\right)} = \frac{1000}{\left(\frac{2,2}{1,23} - 1\right)}$$ R1=1kΩ dans la série E12

La diode électroluminescente R3 avec la résistance R3 en série, sert à visualiser si le régulateur fonctionne correctement.

2.1. F2 : Émettre un signal UHF

Pour transmettre les informations données par l'utilisateur, on utilise la patte 15 du microcontrôleur ATMEGA 8535. Ce dernier génère sur cette patte, un signal carré positif de 9600 Bauds.

La patte est reliée au composant T7G, ce composant est un émetteur UHF. Ce module à bande étroite de radio UHF fournie une exécution très élevée étant donnée qu'il fonctionne à 433MHz. Le signal a envoyé peut aller jusqu'à un débit de 20 Bauds, ce qui rentre dans nos normes.

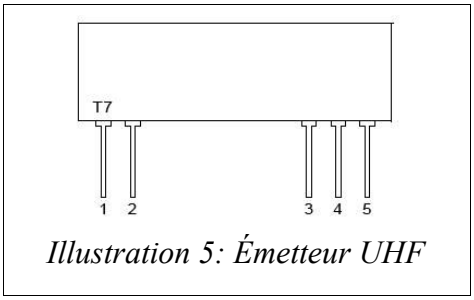


Illustration 5: Émetteur UHF

La patte 1 doit être à la masse. Elle reliée à l'intérieur du boitier à la patte 4 qui elle aussi est reliée à la référence.

Le branchement de l'antenne doit se faire sur la patte 2. La piste la reliant doit être la plus courte possible.

On alimente le composant sur la patte 3. L'alimentation est de 5V pour une puissance de 25mW.

La patte 5 est celle qui reçoit les données à émettre. Cette entrée à une impédance de 47 kΩ, ce qui provoque un faible appel de courant. On l'a relie au microcontrôleur par la patte 15 (TXD).

2.2. F3 : Détecter le karting

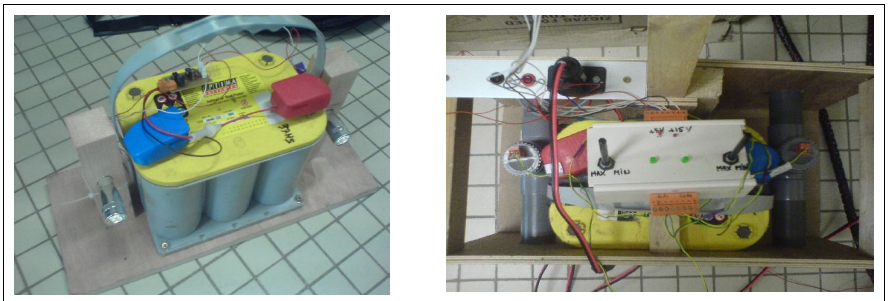


Illustration 6: Système de détection

Le semestre précédant un binôme avait conçu un équipement permettant de détecter la présence du karting devant un capteur.

Le système est en 2 blocs. D'un côté une lampe de poche à focus variable et de l'autre des capteurs photodiodes avec un montage comparateur.

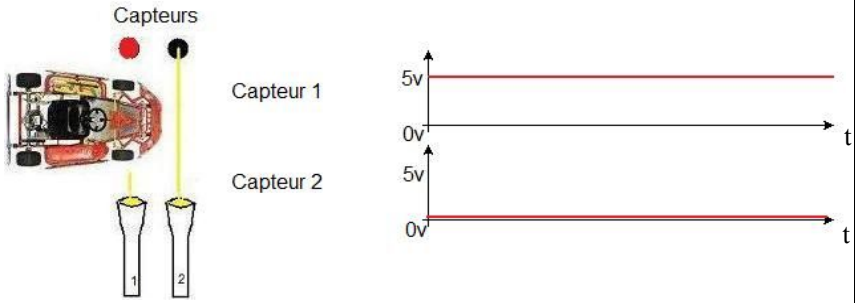


Illustration 7: Détection du karting

Quand le kart s'interpose entre la source de lumière et le capteur, le système nous délivre selon la sortie correspondant au capteur un tension de +5v. Sur le montage, on dispose de potentiomètre pouvant nous aider à régler le seuil de comparaison.

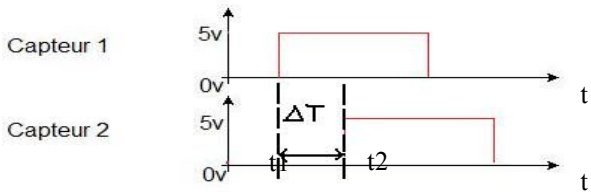


Illustration 8: Passage du karting devant les capteurs

Le premier faisceau est coupé à t_1, le chronomètre est donc lancé à t_1.
Le second faisceau est coupé à t_2, donc le chronométrage s'arrêtera à t_2.

Par exemple on cherche la vitesse, les faisceaux à 30cm l'un de l'autre, et que le temps t_2-t_1 à été mesuré à 12ms, le calcul suivant doit donc être effectué:

$$Vitesse = \frac{3600 * 30.10^{-2}}{1000 * \Delta T} = \frac{3{,}6 * 0{,}3}{12.10^{-3}} = 90\text{km}/h$$

2.3. F4 : Dialoguer avec l'utilisateur

Pour échanger des informations avec l'utilisateur, on dispose en façade de 4 boutons de couleurs différentes ainsi qu'un écran LCD.

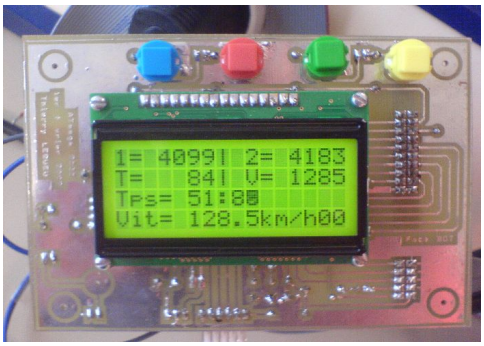


Illustration 9: Face côté composant de la carte microcontrôleur

2.3.1. Fonctionnement des boutons poussoirs

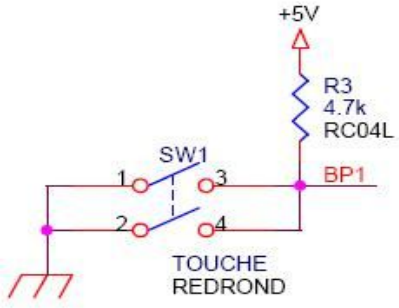


Schéma 3: Schéma structurel d'un bouton poussoir

Les boutons fonctionnent en logique inverse.

C'est à dire que si ils sont au repos le signal de sortie au point BP1 est de 5V.

Dans le cas contraire, lors de l'appuie du bouton, BP1 se trouve à la masse. Enfin on retrouve la différence de potentielle de 5V aux bornes de la résistance R3.

2.3.2. Fonctionnement de l'afficheur LCD

Pour pouvoir visualiser les informations, on utilise un afficheur LCD 4lignes sur 16 caractères.

Il est branché en mode 4 bits. Seuls les 4 bits de poids forts (D4 à D7) de l'afficheur sont utilisés pour transmettre les données et les lire. Les 4 bits de poids faible (D0 à D3) sont dans le vide. On a 7 pattes pour commander l'afficheur. Les données sont écrites ou lues. Une impulsion positive doit être envoyer sur E pour indique que des données sont valides.

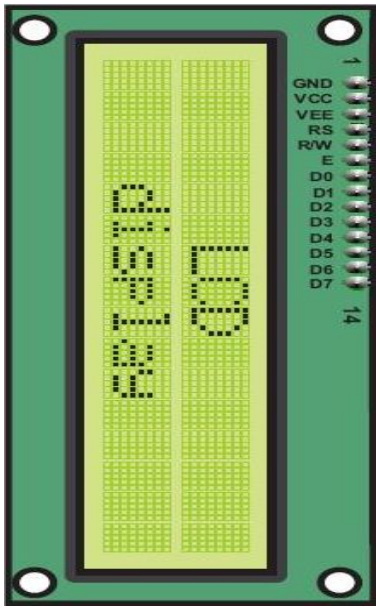


Illustration 10: Afficheur LCD 4x16

Les pattes 15 et 16 (non représentées) sont présentes pour le rétro éclairage.

On commande 7 broches E, R/W, RS, D4, D5, D6 et D7.

On distingue 2 types d'envoi de signaux vers l'afficheur :

- pour exécuter une instruction, la broche RS est mise à la masse et les broches D7 à D0 définissent l'instruction à exécuter, puis E passe à 1,

-pour envoyer des données, il faut mettre la broche RS à 1 et E passe à 1

2.4. F5 : Afficher la vitesse

Pour afficher la vitesse, la borne possède un grand afficheur 7 segments à LED hautes luminosités, ceci permettant de voir de très loin les chiffres affichés.

Illustration 11: Grand afficheur à LED

2.5. F6 : Gérer les commandes données par l'utilisateur et calculer la vitesse

C'est dans cette partie que l'on a consacré l'essentiel de notre temps pour la réalisation du programme.

Dans cette fonction on utilise un microcontrôleur ATMEGA 8535:

2.5.1. Présentation du microcontrôleur:

Le cœur du noyau est basé sur une architecture RISC (Circuit à jeu d'instructions réduit), qui compte 90 à 120 instructions. Les microcontrôleurs de ce type utilisent des instructions codées sur un seul mot. Cela offre des avantages, un gain de place et un gain de vitesse d'exécution. Les circuits RISC utilisent une structure pipe-line qui leur permet d'exécuter une instruction tout en recherchant la suivante en mémoire d'où le gain de vitesse.

Il existe trois grandes familles de microcontrôleur Atmel :

•ATtiny (très petit, possède 8 pattes, 2 Koctets de mémoire)

•AT90 (le plus connue, possède 40 pattes, de 1 à 8 Koctets de mémoire)

•Atmega (le plus gros circuit, 48 pattes d'entrées/sorties parallèles, 128 Koctets)

Le modèle du microcontrôleur utilisé est un AT90S8535. Il possède 8 Koctets de mémoire programme Flash, 512 octets de mémoire Eeprom et RAM, 32 entrées/sorties parallèles, 2 interruptions externes, 3 Times, un CAN.

Il possède des mémoires de type Flash,c'est à dire que la mémoire est programmable et effaçable électriquement (possibilité de faire plus d'environ 1000 cycles d'effacement).

Un Timer est une horloge logique qui permet d'effectuer le comptage du temps, d'événements, de base de temps pour la génération de signaux (comme pour l'émission de trames qui doit être de même vitesse pour récupérer ce dernier sur la carte réceptrice).

Le AT90S8535 est doté de 3 Timers:

•Le Timer 0 : il est composé de :

•un compteur 8 Bits

•un registre de comparaison qui agit sur OC2(PD7)

•Le Timer 1 : il est composé de :

•un compteur 16 Bits

•2 registres de comparaison liées sur 2 sorties (OC1A = PD5

OC1B = PD4)

•Le Timer 2 : il comporte :

•un compteur 8 Bits

•un registre de comparaison qui agit sur OC2(PD7)

•un générateur de PWM

Le mode PWM (Pulse Wave Modulator, en français : MLI, Modulation en Largeur d'Impulsion) sert à générer un train d'onde de fréquence constante, avec un rapport cyclique variable.

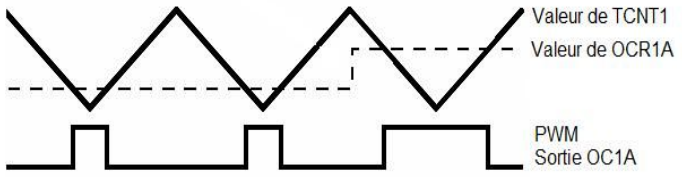


Illustration 12: Fonctionnement du MLI

TCNT1 fixe la fréquence du MLI et OCR1A le rapport cyclique. ON obtient OC1A en patte 19 du microcontrôleur.

Le microcontrôleur est programmable avec un ordinateur via un nappe qui vient du port parallèle. Le reset se fait à la mise sous tension de la carte.

1. *Le programme*

Au départ, notre enseignant nous a fourni un programme qui permettait de compter le temps et de l'envoyer sur le grand afficheur.

Au fur et à mesure, en développant nous sommes arrivé à un programme comprenant un certain nombre de paramètres.

Pour faciliter la compréhension du programme, nous l'avons donc décomposé en plusieurs sous programmes. Pour visualiser le programme complet, regardez dans les annexes.

1.1. La base de temps

Le registre du timer :

```
TCCR1A=0x40;//Horloge à une fréquence de 16 Mhz
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00; // Base de temps = 16 MHz, soit  une période de 62.5 ns.
OCR1AL=0xA0; // Interruption quand on arrive à 160fois (0x00A0=160 soit 10 us)
OCR1BH=0x00;
OCR1BL=0x00;
```

En définissant ces paramètres, la routine d'interruption du timer s'effectuera toutes les 10µs.

Le timer :

```
// Timer 1 output compare A interrupt service routine //Exécution toutes les 10us
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
cms++;                          //cms=centième de ms
if(cms>9)                       //dms=dixième de ms
 {cms=0;                        //ms=milliseconde
  dms++;                        //dixms=dizaine de milliseconde
  if(dms>9)                     //centms=centaine de milliseconde
  {dms=0;
    ms++;
    if(ms>9)
   {ms=0;
    dixms++;                    //à chaque passage dans cette fonction, certaines
variables
    if (dixms>9)                //s'incrémentent pour donner la base de temps
    {dixms=0;
     centms++;
if (centms>9)
     {
      centms=0;
      seconde++;
      cap1=0;                   //remise à 0 des capteurs toutes les secondes
      cap2=0;                   //car on détecte que les fronts montants
      tab[0]=71;
      tab[1]=71;                //tab[0]=kart 1   tab[1]=kart 2
      if(var2==1)
      {
      secondep=0;               //secondep permettant un arrêt de la lecture
      var2=2;                   //du niveau des capteurs pendant 4S
      }
      if(var2==2)
      {
      secondep++;
      }
      if(secondep>3)
      {
      var2=0;
      }
      if (seconde>59)           // à 59secondes remise à 0
      {seconde=0; }
     }
    }
   }
}}}
```

1.2. L'envoi de donnée en série :

Pour générer une trame de données, on lit l'état des capteurs.

Si le kart n'est pas encore passé on écrit G soit 71 en hexadécimal.

Si le kart passe on écrit S soit 83 en héxadécimal.

En fonction de ces valeurs, la borne d'arrivée poura savoir si elle doit arrêter le chronomètre.

Pour envoyer les 2 valeurs (une par karting) , on doit créer un octet de start. Celui-ci permet de nous retrouver dans les correspondances des valeurs de trames en fonction des signaux à générer. On prendra la valeur de l'octet de start à 0x90. Nous devons nous assurez que les valeurs des octets générés par le passage des kartings ne passent par 0x90. Dans aucun cas ceci peu se produire (0X71 ou 0X83).

Les variables sont alors placées dans un tableau à une dimension et 3 colonnes.

tab[0] correspond au passage du karting n°1

tab[1] correspond au passage du karting n°2

tab[2] correspond à l'octet de start

Voici un exemple de trame envoyée :

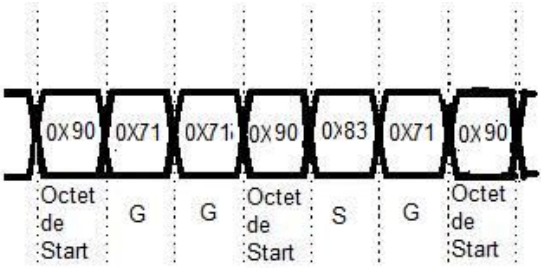


Illustration 13: Trame de données

Pendant la première émission, aucun karting n'est encore passé devant la borne d'arrivée. Dans la suivante, le karting n°1 est arrivé, le chronomètre de la borne de départ peut être arrêté. La vitesse d'envoi sera de 9600 bauds.

```
unsigned char tab[3]={0,0,0x90};

interrupt [13] void usart_txcomplete(void)        //fonction envoi d'une trame en série
{
UDR=tab[k];                                       //UDR est la donnée envoyée
k++;                                              //On incrémente k
if (k==3) k=0;                        //retour à 0 après avoir envoyer toutes les données
}
```

1.3. La mesure du temps de passage

Pour effectuer un calcul précis, nous avons opté pour l'utilisation d'interruptions externes. Celles-ci nous permettent de retenir des variables au moment exacte l'interruption est déclenchée.

Dans les paramètres nous avons définis que les interruptions externes seront activés lors d'un front montant des pattes concernées du microcontrôleur.

Les interruptions externes :

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)   //interruption associée au passage du kart n°1
{
      if((var1==1)&&(var2==0))                 //var2 permet une attente de 4s après le passage
      {
         T1=(centms*1000+dixms*100+ms*10+dms);//mémoristaion du temps dans la variable T1
         secondex=seconde;

       var1=0;                                 //var1 permet de chosir l'ordre des interruptions
      }
      cap1=1;
      tab[0]=83;                               //envoi que le kart n°2 a fini la course
}
```

```
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)   //interruption associée au passage du kart n°1
{
      if((var1==0)&&(var2==0))
      {
         T2=(centms*1000+dixms*100+ms*10+dms);//mémoristaion du temps dans la variable T2

       if(seconde>secondex)                    //ajout de 1s si le kart dépasse 1s entre les deux capteurs
       {
       T2=((seconde-secondex)*10000+centms*1000+dixms*100+ms*10+dms);
       }

       cal_aff_vitesse();                      //lancement du calcul est de l'affichage de la vitesse

       var1=1;
       var2=1;
      }
      cap2=1;
      tab[1]=83;                               //envoi que le kart n°2 a fini la course
}
```

1.4. Le calcul et l'affichage de la vitesse :

```
//Calcul et affiche la vitesse
void cal_aff_vitesse(void)
{
      T=T2-T1;                              //détermination de ΔT
                                            //calcul de la vitesse dans une variable réel
      Vitesse1=18000/(T/(6));                     //formule spéciale pour éviter le dépassement
      Vitesse=(int)Vitesse1;                      //conversion de la valeur pour pouvoir l'afficher

      centainev  = Vitesse/1000;                  //sépartation des unités
      dizainev  = (Vitesse-centainev*1000)/100;
      unitev = (Vitesse-(centainev*1000+dizainev*100))/10;
      centiemev = Vitesse-(centainev*1000+dizainev*100+unitev*10);

         afficheur(0,centainev,0);              //envoi de la vitesse sur le grand afficheur
         afficheur(1,dizainev,0);
         afficheur(2,unitev,1);
         afficheur(3,centiemev,0);
}
```

Le fonction afficheur était déjà créé dans le programme fourni par l'enseignant.

```
// Affiche une valeur sur 8 bits à l'adresse de l'afficheur :
void afficheur(unsigned char adresse,unsigned char caractere, unsigned char point)
{
 if (point == 1)
 {
 PORTA=(valeur_constant[caractere] | 0x01 ); // Un caractère avec le point !
 }

 else
 {
 PORTA=valeur_constant[caractere]; // Un caractère
 }

 PORTB=(0b00010000 | adresse_constant[adresse & 0x0F]); // PB7 PB6 PB5 CS A0 A1 A2 A3
 PORTB.4=1;
 PORTB.4=0;  // CS = 0
 PORTB.4=1;
 PORTA=0x00;
}
```

valeur_constant et adresse_constant sont des fonctions qui retournent une valeur indiquée par sa position dans la fonction. C'est en quelque sorte un tableau de valeurs.

1.5. Le test du grand afficheur :

Nous avons créer une fonction qui permet de tester les afficheurs car parfois certains segments de LEDs ne s'allument plus dûs à des faux contacts. Ces derniers modifient l'interprétation des chiffres que l'on peut avoir.

```
//Test du grand afficheur
void test_aff(void)
{
for(k=0;k<3;k++)
{       for(j=37;j<47;j++)          //j permet de choisir le segment
        {       for(i=0;i<4;i++)   //i permet de choisir l'afficheur
        {
        afficheur(i,j,0);  //test de touts les segments
        delay_ms(50);      //attente de 50ms
        }
        }
}
}
```

1.6. Le menu

Pour que tout le monde puisse utiliser la borne sans avoir à demander à une personne de l'équipe, nous avons donc mis en place un système de menu.

Le menu 3 fenêtres différentes :

- un affichage normal
- un affichage détaillé
- un affichage de test pour le grand afficheur à LEDs

Pour pouvoir créer et afficher un nombre à virgule, on a décidé de multiplier par 10 la vitesse lors de son calcul. Puis lors de son apparition sur l'écran LCD, on calcul la valeur divisée par 10 et le modulo de la vitesse. Ce procédé nous permet d'obtenir 2 nombres entier. Ensuite il suffit donc de jouer sur l'affichage de la vitesse en rajoutant un point entre les deux valeurs calculées.

```
//Affiche les informations sur l'écran LCD
void affichage_lcd(void)
{   if(b1==1)
    {
    sprintf(tampon,"   Tps= %2d:%d%d   ",seconde,centms,dixms);
    lcd_gotoxy(0,0);
    lcd_puts(tampon);

    sprintf(tampon,"   T= %4d ms   ",T/10);
    lcd_gotoxy(0,1);
    lcd_puts(tampon);

    sprintf(tampon," Vit= %4d.%dkm/h",Vitesse/10,Vitesse%10);
    lcd_gotoxy(0,2);
    lcd_puts(tampon);

    lcd_gotoxy(0,3);
    lcd_putsf("4=> Retour Menu ");
    }
 if(b2==1)
    {
    sprintf(tampon,"1=%4d|2=%4d|%d%d",T1/10,T2/10,cap1,cap2);
    lcd_gotoxy(0,0);
    lcd_puts(tampon);

    sprintf(tampon,"T=%4d|Tps=%2d:%d%d",T/10,seconde,centms,c
    lcd_gotoxy(0,1);
    lcd_puts(tampon);

    sprintf(tampon," Vit= %4d.%dkm/h",Vitesse/10,Vitesse%10);
    lcd_gotoxy(0,2);
    lcd_puts(tampon);

    lcd_gotoxy(0,3);
    lcd_putsf("4=> Retour Menu ");
    }
 if(b3==1)
    {
    lcd_gotoxy(0,0);
    lcd_putsf(">Test Grand Aff<");
    lcd_gotoxy(0,1);

    if(k==0)
      {
      lcd_putsf("    EN COURS    ");
      lcd_gotoxy(0,2);
      }
    if((k==3)&&(j==47))
      {
      lcd_putsf("      FINI      ");
      lcd_gotoxy(0,2);
      }
    lcd_putsf("");
    lcd_gotoxy(0,3);
    lcd_putsf("4=> Retour Menu ");
    }
  if(b4==1)
    {
     lcd_gotoxy(0,0);
    lcd_putsf("=====>Menu<=====");
    lcd_gotoxy(0,1);
    lcd_putsf("1=> Aff. normal ");
    lcd_gotoxy(0,2);
    lcd_putsf("2=> Aff. detail ");
    lcd_gotoxy(0,3);
    lcd_putsf("3=> Test Grd Aff");
    }
}
```

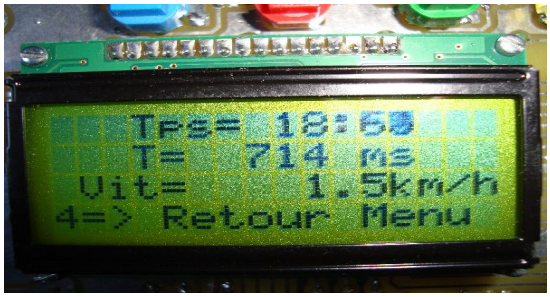


Illustration 14: Affichage normal

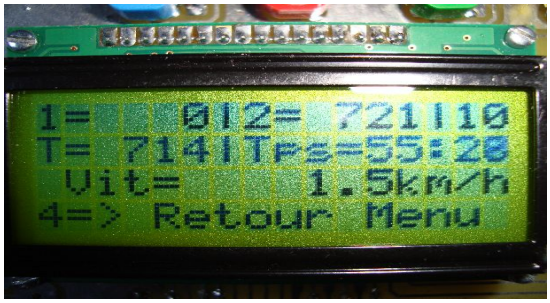


Illustration 15: Affichage détaillé

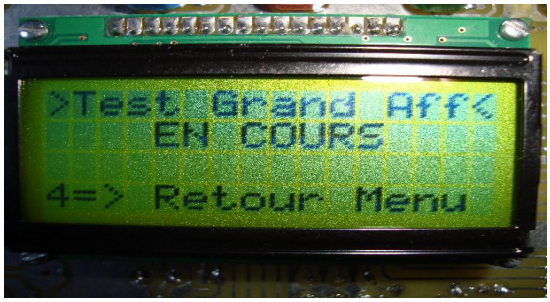


Illustration 16: Affichage du test

Illustration 17: Affichage du menu

1. *Déroulement du travail*

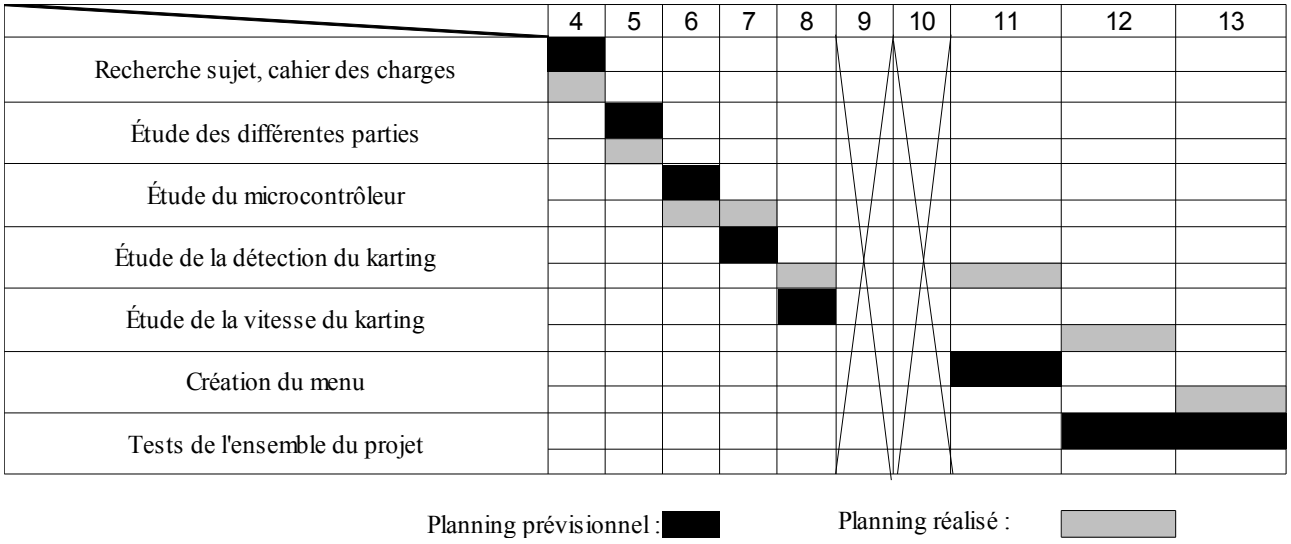
1.1. Planning

Nous avons commencé par comprendre le fonctionnement de l'ensemble du système.

Puis nous avons effectuer une étude pratique du microcontrôleurs en y insérant de petits programmes pour comprendre son fonctionnement.

Ensuite on a réalisé des tests avec un vrai karting.

Pour finir nous avons créé le menu.



1.2. Le typon

On a commencé un typon sur le logiciel Wincircuit. Après plusieurs tentatives, je décide de changer de logiciel à cause des messages d'erreurs imprévisibles qui ferment totalement le logiciel sans sauvegarde.

L'enseignant nous consigne Orcad qui est plus fiable.

Nous commençons par faire toute le schéma structurel, puis le typon sous Orcad Layout.

1.3. Le test des circuits imprimés

Pour le test, on utilise un ohmmètre pour vérifier la continuité des pistes et l'absence de court-circuit.

1.4. Le câblage

On a commencé par souder les composants avec les borniers.

Ensuite on a vérifié que l'alimentation était bien connecté à tous les borniers et que la LED en sortie s'éclairer correctement.

1.5. La programmation

On a commencé à apprendre a se servir du microcontrôleur avec le langage C sur une carte test que nous a délivré le professeur. Notre premier programme avait pour but d'afficher un temps mesuré sur l'écran LCD. Le deuxième devait fonctionner avec le grand afficheur à LEDs. On continua en calculant la vitesse. Enfin en prenant l'ensemble de ces programmes, on a réalisé le programme final avec un menu.

1.6. Le test du projet

Nous avons effectuer un test dehors en pleine journée et on a pu remarqué qu'a la lumière du soleil, la mesure de la vitesse était faussée. Il faudrait trouver un autre système de détection, soit par infrarouge ou avec des lentilles polarisées pour filtrer les rayons lumineux.

1.7. Coût du projet

Nom	référence	Constructeur	Quantité	Prix (unitaire)	Prix total
Régulateur	LM2574_ADJ		4	1,57 €	6,28 €
R1,R2,R3	1,5 k Ω		12	0,02 €	0,24 €
D1	3mm rouge		4	0,01 €	0,04 €
C1,C4	10 µF		8	0,90 €	7,20 €
C2,C3	100 nF		8	0,15 €	1,20 €
Lampe de poche	mini	MagLite	2	21,00 €	42,00 €

Total :	56,96 €

Conclusion

Nous avons pratiquement mené le projet à bien. En effet, nous avons étudié le programme principal. Mais nous n'avons pas eu assez de temps pour réaliser un programme plus simple. En ce moment même, notre projet fonctionne parfaitement. Pour parfaire le projet, on devrez regroupé les programmes des bornes de départ et d'arrivée en un seul et même programme.

En ce qui concerne le projet en lui même, nous avons pris plaisir à le concevoir. Ce fût une réalisation instructive qui faisait appel à une véritable polyvalence. Tout d'abord, nous avons pu exploiter et matérialiser nos compétences acquis au cours de nos options en informatique. De plus, ce projet nous a fait gagner en autonomie. Enfin nous avons mis en œuvre des méthodes de travail et qui nous ont permis d'accomplir un véritable projet.

Index des illustrations

Index des schémas

Bibliographie

Les sources internet:

Document sur Internet	[1] **Atmel Corporation.** *Atmega 8535*, 2006 <*www.atmel.com/dyn/resources/prod_documents/doc2502.pdf*>
Document sur Internet	[2] **R F Solutions Ltd.** *.T7G / R7G*, octobre 2005 <http://www.rfsolutions.co.uk/acatalog/DS307-7.pdf>
Document sur Internet	[3] **T. LEQUEU**.*La documentation de Thierry ,2003* <*http://www.thierry-lequeu.fr/*>

Livre	[3] **JC. Chauveau, G Chevalier et B. Chevalier**.*Mémotech électronique ,2003* *A. Capliez Educalivre, 509 p. , ISBN : 2-7135-1755-9.*

Annexes

A.1. Typon de la carte d'alimentation pour les lampes de poche

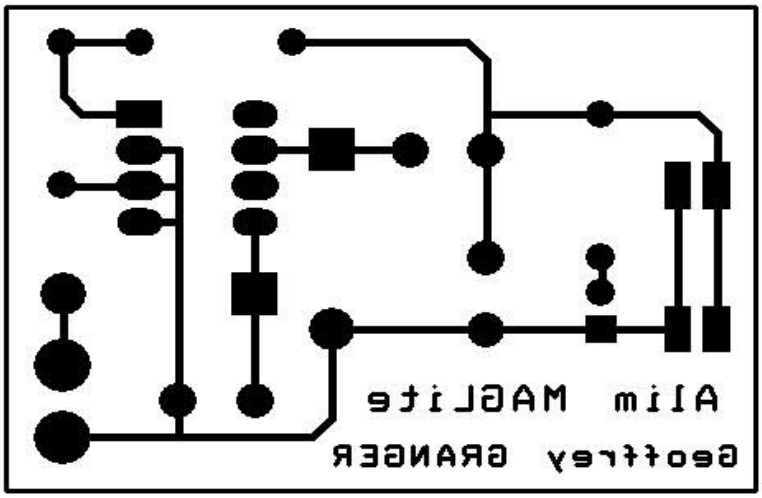
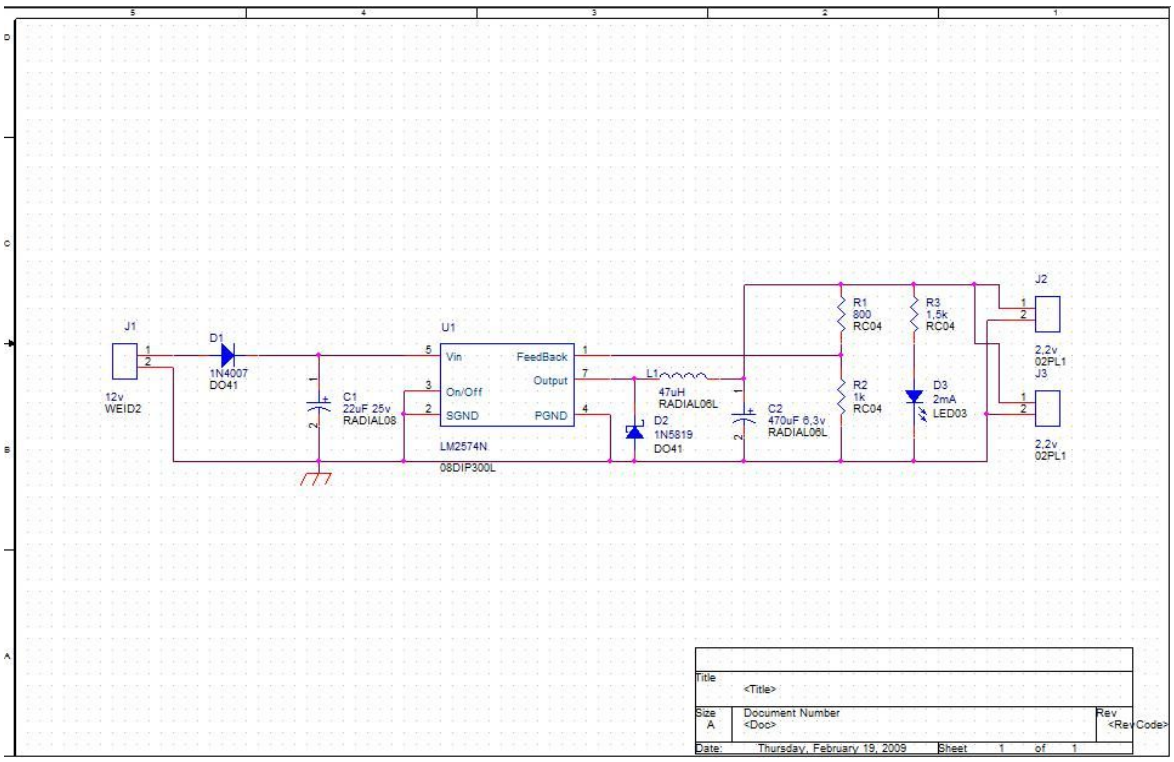


Illustration 18: Typon alimentation MagLite

A.2. Schéma structurel



A.3. Le programme

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project : Borne Départ Arrêter sur 50 m.
Version : 1
Date    : 24/03/2009
Author  : Geoffrey GRANGER
Company : IUT Tours
Comments: Département GEII


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>
#include <stdio.h>

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

#define BP4 PINB.6
#define BP3 PINB.7
#define BP2 PIND.6
#define BP1 PIND.0
#define Capteur1 PIND.2
#define Capteur2 PIND.3
#define ENABLE PORTD.7


//tableau adressage         SS
flash const unsigned char adresse_constant[16] =
{0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15};                //tableau de croisement des bits d'adresse
```

```
//tableau lettrage
flash const unsigned char valeur_constant[] =
{
 0xEE,0x28,0x76,0x7C,0xB8,0xDC,0xDE,0x68,0xFE,0xFC,  //0123456789
  0xFA,0x9E,0x16,0x3E,0xD6,0xD2,0xCE,0xBA,0x82,0x2E, //abcdefghij
  0x9A,0x86,0x1A,0x1A,0xEE,0xF2,0xF8,0x12,0xDC,0x96, //klmnopqrst
  0xAE,0x0E,0xB6,0xBA,0xB2,0x76,0x04,              //uvwxyz_
  0,1,2,4,8,16,32,64,128,256};                    //37->46


// Variables globales
unsigned char
tampon[20],seconde=0,secondex=0,secondep=0,centms=0,dixms=0,ms=0,dms=0,cms=0;
unsigned char tab[3]={0,0,0x90};
int cap1=0,cap2=0,var1=1,var2=0;
unsigned int centainev=0,dizainev=0,unitev=0,centiemev=0;
unsigned int Vitesse=0;
float Vitesse1=0;
unsigned int T1=0,T2=0,T=0;
int i,j,k;
int b1=0,b2=0,b3=0
,b4=0;

// Fonctions

// Affiche une valeur sur 8 bits à l'adresse de l'afficheur :
void afficheur(unsigned char adresse,unsigned char caractere, unsigned char point)
{
  if (point == 1)
  {
  PORTA=(valeur_constant[caractere] | 0x01 ); // Un caractère avec le point !
  }

  else
  {
  PORTA=valeur_constant[caractere]; // Un caractère
  }

  PORTB=(0b00010000 | adresse_constant[adresse & 0x0F]); // PB7 PB6 PB5 CS A0 A1 A2 A3
  PORTB.4=1;
  PORTB.4=0;  // CS = 0
  PORTB.4=1;
  PORTA=0x00;
}

//Test du grand afficheur
void test_aff(void)
{
for(k=0;k<3;k++)
{         for(j=37;j<47;j++)
```

```
        {       for(i=0;i<4;i++)
                {
                afficheur(i,j,0);
                delay_ms(50);
                }
        }
}
}

//Test de  boutons
void test_bouton(void)
{
      if(BP1==0)
      {
      b2=b3=b4=0;
      lcd_clear();
      b1=1;
      }
      if(BP2==0)
      {
      b1=b3=b4=0;
      lcd_clear();
      b2=1;
      }
      if(BP3==0)
      {
      b1=b2=b4=0;
      lcd_clear();
      b3=1;
      }
      if(BP4==0)
      {
      b1=b2=b3=0;
      lcd_clear();
      b4=1;
      }
}

//Affiche les informations sur l'écran LCD
void affichage_lcd(void)
{
   if(b1==1)
     {

     sprintf(tampon,"   Tps= %2d:%d%d   ",seconde,centms,dixms);
     lcd_gotoxy(0,0);
     lcd_puts(tampon);

     sprintf(tampon,"   T= %4d ms   ",T/10);
     lcd_gotoxy(0,1);
```

```
  lcd_puts(tampon);

  sprintf(tampon," Vit= %4d.%dkm/h",Vitesse/10,Vitesse%10);
  lcd_gotoxy(0,2);
  lcd_puts(tampon);

  lcd_gotoxy(0,3);
  lcd_putsf("4=> Retour Menu ");
  }

if(b2==1)
  {

  sprintf(tampon,"1=%4d|2=%4d|%d%d",T1/10,T2/10,cap1,cap2);
  lcd_gotoxy(0,0);
  lcd_puts(tampon);

  sprintf(tampon,"T=%4d|Tps=%2d:%d%d",T/10,seconde,centms,dixms);
  lcd_gotoxy(0,1);
  lcd_puts(tampon);

  sprintf(tampon," Vit= %4d.%dkm/h",Vitesse/10,Vitesse%10);
  lcd_gotoxy(0,2);
  lcd_puts(tampon);

  lcd_gotoxy(0,3);
  lcd_putsf("4=> Retour Menu ");
  }

if(b3==1)
  {

  lcd_gotoxy(0,0);
  lcd_putsf(">Test Grand Aff<");
  lcd_gotoxy(0,1);

  if(k==0)
    {
    lcd_putsf("    EN COURS    ");
    lcd_gotoxy(0,2);
    }
  if((k==3)&&(j==47))
    {
    lcd_putsf("      FINI      ");
    lcd_gotoxy(0,2);
    }

  lcd_putsf("");
  lcd_gotoxy(0,3);
  lcd_putsf("4=> Retour Menu ");
```

```
     test_aff();
     }

    if(b4==1)
     {

     lcd_gotoxy(0,0);
     lcd_putsf("=====>Menu<=====");
     lcd_gotoxy(0,1);
     lcd_putsf("1=> Aff. normal ");
     lcd_gotoxy(0,2);
     lcd_putsf("2=> Aff. detail ");
     lcd_gotoxy(0,3);
     lcd_putsf("3=> Test Grd Aff");
     }
}

//Calcul et affiche la vitesse
void cal_aff_vitesse(void)
{
      T=T2-T1;

      Vitesse1=18000/(T/(6));
      Vitesse=(int)Vitesse1;

      centainev  = Vitesse/1000;
      dizainev  = (Vitesse-centainev*1000)/100;
      unitev = (Vitesse-(centainev*1000+dizainev*100))/10;
      centiemev = Vitesse-(centainev*1000+dizainev*100+unitev*10);

      afficheur(0,centainev,0);
         afficheur(1,dizainev,0);
         afficheur(2,unitev,1);
         afficheur(3,centiemev,0);
}

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
       if((var1==1)&&(var2==0))
       {
          T1=(centms*1000+dixms*100+ms*10+dms);
          secondex=seconde;

        var1=0;
       }
       cap1=1;
       tab[0]=83;
}
```

```
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
        if((var1==0)&&(var2==0))
        {
            T2=(centms*1000+dixms*100+ms*10+dms);

         if(seconde>secondex)
         {
         T2=((seconde-secondex)*10000+centms*1000+dixms*100+ms*10+dms);
         }

         cal_aff_vitesse();

         var1=1;
         var2=1;
        }
        cap2=1;
        tab[1]=83;
}

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
cms++;
if(cms>9)
 {
  cms=0;
  dms++;
  if(dms>9)
  {
   dms=0;
   ms++;
   if(ms>9)
   {
    ms=0;
    dixms++;
    if (dixms>9)
    {
     dixms=0;
     centms++;
     if (centms>9)
     {
      centms=0;
      seconde++;
      cap1=0;
      cap2=0;
      tab[0]=71;
      tab[1]=71;
      if(var2==1)
```

```
      {
      secondep=0;
      var2=2;
      }
      if(var2==2)
      {
      secondep++;
      }
      if(secondep>3)
      {
      var2=0;
      }
      if (seconde>59)
      {
       seconde=0;
      }
     }
    }
   }
  }
 }
}

interrupt [13] void usart_txcomplete(void)                    //fonction interruption avec usart
{
UDR=tab[k];                                                   //UDR est la donnée envoyée
k++;                                                          //On incrémente k
if (k==3) k=0;                                                //retour à 0 après avoir envoyer toutes les
données
}


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

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

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;

// Port C initialization
```

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

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=0 State3=T State2=T State1=T State0=T
PORTD=0x00; // OC1A BP2  IN   IN  Cap2 Cap1 IN  BP1
DDRD=0x80;  // OUT7 IN6 OUT5 OUT4  IN3 IN2  IN1 IN

ENABLE=1;


// 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: 16000,000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// 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=0x40;
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00; // Base de 0temps = 16 MHz, soit 62.5 ns.
OCR1AL=0xA0; // Interruption quand on arrive à 160 fois (0x00A0 soit 10 us)
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
```

```
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: On
// INT1 Mode: Rising Edge
// INT2: Off
GICR|=0xC0;
MCUCR=0x0F;
MCUCSR=0x00;
GIFR=0xC0;


// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;

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

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x10;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// LCD module initialization
lcd_init(16);

/* switch to writing in Display RAM */

//Initialisation

afficheur(0,36,0); // Un caractère
afficheur(1,16,0); // Un caractère
afficheur(2,0,0); // Un caractère
afficheur(3,36,0); // Un caractère
```

```
lcd_gotoxy(0,0);
lcd_putsf("Projet Etude&Rea");
lcd_gotoxy(0,1);
lcd_putsf("__Borne 50m DA__");
lcd_gotoxy(0,2);
lcd_putsf("Borne  d'Arrivee");
lcd_gotoxy(0,3);
lcd_putsf("Rouzies__Granger");
delay_ms(2000);
lcd_clear();

lcd_gotoxy(0,0);
lcd_putsf("=====>Menu<=====");
lcd_gotoxy(0,1);
lcd_putsf("1=> Aff. normal ");
lcd_gotoxy(0,2);
lcd_putsf("2=> Aff. detail ");
lcd_gotoxy(0,3);
lcd_putsf("3=> Test Grd Aff");


PORTB=0xFF;

ENABLE=0;

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



while (1)
      {
       // Place your code here
       test_bouton();
       affichage_lcd();
      }
}
```