Université F. Rabelais

Institut Universitaire de Technologie

Génie Électrique et Informatique Industrielle

Chronomètre géant pour compétition de BMX

| EXPRESSION TECHNIQUE| 2014 |

Basile PERCEVAULT

Eddy VINGERDER

Benjamin DUTERTRE

K4B

Thierry LEQUEU

Philippe AUGER

Université F. Rabelais

Institut Universitaire de Technologie

Génie Électrique et Informatique Industrielle

Chronomètre géant pour compétition de vélo BMX

Étude et réalisation d'un chronomètre avec afficheur

Illustration 1: Grille de départ de course BMX

| EXPRESSION TECHNIQUE| 2014 |

Basile **PERCEVAULT**

Eddy **VINGERDER**

Benjamin **DUTERTRE**

K4B

Thierry **LEQUEU**

Philippe **AUGER**

TABLE DES MATIÈRES

INTRODUCTION

Dans le cadre des projets tutorés d'études et réalisations du semestre 4, nous avons choisi de créer un chronomètre à afficheur géant à LED, avec des afficheurs sept segments.

Ce projet nous a été confié par Monsieur Lequeu, professeur de l'IUT de Tours. En effet Monsieur Toussaint Frédéric, professeur au lycée hôtelier d'Amboise, a demandé à Monsieur Lequeu s'il était possible de réaliser ce projet pour son club de BMX, d'Azay-le-Rideau.

Nous avons pu faire une étude préliminaire avec un système que nous a fourni notre professeur, puisqu'il disposait déjà de ce dispositif pour son club de e-kart.

Monsieur Toussaint nous a aussi apporté le capteur, que nous avons commandé pour en avoir un neuf par la suite, qui commandera l'arrêt du chronomètre.

Nous allons donc commencer par vous évoquer le cahier des charges de ce projet, imposé par Monsieur Lequeu et Monsieur Toussaint. Ensuite, nous allons vous détailler tout ce que nous avons eu besoin de faire ou d'étudier pour ce projet, soient la carte de pilotage des afficheurs, les afficheurs, la carte avec le microcontrôleur pour programmer le chronomètre, et la carte d'alimentation. Une dernière partie sera consacrée à la programmation du microcontrôleur, pour envoyer le code du chronomètre dans celui-ci et valider notre projet.

1. CAHIER DES CHARGES

Monsieur Toussaint nous a demandé de réaliser ce chronomètre avec un budget inférieur à environ 600€-650€, car il avait démarché une entreprise spécialisée qui lui facturait le tout pour environ 1100€.

Le projet devra être fini et opérationnel avant le départ en stage, prévu le 17 novembre 2014, afin de le remettre au « client » en main propre, et surtout que nous soyons les seuls à travailler dessus, pour éviter les erreurs possibles de communication entre groupes de projet.

Par souci d'organisation, nous avions mis au point un planning prévisionnel, que nous n'avons malheureusement pas pu tenir, puisqu'au début du projet, il y a eu une incompréhension des besoins du client et du matériel fourni. Cela nous a amené à établir un planning réel qui correspond à ce que nous avons fait au cours de ce projet.

Planning prévisionnel											
Semaine	36	37	38	39	40	41	42	43	44	45	46
Test du système existant											
Création d'un hacheur élévateur											
Rendez-vous avec le client											
Programmation et tests											
Test final en présence du client											
Rédaction du dossier											
Soutenance orale											

Planning réel											
Semaine	36	37	38	39	40	41	42	43	44	45	46
Rendez-vous avec le client											
Tests des afficheurs existants											
Fabrication des nouveaux afficheurs											
Installation des cartes de pilotage											
Réalisation d'une alimentation											
Programmation du chronomètre											
Tests											
Rédaction du dossier											
Soutenance orale											

Illustration 2: Plannings réel et prévisionnel

Monsieur Toussaint a donc besoin d'un système muni de quatre digits de sept segments, affichant un temps au centième de seconde près, pour une meilleure précision lors des compétitions de BMX et lors des entraînements.

Le déclenchement du chronomètre doit se faire selon la grille de départ. C'est à dire que lorsque celle ci se baisse pour laisser partir les concurrents, le chronomètre démarre le comptage, et lors du passage du premier concurrent au premier virage, le chronomètre doit s'arrêter et afficher le temps auquel le coureur est passé. Ceci sera possible grâce à un capteur optique placé dans le virage souhaité par le client.

Une remise à zéro sera programmée de sorte que, quand la grille de départ est remontée, le chronomètre soit réinitialisé, et que l'afficheur affiche un temps nul.

Le système devra fonctionner en étant branché sur le secteur, et avoir les connectiques nécessaires : nous avons donc besoin d'une alimentation réalisant la transformation de la tension réseau de 230V sinusoïdal en 24V continu. Mais notre projet ne demandant pas une si forte tension, une carte d'alimentation partant de 24V continu en entrée à 9V minimum et 15V grand maximum en sortie devra être réalisée.

2. AFFICHEUR 7-SEGMENTS

2.1. Analyse

Chaque segment de cet afficheur est composé de 5 LED rouge de 10 mm de diamètre en série associé à une résistance de 120 Ω, elle aussi en série pour protéger les LED.

Seul le segment contenant le « point » de l'afficheur contient une seule LED, il est donc nécessaire de placer une diode ZENER afin de réguler le courant que cette dernière reçoit et lui assurer un fonctionnement optimal.

Tous les segments sont reliés à la même masse et il suffit de relier un segment à l'alimentation variable pour allumer ce segment. Il faut également savoir que les LED supportent un courant maximal de 20 mA (30 mA dans l'absolu) et il faut donc dimensionner l'alimentation en conséquence.

De plus, derrière chaque afficheur se trouve une carte de pilotage qui reçoit les données envoyées par le microcontrôleur et contrôle l'allumage des segments en conséquence.

2.2. Réalisation

Nous avons ensuite commencé la réalisation des afficheurs à LED. Pour cela, Nous avions à notre disposition le typon de l'afficheur. Nous avons observé comment il avait été conçu et nous en avons déduit que pour plus de confort au perçage et à la soudure, il faudrait élargir les pastilles des LED.

Mais les pattes des LED étant trop proches les une des autres pour pouvoir élargir les pastilles en les laissant sous forme de cercle, nous avons donc décidé de les élargir de façon ovale.

Une fois la carte finie sur le logiciel et imprimé sur papier calque, nous avons pu passer à la réalisation de la carte. Nous avons du prendre une plaque de dimension 20cm de largeur et 30cm de longueur car, un afficheur mesurant 13,3 cm sur 17,3 cm, il nous était possible de graver deux afficheurs sur une même carte.

La première étape de cette réalisation est de superposer le papier calque à la carte vierge et de passer le tout dans l'insoleuse durant 2 minutes et 30 secondes afin d'éliminer la couche photosensible protectrice présente sur le cuivre, sauf sur les pistes que l'on souhaite conserver.

La deuxième étape consiste à plonger la carte dans le révélateur pour accélérer le processus d'élimination de cette couche protectrice.

Voici ce que l'on obtient :

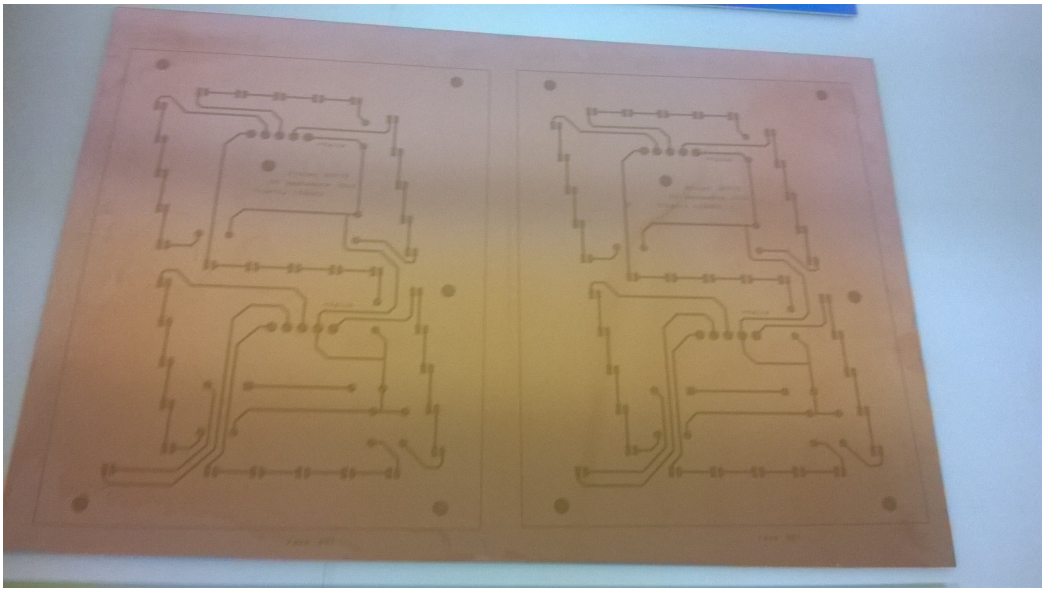

Illustration 3: Cartes d'afficheur insolées

On peut voir sur cette photo que les pistes de la carte apparaissent et que le cuivre présent autour de ces pistes à été affaibli, on peut donc passer à la troisième étape qui consiste à passer la carte dans la graveuse qui permet de retirer le cuivre inutilisé par le biais de perchlorure de fer.

Voici le résultat :

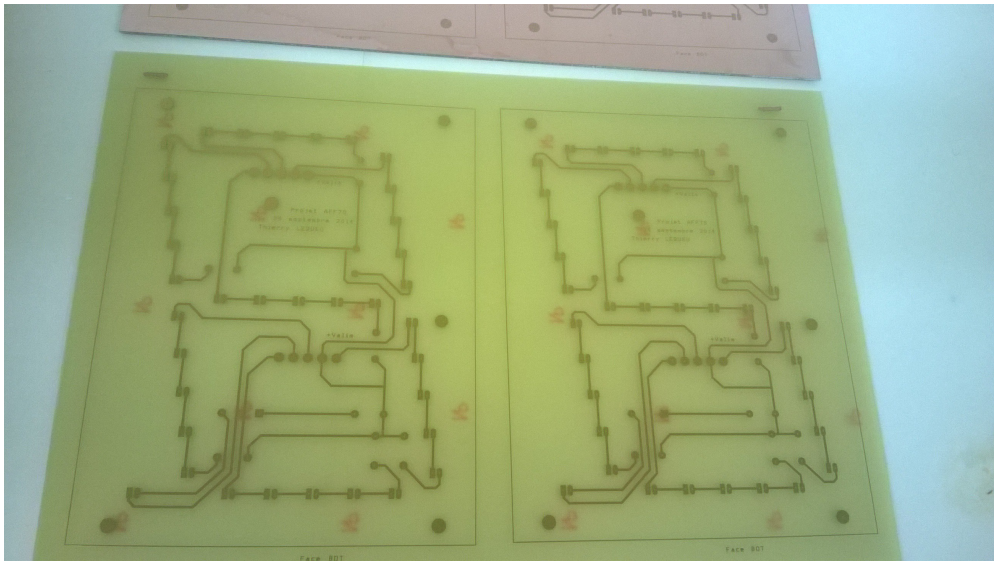

Illustration 4: Cartes d'afficheur gravées

On peux voir sur cette photo que les pistes sont bien présentes et que le cuivre autour de ces pistes à bien disparu grâce à cette dernière étape. On nettoie ensuite la carte avec l'éliminateur qui va aider à enlever la couche protectrice restante sur les pistes de la carte afin de permettre la soudure future des composants.

Une fois le corps de la carte terminé, nous avons dû percer les emplacements des composants. Pour plus de précision et pour avoir un perçage de meilleur qualité, nous avons utilisé des forets en carbure.

Nous avons donc percé les résistances en 0,8mm, les fixations des afficheurs en 4mm , les points d'alimentations en 1,5mm . Nous avons ensuite testé la continuité des pistes de la carte pour s’assurer qu’il n’y avait pas de défauts de continuité ou de fabrication.

Après avoir finalisé les deux premiers afficheurs, nous avons répété cette opération deux autres fois afin d'avoir les quatre afficheurs plus deux afficheurs pour le service après vente en cas d'un quelconque dysfonctionnement .

Une fois toutes les cartes gravés et percées, nous avons coupé le surplus de carte et séparé les afficheurs grâce au massicot.

Suite à cela, nous nous sommes lancés dans la peinture des afficheurs afin que le rendu soit le plus agréable possible (LED rouge sur fond noir). Avec un test préalable, nous nous sommes assurés que la peinture ne boucherait pas les trous de perçage si nous procédions dans cet ordre.

Il y avait différentes sortes de peintures noires en bombes qui s'offraient à nous, nous avons procédé à plusieurs tests pour déterminer la meilleur teinte.

Voici les résultats que nous avons obtenus :

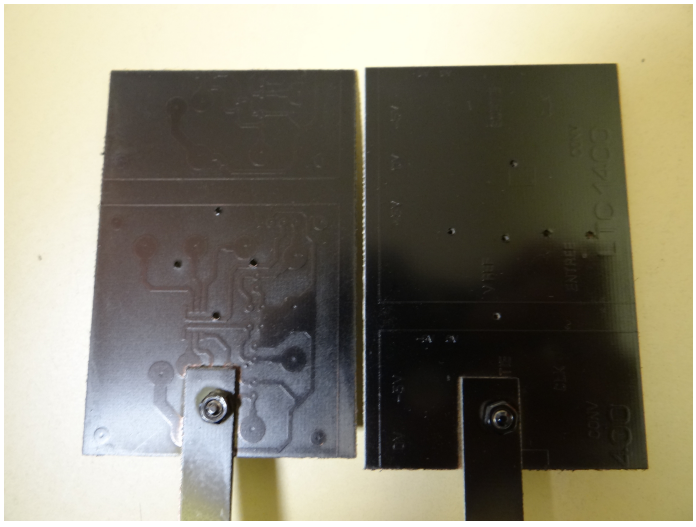

Illustration 5: Peintures mate et brillante

Le premier test nous paraissait trop brillant pour les afficheurs car il refléterait une partie de la lumière ce qui n'est pas souhaitable , nous avons donc opté pour le deuxième test qui lui se révélait bien plus mat et nous avons procédé à la peinture des afficheurs.

Pour cette étape nous nous sommes munis d'un carton pour peindre en extérieurs et ainsi éviter les dépôts de poussières sur la peinture fraîche , nous avons ensuite peint la face des cartes. Nous avons fait deux couches de peinture afin de rendre les cartes uniformes.

Après avoir attendu une journée de séchage, nous avons pu passer à la soudure des composants. Nous avons soudé les résistances en premier car ce sont les plus petits composants donc plus simple à manipuler. Ensuite nous avons soudé les diodes ZENER puis les LED

Une fois terminé, nous avons testé les différents segments en mettant à leurs bornes une tension de 14V pour nous assurer que toutes les LED s'allumaient correctement.

L'étape suivante était de souder les fils (jaune pour chaque alimentation des segments et un rouge pour V+ qui iraient alimenter les segments par le biais de la carte de contrôle des afficheurs

Nous avons ensuite finalisé tous les afficheurs en vernissant le dos des cartes avec du vernis isolant.

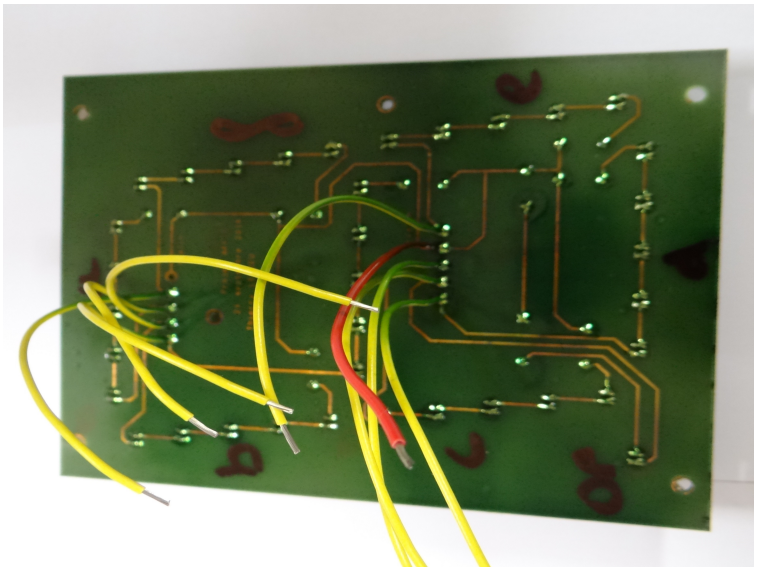

Illustration 6: Carte d'afficheur câblée

Suite à quelques tests, nous nous sommes rendu compte que les diode ZENER de 8,2V ne convenaient pas car le point en bas de chaque afficheur ne s'allumait pas en même temps que les autres LED . La tension de seuil de cette ZENER était trop élevée . On a donc effectué plusieurs tests sur les afficheurs avec une ZENER 8,2V, une 7,5V et une 6,8V.

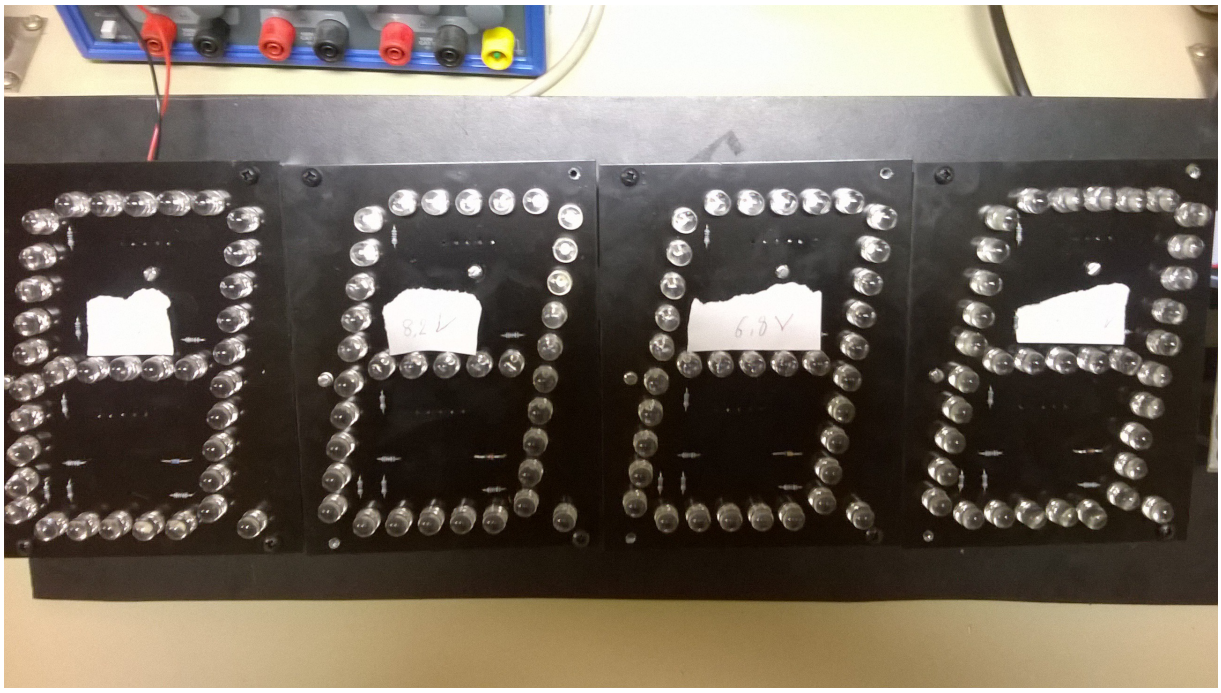

Illustration 7: Test des différentes diodes ZENER

Les tests ont révélé que la meilleure solution était la diode ZENER de 7,5V car le point s'allume à peu près en même temps que les autres LED.

3. CARTE DU MICROCONTRÔLEUR

3.1. Analyse

Le circuit imprimé sur lequel est installé l'ATMEGA 8535 possède beaucoup de composants différents, notamment des connecteurs ISP et RS 232, un port parallèle pour la programmation, un bornier pour l'alimentation ainsi que des boutons poussoirs.

De l'autre côté de la carte de trouve un afficheur LCD de 4 lignes et pouvant afficher jusqu'à 16 caractères par ligne. Nous afficherons donc la valeur du chronomètre sur cet afficheur afin d'avoir un afficheur de secours en attendant le remplacement des digits hors service par les afficheurs de secours que nous avons réalisé.

Cette carte doit donc permettre au microcontrôleur de communiquer avec les afficheurs 7-segments par l'intermédiaire d'une « nappe » connectée sur les cartes de pilotage de chaque afficheur pour envoyer les données nécessaires à l'allumage des différents digits suivant la valeur du chronomètre.

Nous avons également branché sur le connecteur DATA un simulateur des capteurs de la grille de départ et du capteur optique, symbolisés respectivement par un interrupteur et un bouton poussoir, pour simuler la carte sur le terrain de BMX.

3.2. Réalisation

La carte comportant le microcontrôleur ATMEGA 8535 de chez Atmel nous a été fournie par Monsieur Lequeu directement, cette dernière ayant été fabriquée par un sous-traitant. La liste des composants nécessaires à sa fabrication ainsi que leur prix sont disponible en Annexe.

Le fichier PDF contenant la netlist, le typon du circuit imprimé ainsi que l'implantation des composants sur celui-ci est disponible dans la bibliographie. Il est donc possible, dans le cas où la carte du microcontrôleur venait à tomber en panne, d'en refabriquer une 2eme en utilisant cette documentation.

4. CARTE DE PILOTAGE DE L'AFFICHEUR

4.1. Analyse

La carte de pilotage de l'afficheur doit, lorsque le microcontrôleur lui envoie l'ordre d'allumer un digit comme le chiffre 8, par exemple, garder en mémoire cet ordre et « fixer » l'afficheur sur le chiffre 8 : c'est-à-dire que, même si l'ordre d'allumage provenant de l'ATMEGA 8535 n'est plus reçu, le digit doit rester allumé jusqu'à nouvel ordre. Ceci sera expliqué dans la partie Programmation de ce dossier.

La carte de pilotage doit donc, à minima, être équipée d'un connecteur RS232 pour communiquer avec l'ATMEGA 8535 ainsi qu'un driver, dans notre cas le ULN2803 qui contrôle les sorties, c'est-à-dire l'afficheur en lui-même.

4.2. Réalisation

Nous n'avons pas réalisé nous-mêmes les cartes de pilotage des afficheurs, en effet celles-ci nous ont été fournies directement par Mr. Lequeu qui avait fait sous-traiter leur fabrication.

Cependant, sur le site de Mr. Lequeu, il est possible de retrouver les fichiers utilisés pour les faire, comprenant le typon, la liste des composants, leurs désignations, leurs références et leurs valeurs afin de fabriquer une carte identique si le besoin se présente.

Nous avons donc actuellement 4 cartes de pilotage fonctionnelles, ce qui ne nous laisse aucune marge en cas de panne ou de dysfonctionnement de l'une d'entre elles.

5. CARTE D'ALIMENTATION

Pour réaliser cette alimentation, nous allons utiliser un circuit intégré que l'on connaît qu'est le LM2575. Le montage sera un hacheur de type Buck, puisque nous avons en entrée du montage une tension de 24V et que nous souhaitons disposer d'une tension variant entre 9V et 15V en sortie.

Pourquoi ne pas avoir directement acheter une alimentation 15V ?

Parce que notre capteur (OMRON), qui est celui disponible sur la piste de BMX et déjà utilisé par le club, est alimenté entre 10V et 30V continu. Donc on choisit 24V car c'est une valeur assez standard et que ça nous permet de réaliser l'alimentation souhaitée facilement. De plus ça nous permet d'alimenter le capteur et notre système d'affichage.

Et ce que l'on souhaite, c'est donc une alimentation qui, grâce à un potentiomètre variable, aurait une tension variant entre 9V et 15V environ.

Pour ce faire nous avons repris le schéma de base sur le site de Monsieur Lequeu que nous avons modifié pour le rendre plus approprié à notre besoin.

Voici donc le schéma étudié :

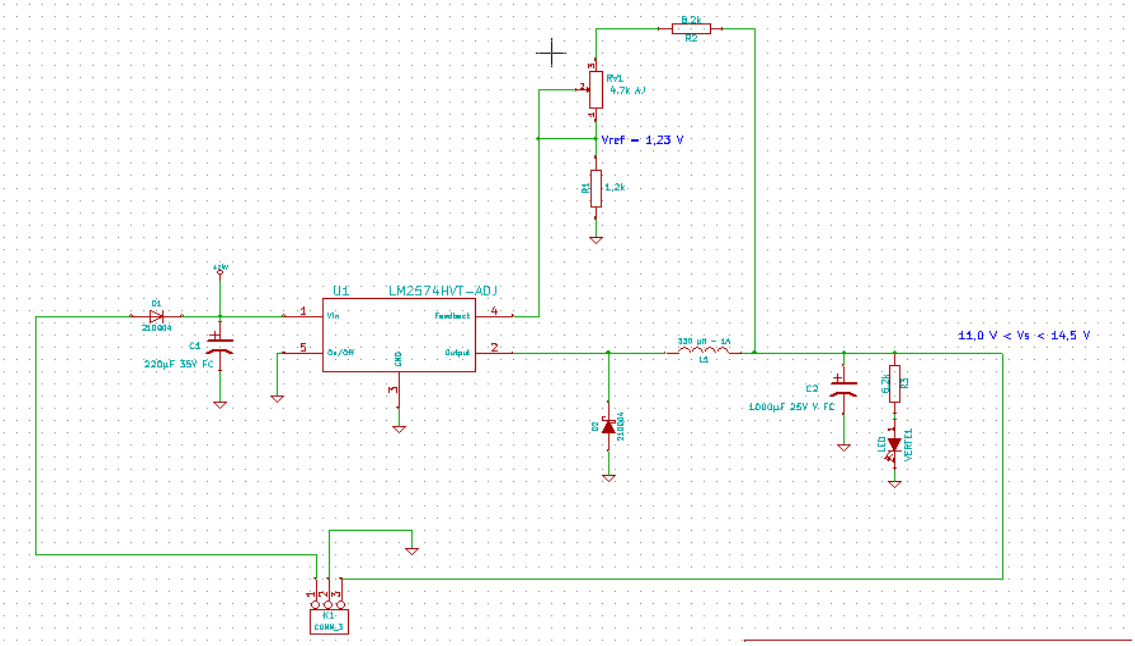

Illustration 8: Schéma de la carte d'alimentation

On garde les mêmes valeurs que le montage d'origine pour les condensateurs de filtrage C1 et C2 ainsi que pour l'inductance L1. La diode ZENER restera la même aussi ainsi que la résistance R1.

En fait, les seules composants que nous devions changer sont le pont de résistance sur le haut du montage, soit la résistance R2 et la résistance variable RV1, qui permettra de faire varier la tension de sortie du montage.

Par un petit calcul de diviseur de tension on obtient la valeur d'une résistance tout en ayant fixé l'autre. En fixant la valeur de RV1 à 4,7kΩ au maximum, pour avoir Vs=11V en théorie, on obtient 8,2kΩ pour R2.

Et quand RV1 est à 0Ω, on obtient Vs environ égal à 14,5V.

Malheureusement ceci est en théorie, puisqu'en pratique, nous n'obtenons pas vraiment les mêmes valeurs. En effet la plage de tension est d'environ : 9V<Vs<15V.

Remarque : Une masse commune est mise en place, de ce fait on peut câbler sur un même bornier 3/1 l'entrée, la masse ainsi que la sortie du montage.

Donc, en conclusion voici la liste des composants finale avec en plus l'affectation des empreintes des composants pour le routage.

```
 1       C1 -     220µF 35V FC : C2V8
 2       C2 -  1000µF 25V V FC : CV14
 3       D1 -           21DQ04 : DO-41
 4       D2 -           21DQ04 : DO-41
 5       K1 -           CONN_3 : bornier31
 6       L1 -      330 µH - 1A : INDUCTANCE
 7       R1 -             1,2k : R5
 8       R2 -             8,2k : R5
 9       R3 -             6,2k : R5
10      RV1 -          4,7k AJ : pot
11       U1 -    LM2574HVT-ADJ : TO220-5B
12   VERTE1 -              LED : LED-3mm
```

Illustration 9: Netlist de la carte d'alimentation

Une fois les bonnes empreintes affectées aux composants, il nous a fallu faire le routage de la carte. Ce qui permet de relier les composants comme indiqué sur le schéma du montage, à l'aide du chevelu qui indique les pattes de composants restantes à relier.

Le routage une fois terminé est comme ci-dessous.

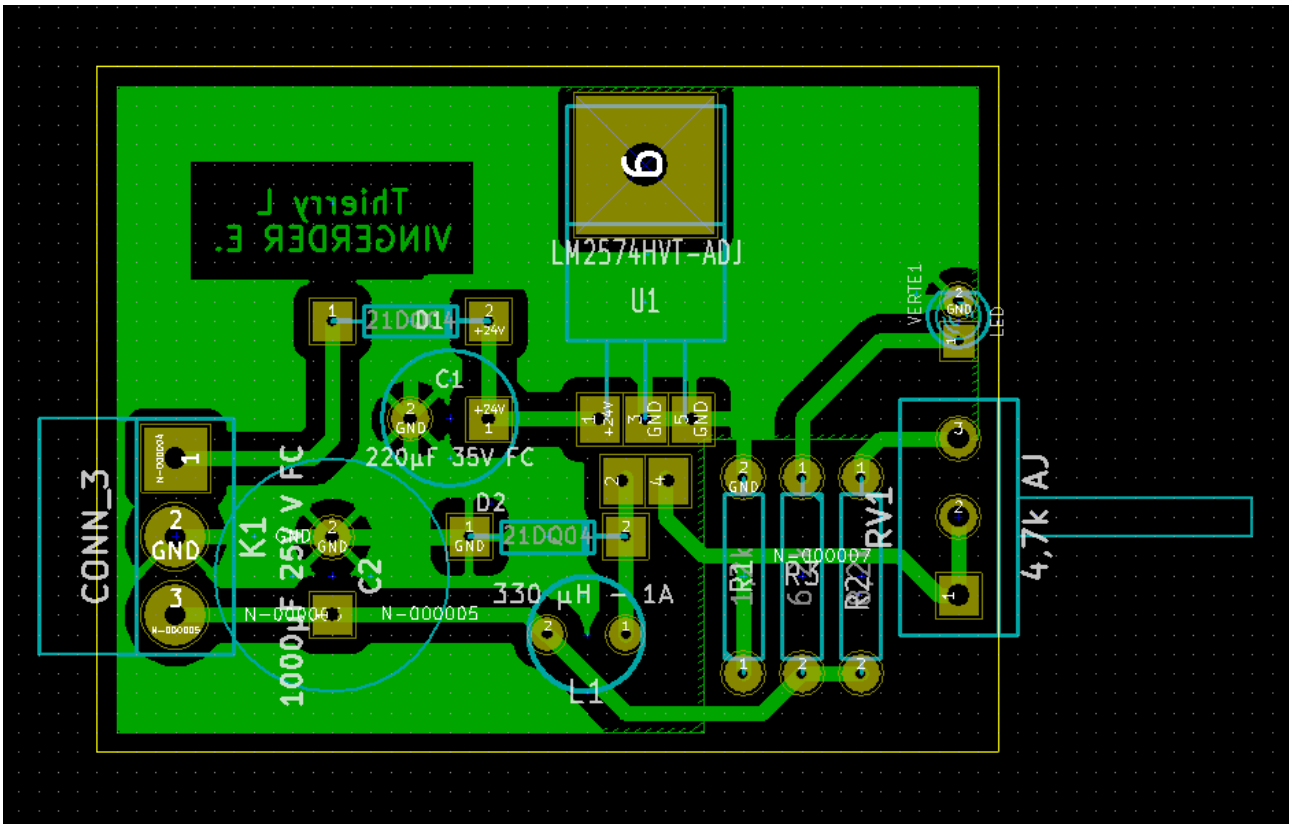

Illustration 10: Routage de la carte d'alimentation

Nous avons réalisé un plan de masse commune sur la carte, car cela évite quelques parasites sur l'alimentation. Voici un aperçu de ce que sera notre carte d'alimentation, grâce à l'outil de visualisation 3D de KiCad :

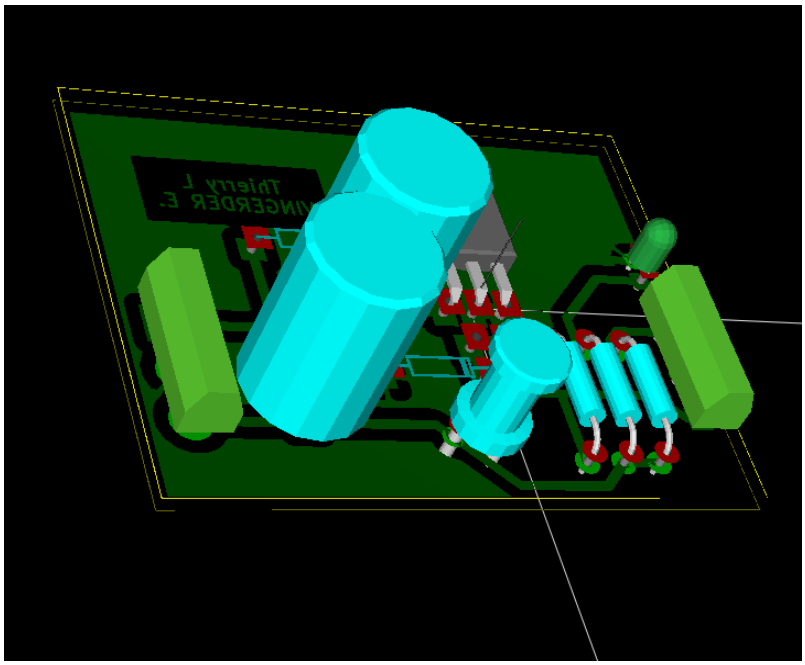

Illustration 11: Vue 3D de la carte d'alimentation

On remarque que visuellement, le circuit intégré LM2575 n'a que 3 pattes de connectées, et non les 5 utilisées dans le montage. C'est dû au fait que nous avons dû refaire certaines empreintes de composants manuellement, car elles n'étaient pas disponibles dans les librairies proposées par l'IUT.

Maintenant, voici ce que nous avons imprimé sur papier calque afin d'insoler notre carte de cuivre et de ne laisser que le cuivre nécessaire comme sur le routage.

Voici une photo du prototype réalisé de la carte d'alimentation.

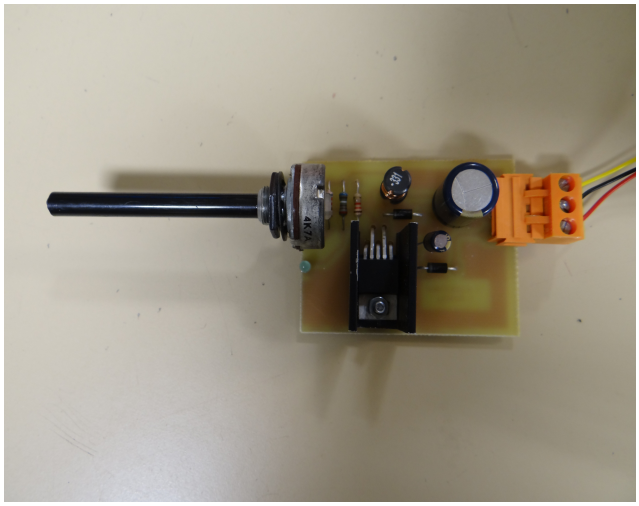

Illustration 12: Carte d'alimentation finie

C'est un prototype car la carte était une carte d'essai. La réalisation de deux autres cartes, une pour le client et une pour service après vente, sera faite en fin de projet.

6. PROGRAMMATION

6.1. Principe

Le chronomètre doit être piloté à l'aide d'un circuit imprimé comportant comme microcontrôleur un ATMEGA 8535. La liste complète des composants de cette carte est disponible dans les annexes. Pour programmer l'ATMEGA 8535 nous utiliserons le logiciel CodeVision AVR dans sa version gratuite.

Voici la liste des actions qui devront être accomplies par cette carte :

- Lancer le chronomètre lorsque la grille de départ descend, actionnant un capteur mécanique dont nous devrons récupérer l'état.

- Arrêter le chronomètre quand le premier participant atteint le premier virage de la course, ce qui sera fait à l'aide d'un capteur optique de la marque OMRON.

- Remettre à zéro la valeur du chronomètre lorsque la grille de départ sera relevée.

Bien évidemment, pendant toutes ces étapes il faudra également afficher la valeur actuelle du chronomètre au centième de seconde sur un l'afficheur 7-segments à LED.

Etant donné que nous pouvons aussi afficher la plupart des lettres (de façon plus ou moins représentative de la réalité) sur l'afficheur, nous avions pensé afficher le message « Pret » lorsque la grille de départ est levée, au lieu d'afficher un temps de 0 secondes et 0 centièmes, mais il nous faut encore voir ceci avec notre client.

6.2. Affichage

Pour programmer l'affichage du chronomètre, il nous faut tout d'abord associer à un caractère, par exemple le nombre 8, les segments correspondants que nous devrons allumer sur un des digit de l'afficheur 7-segments.

Nous avons, pendant la réalisation des afficheurs, choisi le poids de tel ou tel segment comme suit :

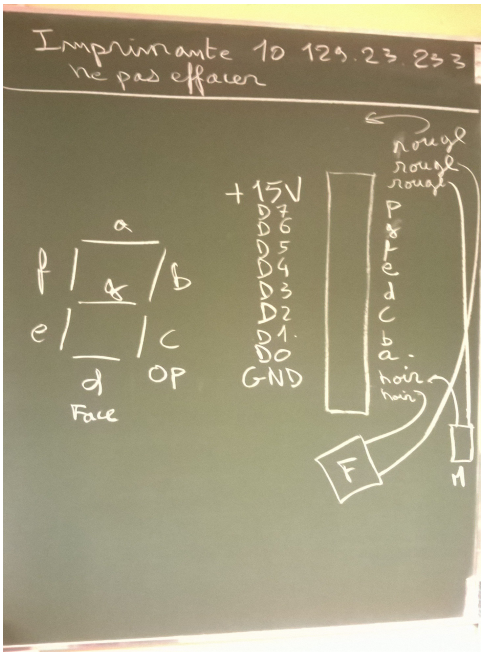

Illustration 13: Choix du poids des segments

Nous avons ainsi associé le bit de poids fort au point, puis dans l'ordre décroissant des segments « g » à « a ». Ceci nous permet donc, dans un seul octet, de ranger la valeur de tous les segments de l'afficheur. Il nous a ensuite fallu faire correspondre à un chiffre particulier la valeur de l'octet qui lui correspond pour l'afficher correctement.

Pour ce faire, nous avons réalisé le tableau suivant, qui associe à chaque chiffre la valeur à ranger dans l'octet que nous enverrons sur l'afficheur :

Caractère	P	G	F	E	D	C	B	A	CODE
0	0	0	1	1	1	1	1	1	0x3f
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5b
3	0	1	0	0	1	1	1	1	0x4f
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6d
6	0	1	1	1	1	1	0	1	0x7d
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7f
9	0	1	1	0	1	1	1	1	0x6f

Illustration 14: Code de chaque chiffre en hexadécimal

De plus notre professeur, Mr. Lequeu, avait dans l'un de ses programmes déjà existants un tableau contenant la plupart des caractères, référencés par leur code ASCII, ainsi que le code à utiliser pour les afficher. Nous avons donc inséré ce tableau dans notre programme, ce qui nous permet désormais d'afficher les lettres en plus des chiffres.

6.3. Chronomètre

Maintenant que nous pouvons afficher ce que nous voulons, ou presque, sur l'afficheur 7-segments, il nous faut mettre en place le chronomètre. Pour ce faire, nous allons utiliser le TIMER1 de l'ATMEGA 8535 en mode comparaison afin de générer une interruption dans le microcontrôleur tous les centièmes de seconde : il faudra donc autoriser les interruptions et programmer la routine afin d'incrémenter les valeurs des variables correspondant aux secondes, dizaines de secondes, dixièmes et millièmes et nous aurons ainsi notre chronomètre.

Dans le programme principal, il nous faudra scruter les capteurs qui nous intéressent comme le fait que la grille de départ soit baissée, qui doit normalement lancer le chronomètre. Il nous faudra ensuite scruter l'état du capteur optique situé dans le premier virage afin d'arrêter le chronomètre quand le premier de la course passera à son niveau.

Enfin, il faudra, lorsque l'on remonte la grille, réinitialiser le chronomètre et n'afficher qu'un temps nul ou bien le message « Pret. » jusqu'à ce que la grille de départ se baisse à nouveau.

6.4. Codage

Afin de « fixer » l'affichage des afficheurs, comme expliqué dans la partie 3, il faut rajouter quelques lignes de code dans les fonctions servant à envoyer le caractère à afficher. Il faut passer le CS à 1, puis le mettre à 0 et le repasser à 1 afin de bloquer l'affichage sur un caractère.

La fonction d'affichage en elle-même fait juste correspondre au caractère le code utilisé pour l'afficher, en hexadécimal et fait un masque « ou » avec le code correspondant au point, selon si ce dernier est activé ou pas ce qui correspond à une variable booléenne dans le prototype de la fonction. De plus, on signale à cette fonction l'adresse de l'afficheur utilisé : 0, 1, 2 ou 3 afin d'afficher le caractère sur le digit qui correspond. Cet adressage est réalisé à l'aide de cavaliers sur les cartes de pilotage situées derrière les afficheurs.

```
// Affiche une valeur sur 8 bits à l'adresse de l'afficheur :
void afficheur1(unsigned char adresse,unsigned char caractere, unsigned char point)
{
  if (point == 1)
     {
      PORTA=(valeur_constant[caractere] | 0x80); // 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;
}
```

Illustration 15: Fonction d'affichage en code C

Pour programmer le chronomètre, nous avons utilisé le timer 1 de l'ATMEGA 8535 en divisant la fréquence du microcontrôleur par 8, ce qui nous donne une fréquence du timer de 2 MHz. Nous avons ensuite programmé le timer en comparaison pour avoir une interruption toutes les 10 ms, ce qui correspond à notre cahier des charges.

```
// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
temps++;
if (temps>=100)
   {
   temps=0;
   seconde++;
   if (seconde>=60)
      {
      seconde=0;
      };
   };
}
```

Illustration 16: Routine d'interruption du timer

Nous avons ensuite autorisé les interruptions sur le microcontrôleur et programme une routine d'interruption sur l'interruption du timer qui incrémente une variable, correspondant aux centièmes de secondes à chaque interruption. Lorsque cette variable atteint 100, nous incrémentons une autre variable, qui correspond elle aux secondes.

Il ne nous reste, dans le programme principal, à traiter ces deux variables avec des divisions et des modulo pour obtenir les dizaines et les unités des secondes ainsi que les dixièmes et centièmes de secondes, codés sur un seul caractère, afin de les envoyer sur l'afficheur.

```
//Traitement du chronomètre
dizaine  = seconde/10;
unite    = seconde-dizaine*10;
dixieme  = temps/10;
centieme = temps-dixieme*10;
```

Illustration 17: Traitement du temps du chronomètre

Il nous faut ensuite coder les tests des capteurs de la grille et du virage : nous devons donc garder en mémoire le passage à l'état haut du capteur optique, lorsque la grille est relevée uniquement, afin d'arrêter le chronomètre, et ne pas le relancer lorsque le capteur optique revient à son état d'origine alors que la grille n'est pas encore relevée. Enfin, nous devons réinitialiser toutes les valeurs des variables pour remettre le chronomètre à 0.

Voilà ce que cela donne en code C, sachant que CAPT2 correspond à la grille de départ et que CAPT1 correspond au capteur optique OMRON. La variable Timer sert à lancer et arrêter le chronomètre tandis que la variable TimerOFF est utilisée pour forcer l'arrêt de ce dernier.

```
// Test de l'ouverture de la grille pour départ chrono -> Timer ON
if (CAPT2 == 0)
{
Timer = 1 ;  //Lancer le timer

  // Test du capteur au premier virage -> Timer OFF
  if (CAPT1 == 0)
  {
  TimerOFF = 1 ;   //Forcer l'arrêt du timer
  }
}

// Test si grille remontée
if (CAPT2 == 1)
{
Timer = 0 ;
TimerOFF = 0 ;
seconde = 0 ;
temps = 0 ;
}
```

Illustration 18: Test des capteurs dans le programme

Enfin, suivant l'état de ces variables Timer et TimerOFF, nous autorisons ou pas le timer1 de l'ATMEGA à se mettre en marche :

```
if(TimerOFF == 1)    //Arret du timer, mémorisation et affichage du temps
{
Timer = 0 ;
TCCR1B = 0x08 ;
afficheur1(0,dizaine,0) ;
afficheur1(1,unite,1) ;
afficheur1(2,dixieme,0) ;
afficheur1(3,centieme,0) ;
}

if (Timer == 1)        //Lancement du timer, affichage du temps
{
TCCR1B = 0x0A ;
afficheur1(0,dizaine,0) ;
afficheur1(1,unite,1) ;
afficheur1(2,dixieme,0) ;
afficheur1(3,centieme,0) ;
}

if (Timer == 0)     //Arrêt du timer
{
TCCR1B = 0x08 ;
afficheur1(0,dizaine,0) ;
afficheur1(1,unite,1) ;
afficheur1(2,dixieme,0) ;
afficheur1(3,centieme,0) ;
}
```

Illustration 19: Mise en route et arrêt du chronomètre

7. ASSEMBLAGE ET TESTS

7.1. AFFICHEURS ET CARTES DE PILOTAGE

Une fois les afficheurs réalisés, peints et vernis, nous avons vissé sur leur face arrière les cartes de pilotage à l'aide d'entretoises.

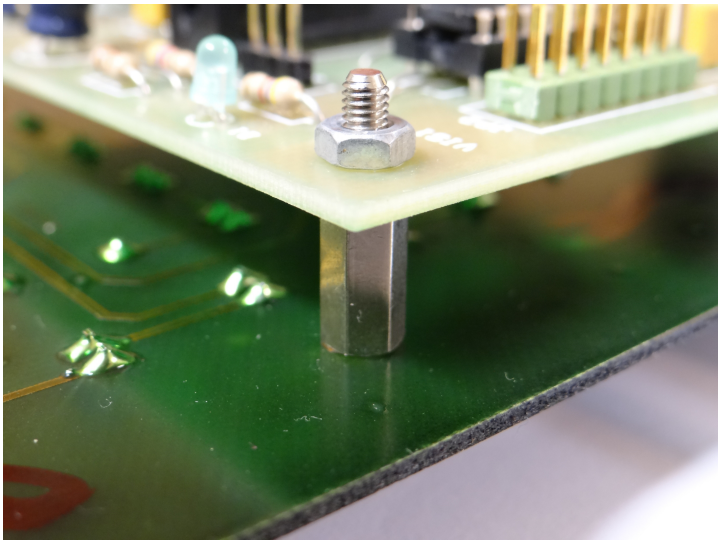

Illustration 20: Entretoise d'un afficheur

Nous avons ensuite eu à câbler chaque segment de l'afficheur au bornier situé sur les cartes de pilotage ainsi que les fils d'alimentation, communs à chaque segment.

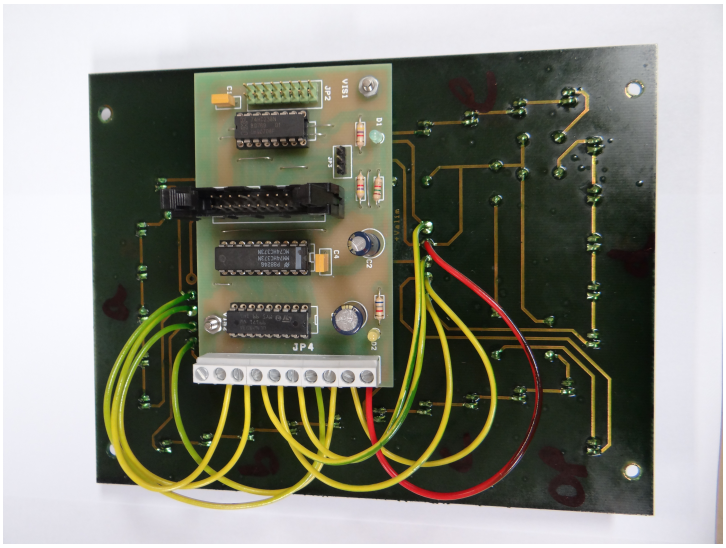

Illustration 21: Carte d'afficheur câblée

7.2. Alimentation des afficheurs

Afin d'alimenter les afficheurs en parallèle, nous avons utilisé des connecteurs MOLEX (55-59 pour les connecteurs femelle et 55-57 pour les connecteurs mâles).

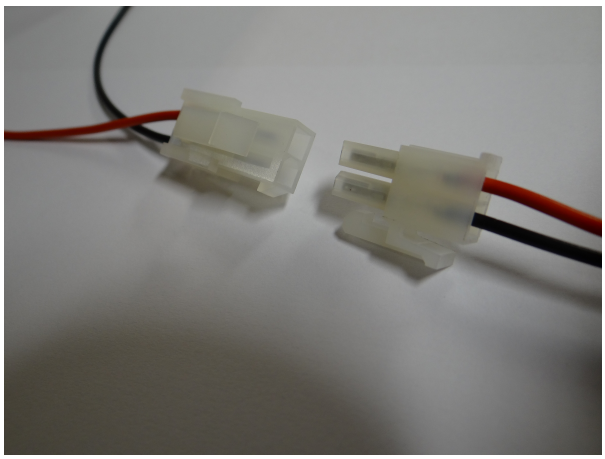

Illustration 22: Embases MOLEX mâle et femelle

Nous avons donc sertis sur chaque afficheur un câble d'alimentation avec une fiche mâle et un autre avec une fiche femelle avec de placer chaque carte d'afficheur en parallèle les une aux autres et les relier à l'alimentation.

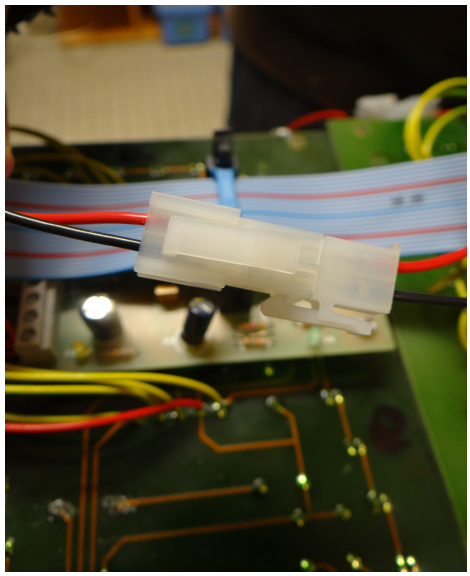

Illustration 23: Connecteur MOLEX

7.3. Connexion avec le microcontrôleur

Pour permettre aux afficheurs ainsi que leurs cartes de pilotage de communiquer avec l'ATMEGA 8535 situé sur la carte de contrôle du chronomètre, nous avons utilisé une nappe sur laquelle nous avons serti des embases femelles RS 232.

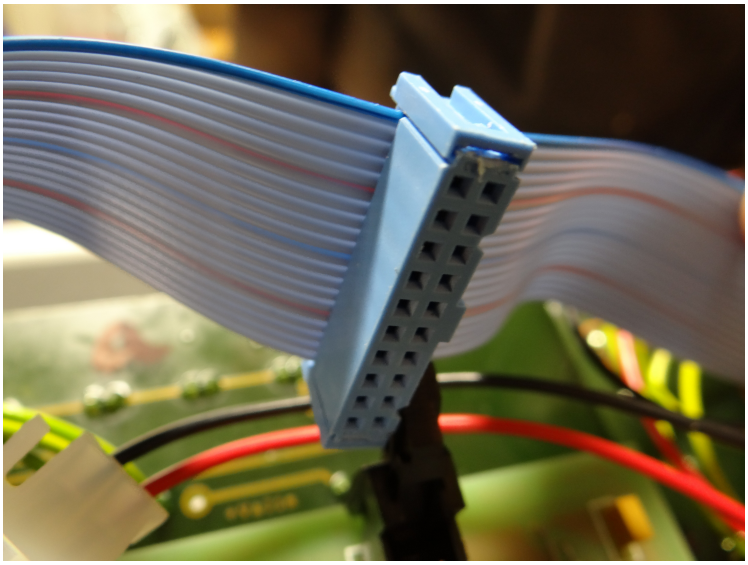

Illustration 24: Embase femelle RS 232

Ces connecteurs sont ensuite clippés sur les cartes de pilotage des afficheurs. Ils sont munis d'un détrompeur afin de ne pas se tromper de sens.

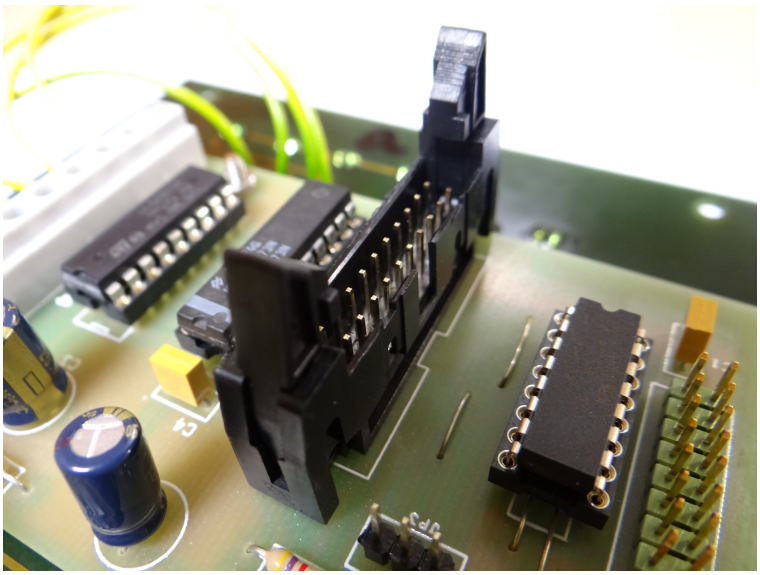

Illustration 25: Embase mâle RS 232

Une fois ces deux embases connectées, le microcontrôleur peut communiquer avec la carte de pilotage de l'afficheur. Il nous faut donc connecter de la même façon toutes les cartes d'afficheurs pour que l'ATMEGA puisse commander les quatre afficheurs.

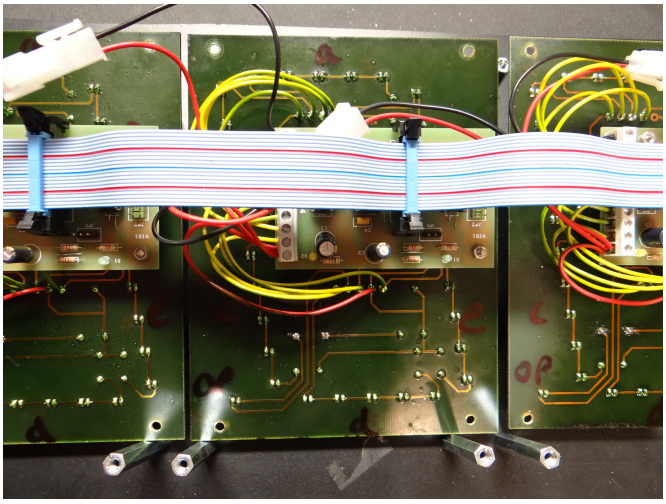

Illustration 26: Afficheur connecté sur la nappe

Une fois tous les afficheurs reliés ensemble avec le microcontrôleur, voici ce que nous obtenons en recto et verso:

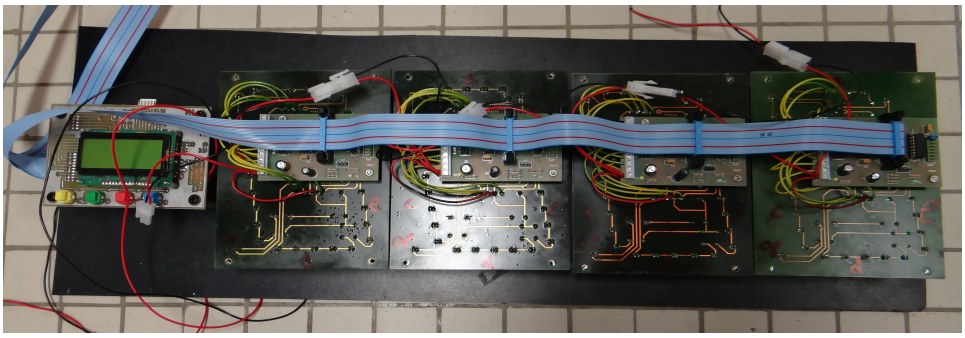

Illustration 27: Chronomètre face verso

Illustration 28: Chronomètre face recto

7.4. Tests

Notre chronomètre est maintenant prêt à être testé. Nous insérons donc le programme C, réalisé sous CodeVision AVR dans le microcontrôleur à l'aide d'un câble de programmation et nous alimentons séparément la carte de contrôle et les quatre afficheurs.

Afin de simuler la grille de départ et le capteur optique, nous avons utilisé respectivement un interrupteur à deux positions et un bouton poussoir que nous avons installé sur un boîtier et que nous avons relié à la carte du microcontrôleur.

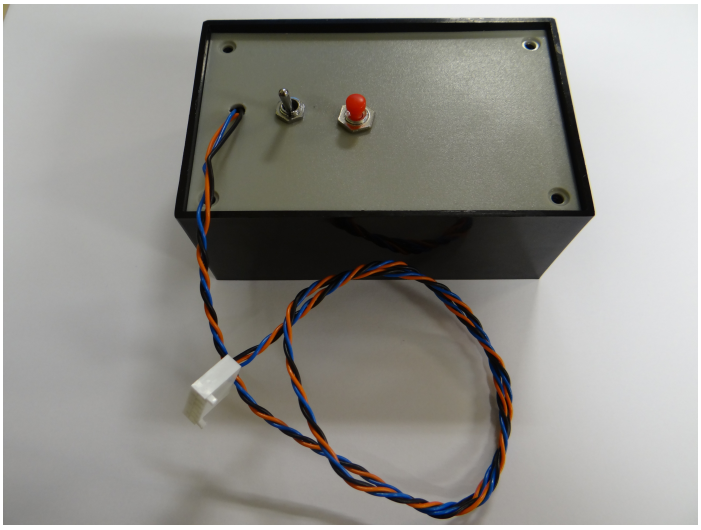

Illustration 29: Boîtier de test pour les capteurs

Nos tests ont été concluants et le cahier des charges a été respecté, il reste donc à aller sur place et se relier à la grille de départ de la piste de BMX et au capteur optique afin de faire les tests dans les conditions réelles d'utilisation du chronomètre.

CONCLUSION

Durant ce projet, nous avons eu l'occasion de mettre nos connaissances en pratique afin de respecter le cahier des charges. Il nous a aussi permis de gagner en assurance ainsi que d'augmenter nos connaissances dans le domaine de l'informatique et l'électricité.

Le projet qui nous avait été fournit avait pour but d'être utilisé par un club de BMX pour l'entraînement de haut niveau, cela nous a permis d'être plus assidus dans le travail et nous motiver à faire de notre mieux afin de satisfaire le client.

Nous avons rencontré quelques problèmes durant ce projet comme une incompréhension du sujet en début de semestre jusqu au rendez-vous avec le client, une grande perte de temps suite au listing de tous les composants afin de connaître le prix de revient du projet et aussi des petits soucis de programmation.

A ce jour, nous avons réussi à terminer le chronomètre en lui-même ainsi que le programme permettant de le faire compter, nous avons aussi programmé le micro contrôleur pour qu'à la suite de la chute de la grille de départ de la piste, le chronomètre se mette en route mais aussi pour que le chronomètre s'arrête et se fige après le passage devant le capteur. Le chronomètre se remettra à 0 lorsque la grille de départ sera remontée.

Nous n'avons pas encore été sur les lieux pour vérifier le fonctionnement avec la grille de départ ainsi que le capteur.

Nous remercions M. Lequeu pour son aide précieuse tout au long de ce projet mais aussi à Alain le magasinier pour sa disponibilité.

RÉSUMÉ

Au cours de ce projet nous avons pris contact avec le client pour savoir exactement ce qu'il attendait de nous.

Notre projet était donc de créer et de programmer un chronomètre géant pour un club de BMX qui devra se déclencher lors de la descente de la grille de départ et qui devra s'arrêter lorsque le premier coureur passe le premier virage et afficher en continu le temps écoulé. Enfin, il devra se réinitialiser lorsque la grille sera remontée et devra être prêt à repartir de nouveau.

Nous avons, avant tout, établi le cahier des charges puis nous avons effectué le listing des composants nécessaires à la fabrication du produit pour avoir le coût de revient final du projet.

Nous avons commencé par réaliser les cartes des afficheurs, puis nous les avons percées avant de les peindre et de souder les composants. Nous avons ensuite ajouté une couche de vernis sur les pistes de cuivre de chaque afficheur afin de les protéger.

Nous avons attaché, à l'aide d'entretoises, les cartes de pilotage à chaque afficheur et nous avons branché l'alimentation ainsi que la nappe pour permettre la communication avec le microcontrôleur.

Une fois les afficheurs accrochés à l'aide d'entretoises sur une plaque de test provisoire, nous avons alimenté le tout et inséré le programme dans le microcontrôleur afin de tester le produit fini.

Les tests ont été concluants et il nous reste désormais à remplacer les capteurs de test par les vrais capteurs, c'est-à-dire celui de la grille de départ et le capteur optique afin de tester le chronomètre dans les conditions réelles.

(263 mots)

GLOSSAIRE

Réalisation

Afficheur

LED

Programmation

microcontrôleur

BMX

Capteur

Grille de départ

Chronomètre

Compétition

Demande

Prix

Données

Typons

Pilotage

CodeVision AVR

Étude

Routage

Câblage

Perçage

ANNEXES

Programme

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.4 Evaluation
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Chronomètre BMX - AFFICHEURS 5 LED 10mm !
Version : 1
Date    : 9 octobre 2014
Author  : Thierry LEQUEU
Company : Association e-Kart
Comments: pour mesurer le temps mis
Comments: à la fin de la première ligne

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

```
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
// Alphanumeric LCD functions
#include <lcd.h>
// Declare your global variables here

#define BP1    PIND.5
#define BP2    PIND.6
#define ENABLE PORTD.7
```

```
#define CAPT4 PIND.2
#define CAPT3 PIND.3
#define CAPT2 PIND.4
#define CAPT1 PIND.5

// Declare your global variables here

flash const unsigned char adresse_constant[16] =
{0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15};

// Pour les afficheurs 5 LED 10mm :
#define DIGIT_A  0x01; { bit b0 }
#define DIGIT_B  0x02; { bit b1 }
#define DIGIT_C  0x04; { bit b2 }
#define DIGIT_D  0x08; { bit b3 }
#define DIGIT_E  0x10; { bit b4 }
#define DIGIT_F  0x20; { bit b5 }
#define DIGIT_G  0x40; { bit b6 }
#define DIGIT_P  0x80; { bit b7 }

// Pour afficheurs 5 LED 10mm - octet de 0 à 255 en ASCII
flash const unsigned char valeur_constant[] =
            {
            0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,   // octet   0 …   7 .
            0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71,   // octet   8 …  15 .
            0x01,0x02,0x04,0x08,0x10,0x20,0x00,0x00,   // octet  16 …  23 .
            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet  24 …  31 .
            0x00,0x30,0x20,0x00,0x00,0x00,0x00,0x20,   // octet  32 …  39 .
            0x39,0x0F,0x80,0x00,0x80,0x40,0x80,0x52,   // octet  40 …  47 .
             0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,   // octet  48 …  55 .
```

```
0x7F,0x6F,0x04,0x00,0x46,0x48,0x70,0xA7,   // octet  56 …  63 .
0x7B,0x77,0x7C,0x39,0x5E,0x79,0x71,0x3D,   // octet  64 …  71 .
0x76,0x06,0x1E,0x76,0x38,0x37,0x37,0x3F,   // octet  72 …  79 .
0x73,0x67,0x50,0x6D,0x78,0x3E,0x3E,0x3E,   // octet  80 …  87 .
0x76,0x72,0x5B,0x39,0x64,0x0F,0x01,0x08,   // octet  88 …  95 .
0x20,0x77,0x7C,0x58,0x5E,0x79,0x71,0x6F,   // octet  96 … 103 .
0x74,0x04,0x0E,0x76,0x30,0x54,0x54,0x5C,   // octet 104 … 111 .
0x73,0x67,0x50,0x6D,0x78,0x1C,0x1C,0x1C,   // octet 112 … 119 .
0x76,0x66,0x5B,0x39,0x30,0x0F,0x01,0x80,   // octet 120 … 127 .
0x71,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 128 … 135 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 136 … 143 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 144 … 151 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 152 … 159 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 160 … 167 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 168 … 175 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 176 … 183 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 184 … 191 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 192 … 199 .
0x00,0x00,0x00,0x00,0x00,0x41,0x00,0x00,   // octet 200 … 207 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 208 … 215 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 216 … 223 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 224 … 231 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 232 … 239 .
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,   // octet 240 … 247 .
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00    // octet 248 … 255 .
};

unsigned char tampon[20],seconde,seconde1,temps,temps1,dizaine,unite,
 unite_0,dixieme,centieme,test,var,feux,stopsec,stoptemp;

int Timer, TimerOFF;

// Declare your global function  here



// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
temps++;
if (temps>=100)
   {
  temps=0;
```

```
   seconde++;
   if (seconde>=100)
      {
      seconde=0;
      } ;
if (test == 1) {
   temps1++;
   if (temps1>=100)
      {
      temps1=0;
      seconde1++;
      if (seconde1>=100)
         {
         seconde1=0;
         };
      };
  }

// Affiche un caractère sur 8 bits à l'adresse de l'afficheur :
void afficheur1(unsigned char adresse,unsigned char caractere, unsigned char point)
{
  if (point == 1)
     {
      PORTA=(valeur_constant[caractere] | 0x80); // 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;
}

// Affiche une valeur sur 8 bits à l'adresse de l'afficheur :
void afficheur2(unsigned char adresse,unsigned char caractere, unsigned char point)
{
  if (point == 1)
     {
```

```
     PORTA=(caractere | 0x80); // Un caractère avec le point en 0x80 pour D7 !
    }
    else
    {
    PORTA=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;
}

// Routine de test des segments :
void test_segments_1(void)
{
     for (var=11;var<19;var++)
       {
       afficheur1(0,var,0);
       afficheur1(1,var,0);
       afficheur1(2,var,0);
       afficheur1(3,var,0);
       afficheur1(4,var,0);
       afficheur1(5,var,0);
       afficheur1(6,var,0);
       afficheur1(7,var,0);


    delay_ms(500);
    }
}

// Routine de test des segments :
void test_segments_2(void)
{
afficheur1(0,8,1); // Un caractère
afficheur1(1,8,1); // Un caractère
afficheur1(2,8,1); // Un caractère
afficheur1(3,8,1); // Un caractère
afficheur1(4,8,1); // Un caractère
afficheur1(5,8,1); // Un caractère
}
```

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

// 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; en entrée
DDRA=0xFF; // en sortie

// Port B 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
PORTB=0xFF;
// DDRB=0x00; en entrée
DDRB=0xFF; // en sortie




// 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=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=P State4=P State3=T State2=T State1=T State0=T
PORTD=0x30;
ENABLE=1;
DDRD=0x80;
// DDRD=0x00; en entrée
// DDRD=0x38; // BP1 BP2 OC1A IN   INT1 IN  IN  IN
```

```
        // IN7 IN6 OUT5 OUT4 OUT3 IN2 IN1 IN0

// 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: 2000,000 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: On
// Compare B Match Interrupt: Off
TCCR1A=0x40;
TCCR1B=0x0A;



TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E; // Base de temps = 2 MHz, soit 0,5 us.
OCR1AL=0x20; // Interruption quand on arrive à 20 000 (0x4E20 soit 10 ms)
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: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// 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: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// LCD module initialization
lcd_init(16);
```

```
lcd_clear();


/* switch to writing in Display RAM */
lcd_gotoxy(0,0);
lcd_putsf("Chronometre");
lcd_gotoxy(0,1);
lcd_putsf("Pour BMX");

PORTB=0xFF;
seconde=0;
temps=0;
seconde1=0;
unite=0;
unite_0=0;
temps1=0;
test=0;
var='0';
ENABLE=0;
//test_segments_1();

afficheur2(0,0,0); // Un caractère
afficheur2(1,0,0); // Un caractère
afficheur2(2,0,0); // Un caractère
afficheur2(3,0,0); // Un caractère

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

while (1)
      {
      // Place your code here

      //Traitement du chronomètre
      dizaine  = seconde/10;
      unite    = seconde-dizaine*10;
      dixieme  = temps/10;
      centieme = temps-dixieme*10;

      if (centieme >> 9)
      {
      centieme = 0 ;
      }
```

```
// Affiche sur l'écran LCD de la carte du microcontrôleur
 sprintf(tampon,"Temps = %2d:%2d",seconde,temps);
  lcd_gotoxy(0,2);
  lcd_puts(tampon);

  // Affiche l'état des entrées sur l'écran LCD
  sprintf(tampon,"PIND = %3X",PIND);
  lcd_gotoxy(0,3);
  lcd_puts(tampon);

  // sprintf(tampon,"Temps = %1d%1d:%1d%1d",dizaine, unite,dixieme,centieme);
  // Ne marche pas, le caractère des centièmes déborde

  // Test de l'ouverture de la grille pour départ chrono -> Timer ON
 if (CAPT2 == 0)
     {
     Timer = 1 ;  //Lancer le timer

       // Test du capteur au premier virage -> Timer OFF
       if (CAPT1 == 0)
       {
       TimerOFF = 1 ;   //Forcer l'arrêt du timer
       }
}
```

```
// Test si grille remontée
if (CAPT2 == 1)
{
Timer = 0 ;
TimerOFF = 0 ;
seconde = 0 ;
temps = 0 ;
}

if(TimerOFF == 1)    //Arret du timer, mémorisation et affichage du temps
{
Timer = 0 ;
TCCR1B = 0x08 ;
afficheur1(0,dizaine,0) ;
```

```
    afficheur1(1,unite,1) ;
    afficheur1(2,dixieme,0) ;
    afficheur1(3,centieme,0) ;
    }

if (Timer == 1)       //Lancement du timer, affichage du temps
{
    TCCR1B = 0x0A ;
    afficheur1(0,dizaine,0) ;
    afficheur1(1,unite,1) ;
    afficheur1(2,dixieme,0) ;
    afficheur1(3,centieme,0) ;
    }

    if (Timer == 0)     //Arrêt du timer
    {
    TCCR1B = 0x08 ;
    afficheur1(0,dizaine,0) ;
    afficheur1(1,unite,1) ;
    afficheur1(2,dixieme,0) ;
    afficheur1(3,centieme,0) ;
    }
    }
}
```

TABLEAUX DE PRIX

1	Composant	Désignation	Référence	Quantité	Prix unitaire HT	Prix total TTC	Fournisseur	quantité achetée
2	LED	D1 – D37	10034R1C-CSE-D	148	0,190 €	33,744 €	E KART	148
3	Résistance	R1 – R8	120 Ohm	32	0,020 €	0,768 €	radiospares	32
4	diode Zener	D36	BZX85C12V	4	0,156 €	0,749 €	radiospares	5
5	Vis	Vis1 – Vis 8		16		0,000 €		
6	écrous	--				0,000 €		
7	plaque de gravure	--	PRC4231	2	20,300 €	48,720 €	Bernier	2
8								
9					Prix total :	83,981 €		

Illustration 30: Tableau de prix des afficheurs 7-segments

Nom	Référence	Valeur	Empreinte	Vendeur	Code commande	Quantité	Prix HT	Total H.T.	TVA	Prix total par composant
Afficheur LCD 16x4	AFF1	LCD	MC1604C	Farnell	9448659	1	22,870 €	22,870 €	4,574 €	27,444 €
Condensateur	C2	10 µF ; 6,3V	RADIAL06	RS	381-508	1	0,686 €	0,686 €	0,137 €	0,823 €
Condensateur	C3	100 nF	CK06	RS	699-5070	1	0,276 €	0,276 €	0,055 €	0,331 €
Condensateur	C4, C5	22 pF	CK06	RS	538-1219	2	0,150 €	0,300 €	0,060 €	0,360 €
Condensateur	C6	22 µF ; 25V	RADIAL08	RS	116-874	1	0,092 €	0,092 €	0,018 €	0,110 €
Condensateur	C7	470 µF ; 6,3V	RADIAL06L	RS	224-4094	1	0,298 €	0,298 €	0,060 €	0,358 €
Condensateur	C8	10 µF ; 6,3V	RADIAL04	RS	381-508	1	0,686 €	0,686 €	0,137 €	0,823 €
LED Jaune	D2	3 mm	LED03	RS	247-1612	1	0,276 €	0,276 €	0,055 €	0,331 €
Diode de redressement	D3	1N4007	DO41	RS	748-4975	1	0,050 €	0,050 €	0,010 €	0,060 €
Diode de redressement	D4	1N5819	DO41	RS	652-7337	1	0,095 €	0,095 €	0,019 €	0,114 €
LED Verte (2 mA)	D5	2 mA Verte	LED03	RS	247-1606	1	0,124 €	0,124 €	0,025 €	0,149 €
Embase femelle 2,54mm sécable (à l'unité)	JP1	RS 232	04PL1	Farnell	9728910	23	0,153 €	3,519 €	0,704 €	4,223 €
Conn. ISP (Embase droite 10 voies)	JP2	CON ISP	10SH100L	Farnell	1298785	1	1,870 €	1,870 €	0,374 €	2,244 €
Connecteur Data	JP3	DATA	20SH100L	Farnell	2215266	1	1,460 €	1,460 €	0,292 €	1,752 €
Conn. Inputs (embase 6 voies coudée)	JP4	INPUTS	06PL1	Farnell	1360146	1	1,330 €	1,330 €	0,266 €	1,596 €
Conn . 12V	JP10	12V	WEID2	RS	294-7349	1	0,586 €	0,586 €	0,117 €	0,703 €
Inductance	L1	47 µH	RADIAL06L	Farnell	2309773	1	0,790 €	0,790 €	0,158 €	0,948 €
Résistance variable	P1	10K	RAJ1	Farnell	1689844	1	0,480 €	0,480 €	0,096 €	0,576 €
Quartz	Q1	16 MHz	HC18UV	RS	672-0218	1	0,426 €	0,426 €	0,085 €	0,511 €
Résistance	R2, R10	1,5K	RC04L	Farnell	2329891	2	0,034 €	0,068 €	0,014 €	0,082 €
Résistance	R3, R4, R6, R7	4,7K	RC04L	Farnell	2329947	4	0,034 €	0,136 €	0,027 €	0,163 €
Résistance	R5	10K	RC04L	Farnell	2329855	1	0,029 €	0,029 €	0,006 €	0,035 €
Strap	R8, R9	0	/	/	/	2	0,000 €	0,000 €	0,000 €	0,000 €
Bouton poussoir	SW1, SW2, SW3, SW4	TOUCHE	REDROND	e44.com	D6RR	4	0,700 €	2,800 €	0,560 €	3,360 €
Microcontrôleur	U1	Atmega 8535	40DIP300L	Farnell	9171444	1	2,890 €	2,890 €	0,578 €	3,468 €
Régulateur de tension	U2	LM2574-5.0	08DIP300L	Farnell	2100264	1	3,040 €	3,040 €	0,608 €	3,648 €
Visserie	VIS1, VIS2, VIS3, VIS4	/	M3L	RS	546-6196	4	0,035 €	0,141 €	0,028 €	0,169 €
Embase mâle 2,54mm sécable (à l'unité)	/	/	04PL1	Farnell	247-1606	15	0,402 €	6,030 €	1,206 €	7,236 €
Entretoises 8mm	/	/	/	RS	105-7760	4	0,403 €	1,612 €	0,322 €	1,934 €
Plaque de gravure 20x30cm	/	/	/	Bernier	PRC4231	1	20,300 €	20,300 €	4,060 €	24,360 €
Borne de raccordement	/	/	/	RS	403-875	1	0,916 €	0,916 €	0,183 €	1,099 €
Ecrous pour visserie	/	/	/	RS	560-293	4	0,02184	0,087 €	0,017 €	0,105 €

PRIX TOTAL	89,116 €

Illustration 31: Tableau de prix de la carte du microcontrôleur

N°	Quantité	Référence (schéma)	Valeur	Empreinte	Informations	ode Commande	Fournisseur	Prix unit. HT	Prix Total HT
1	2	C1, C4	100 nF	CK06	AVX CK06BX104K, 100 nF, 100 V	699-4948	Radiospare	0,356 €	0,712
2	1	C2	220 µF / 6,3 V FC	RADIAL08		449-0839	Radiospare	0,092 €	0,092
3	1	C3	220 µF / 25 V FC	RADIAL08		526-1480	Radiospare	0,300 €	0,3
4	2	D1, D2	3mm / 2mA	LED3	LED 3 mm Vert	247-1606	Radiospare	0,124 €	0,248
5	1	JP1	DATA	20SH100L	T816120A1S101CEU - EMBASE VERROUILL. DROITE 2.54MM 20VOIES	2215266	Farnell	1,460 €	1,46
6	8	JP2	ADRESSE	16SH100	TE Connectivity, 16 Contacts, 2 Rangées, entraxe 2.54 mm, 3 A, Droit	6689618	Radiospare	0,136 €	1,088
7	3	JP3	SELECT	03PL1	Stelvio Kontek, 40 Contacts, 1 Rangées, entraxe 2.54 mm, 3 A, Droit	156-049	Radiospare	0,039 €	0,117
8	2	JP4	+15 V	10PL2	Bornier a vis 2 way	408-7871	Radiospare	0,150 €	0,3
8	2	JP4	+15 V	10PL2	Bornier a vis 3 way	408-7887	Radiospare	0,095 €	0,1896
10	2	R10,R9	4,7 k	RC04	±5%, 0,25W, Dimensions 2.3 Dia. x 6.3mm	707-7726	Radiospare	0,019 €	0,038
11	1	R11	1,5k	RC04	±5%, 0,25W, Dimensions 2.4 Dia. x 6.3mm	806-6424	Radiospare	0,137 €	0,137
12	1	R12	6,8k	RC04	±5%, 0,25W, Dimensions 2.4 Dia. x 6.3mm	806-6421	Radiospare	0,137 €	0,137
13	1	U2	ULN2803	18DIP300L	NPN 0.5 A 50 V, Octuple Emetteur commun, PDIP, 18 broches	714-1167	Radiospare	0,610 €	0,61
14	1	U4	74HC373N	18DIP300L	Verrou octal type D Type D 3 états PDIP 20 broches 2 to 6V	169-6135	Radiospare	0,670 €	0,67
15	1	U5	74HC238N	20DIP300	Déco. Multiplexeur HC, Sans inversion PDIP 16 broches 2 to 6V	483-6493	Radiospare	0,623 €	0,623
16	2	écrou			Hexagonal / Type M3		la visserie	0,050 €	0,1
17	2	rondelle de frein			Type M3	521800 – 62	Conrad	0,016 €	0,032
18	2	entretoise	Lg : 10mm		Type mâle Type M3x10 / femelle type M3	463010	Modelisme-vartex.com	0,320 €	0,64
19	2	Vis	Lg : 8mm		Type Cylindre tête fendue / M3x8	Z517	RC – diffusion	0,330 €	0,66

Illustration 32: Tableau de prix de la carte de pilotage

			Prix Total	7,8536				
			TVA	20 %				
			Prix TTC	9,42432				
					Ceci est le prix pour UNE carte, SANS main d'œuvre			
					Il faut ajouter le support de carte :			
20	1	Support de carte	20x30		Support de carte pour circuit imprimé 20x30 (un seul suffit pour les 4 cartes)	PRC4231	Bernier Electronik	11,288
			Coût total des 4 cartes :	51,24288				

Illustration 33: Prix total de la carte de pilotage

BIBLIOGRAPHIE

1: Thierry Lequeu, Documentation de la carte de l'ATMEGA 8535, 2013, http://www.thierry-lequeu.fr/data/DI8535-2.pdf

2: Thierry Lequeu, Documentation du module afficheur 8 sorties, 2009, http://www.thierry-lequeu.fr/data/DISPLAY2.pdf

3: Thierry Lequeu, Documentation de l'afficheur 7-segments, 2014, http://www.thierry-lequeu.fr/data/AFF7S.pdf

4: Thierry Lequeu, Site de Thierry Lequeu, 2014, http://www.thierry-lequeu.fr/

[1][2][3][4]

INDEX DES ILLUSTRATIONS