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

Étude et Réalisation 2ème année

Balise de mesure de vitesse pour l'épreuve de 50 mètres départ arrêté

Clément TOURNILLON
Anthony BRAIN
Groupe Q1
2010/2012

Enseignants
Thierry LEQUEU
Charles GLIKSOHN

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

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

Étude et Réalisation 2ème année

Balise de mesure de vitesse pour l'épreuve de 50 mètres départ arrêté

Clément TOURNILLON
Anthony BRAIN
Groupe Q1
2010/2012

Enseignants
Thierry LEQUEU
Charles GLIKSOHN

Sommaire

Introduction

Au cours du semestre 3, nous avions choisi de continuer un projet qui n'avait pas encore été terminé. Ce projet était « Balise de mesure de temps pour l'épreuve de 50m départ arrêté ». Ce projet avait été entièrement terminé. Nous avons donc décidé dans le cadre du semestre 4, d'apporter diverses améliorations à ce projet, afin de le rendre encore plus complet, et dont la principale est une fonction de mesure de vitesse à l'arrivée du kart.

Notre projet pour ce semestre sera donc « Balise de mesure de vitesse pour l'épreuve de 50m départ arrêté». Ce projet est réalisé pour l'association e-kart de M. Thierry LEQUEU.

Après une présentation des balises de mesure de vitesse, nous présenterons le principe et la la réalisation, à l'aide d'un timer, de la mesure de vitesse. Puis nous présenterons l'étude de la partie programmation. Enfin nous terminerons par les diverses améliorations que nous avons pu rajouter en fin de projet.

1. Présentation des balises de mesure de vitesse

1.1. L'épreuve du 50 mètres départ arrêté

L'épreuve du 50 mètres départ arrêté est une épreuve de vitesse qui peut voir concourir 1 ou 2 karts simultanément. Les karts doivent donc parcourir 50 mères en ligne droite avec un départ arrêté en effectuant le meilleur temps possible.

Pour chronométrer le temps de parcours du kart, on dispose de 2 bornes, une au départ et une à l'arrivée, équipées de capteurs capables de détecter le passage d'un kart. De plus, on dispose d'un afficheur 7 segments pour afficher le temps jusqu'au centième de seconde près, étant donné que les karts parcourent ces 50 mètres en moins de 10 secondes généralement.

1.2. Cahier des charges

Ce projet est réalisé pour le compte de l'association e-Kart présidé par Thierry LEQUEU pour le challenge e-Kart, afin de fournir un moyen de mesurer la vitesse de façon précise et fiable au km/h près(au dixième de km/h près si possible).

Pour réaliser ce projet, nous devons répondre aux exigences suivantes:

- détection sans contact du passage du kart d'une portée de 4 mètres.
- affichage de la vitesse du kart à l'aide de grands afficheurs à LEDs.

Pour ce faire, nous disposons d'une borne de départ et d'une borne d'arrivée, muni chacune d'un microcontrôleur Atmega8535, qui sont capables de communiquer entre elles à l'aide d'une liaison série filaire RS232. Le dispositif devra afficher la vitesse en km/h.

Le kart sera détecté par les bornes de départ et d'arrivée à l'aide de capteurs infrarouges. Il y aura 2 capteurs sur chaque borne, le chronomètre se déclenchera lorsque le kart franchira le 2^{e} capteur de la borne de départ, après être resté 10 secondes devant le 1er capteur au préalable, et s'arrêtera lorsqu'il franchira, 50 mètres plus loin, le 1er capteur de la borne d'arrivée.

On peut voir les différents cas sur l'illustration suivante:

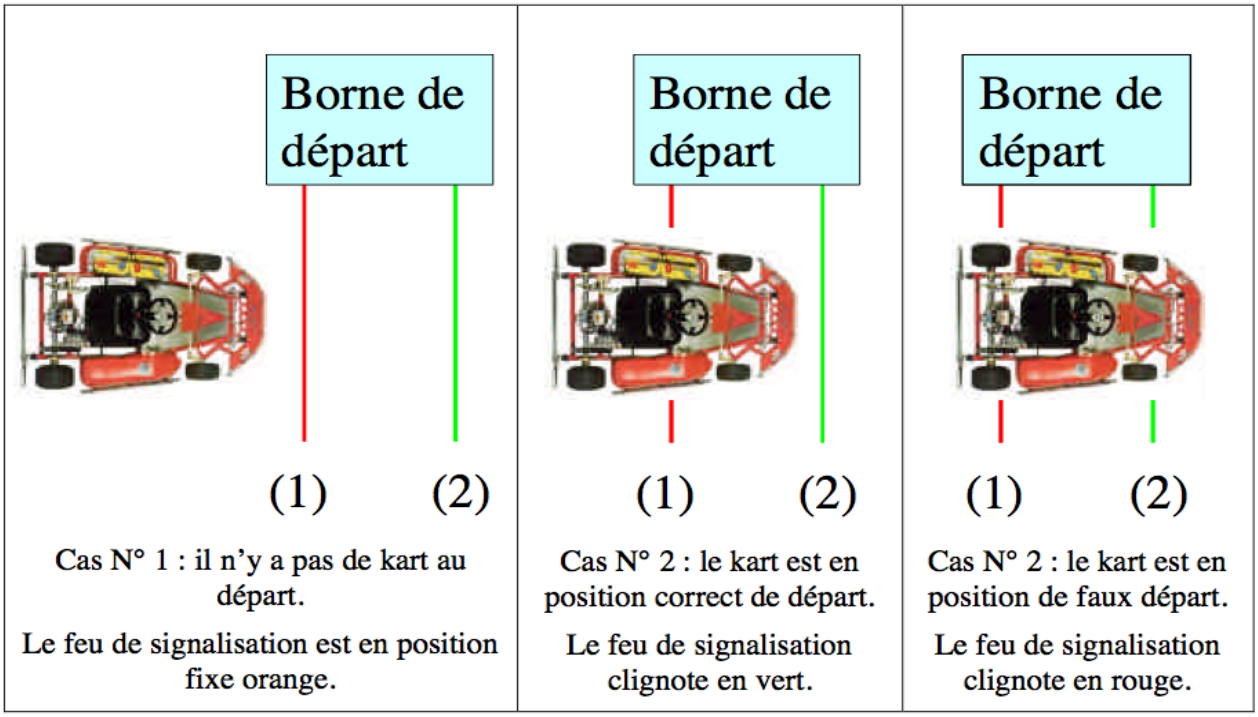


Illustration 1: Schéma de principe de la mesure du temps[1]

Pour calculer la vitesse, on la mesurera entre les 2 capteurs de la borne d'arrivée et on l'affichera sur un second afficheur.

On devra réaliser la programmation du microcontrôleur AtMéga8535 à l'aide du logiciel Code Vision AVR. Ce microcontrôleur peut calculer le temps mis par le kart pour parcourir les 50 mètres, et devra pouvoir calculer sa vitesse, et également les afficher sur des afficheurs à LEDs. On pourra tester le fonctionnement du microcontrôleur à l'aide de l'écran LCD présent sur la carte.

Nous avons effectué une planification du travail que l'on effectuera à chaque séance sous forme du planning suivant:

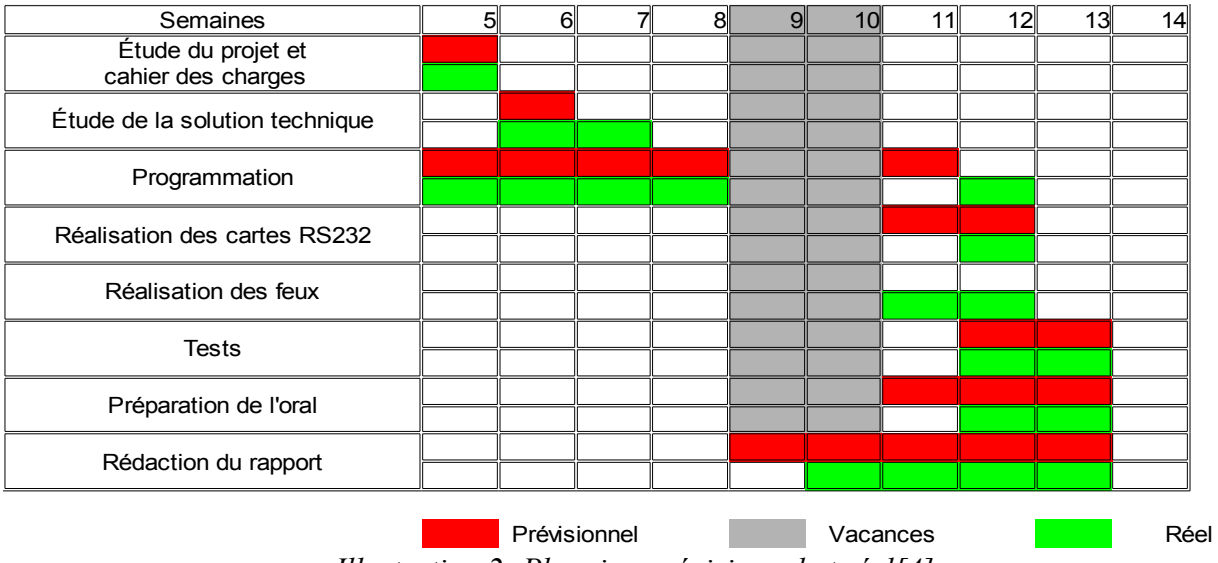


Illustration 2: Planning prévisionnel et réel[4]

Nous pouvons remarquer que notre planning réel a été très proche de notre planning prévisionnel. Cependant comme nous avons été plus rapide que prévu, nous avons pu en semaine 11 et 12 effectuer la tâche de la réalisation des feux tricolores pour le départ de l'épreuve du 50 mètres.

De plus la réalisation des cartes RS232 a pu se faire en seulement une demie séance, grâce à Monsieur LEQUEU, au lieu de 2 séances prévues.

2. Mesure de la vitesse

2.1. Principe de la mesure

Pour mesurer la vitesse de passage du kart à l'arrivée, nous nous servirons des 2 capteurs présents sur la borne d'arrivée. Lorsque que le kart franchira le 1er capteur, nous déclencherons un timer puis lorsque le kart franchira le deuxième capteur nous arrêterons ce timer.

Connaissant maintenant ce temps, ainsi que la distance séparant ces 2 capteurs qui est de 40 centimètres, nous pouvons calculer la vitesse du kart à l'aide de la formule :

$$\boxed{v = \frac{d}{t}}$$ avec :

- v : vitesse en km/h.
- d : distance entre les 2 capteurs en km.
- t : temps de passage du kart entre les 2 capteurs en h.

Il a donc fallu convertir la distance de 40 centimètres en : $\boxed{d = 0{,}4.10^{-3}\,km}$ et le temps qui nous sera donné par le timer en secondes a du être divisé par 3600 afin de l'avoir en heures. Nous sommes donc arrivés à la formule suivante :

$$\boxed{v = \frac{0{,}4.10^{-3} * 3600}{t} = \frac{1{,}44}{t}}$$ avec : t : le temps donné par le timer en secondes

L'affichage de cette vitesse se fera sur le même afficheur que celui utilisé pour afficher le temps. Le protocole sera d'afficher le temps de parcours et la vitesse d'arrivée du kart avec une alternance toutes les 2 secondes, qui se répètera 5 fois avant de terminer sur un affichage fixe du temps.

2.2. Choix du timer

Pour effectuer la mesure du temps de parcours du kart, projet réalisé au cours du semestre 3, nous utilisons le timer 1 de 16 bits du microcontrôleur AtMéga8535 . Nous avons donc la possibilité d'utiliser également ce timer pour notre mesure de vitesse, cependant ce timer s'exécute toutes les 10 millisecondes, nous aurons donc dans ce cas une approximation au centième de seconde. Ce qui donnerait par exemple, pour une vitesse réelle de 100km/h, un temps de passage entre les 2 capteurs de :

$$\boxed{v = \frac{d}{t}\; d'où\; t = \frac{d}{v} = \frac{0{,}4}{100.10^{3}} = 4.10^{-6}\,h = 14{,}4\,ms}$$

Or, comme le timer s'exécute toutes les 10 ms, en 14,4ms il s'exécuterait une fois et irait jusqu'à la fin de son deuxième cycle, on trouverait donc un temps de 20ms. Ce qui donnerait une vitesse calculée de :

$$v = \frac{1,44}{20.10^{-3}} = 72\,km/h$$

Ainsi pour une vitesse réelle de 100km/h, on mesurerait une vitesse de 72km/h ce qui donne un défaut de 28%, beaucoup trop élevé pour qu'il soit acceptable.

Nous avons donc étudié la solution d'un nouveau timer dans notre projet, qui s'exécutera toutes les millisecondes. De même, pour une vitesse de 100km/h et donc un temps de passage entre les 2 capteurs de 14,4ms, le timer s'exécuterait 15 fois ce qui donnerait un temps de 15ms. La vitesse calculée serait alors de :

$$v = \frac{1,44}{15.10^{-3}} = 96\,km/h$$

Le défaut ne serait plus alors que de 4%, ce qui est acceptable pour notre projet.

De plus, on sait que l'allure de la courbe du défaut en fonction de la vitesse aurait la forme d'une exponentielle, car plus le kart passerait rapidement et plus le défaut serait important. Et comme les karts les plus rapides auront généralement une vitesse autour de 100km/h, la solution d'un timer s'exécutant toutes les millisecondes a été retenue. Nous avons donc le choix entre les 2 timers de 8 bits, comme ces 2 timers sont identiques nous avons choisi librement de prendre le timer 0.

Pour valider totalement notre choix, nous avons également vérifié à l'oscilloscope le temps de cycle du microcontrôleur AtMéga8535, pour être certain que notre fonction aurait bien la possibilité de s'exécuter toutes les millisecondes. La capture d'écran de l'oscilloscope donne:

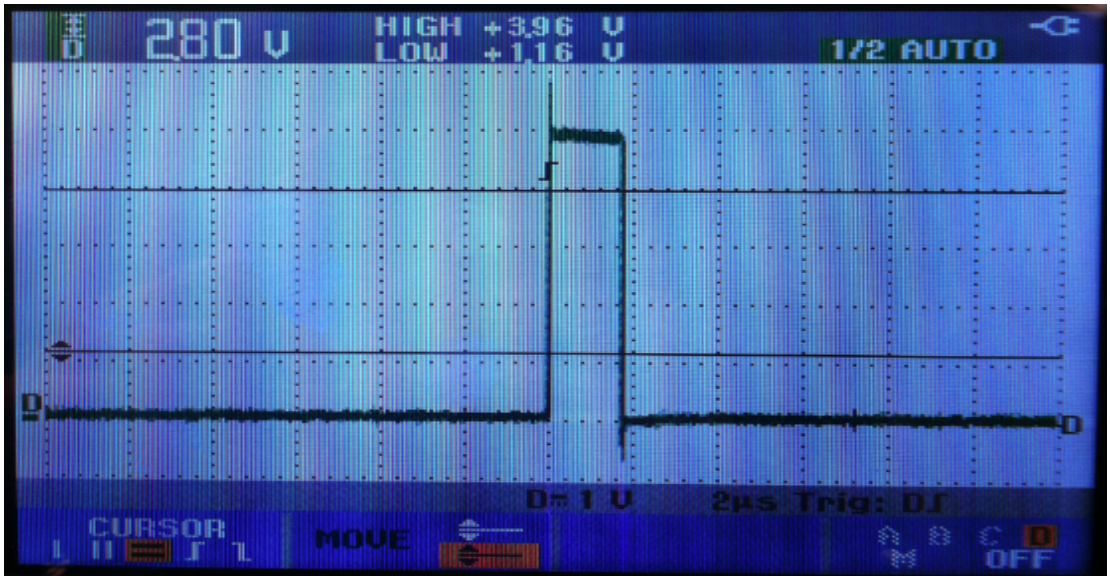


Illustration 3: Temps de cycle du microcontrôleur à l'oscilloscope[4]

Nous pouvons ainsi voir que le temps de cycle est de l'ordre de 2 microsecondes, et comme il est bien inférieur à 1 milliseconde, cela nous permet d'affirmer que notre fonction pourra bien s'exécuter toutes les millisecondes.

2.3. Configuration du timer

Afin que le timer 0 se comporte comme on le souhaite, il y a différents registres à configurer avant son utilisation.

2.3.1. Registre OCR0

La fréquence de l'oscillateur de la carte AtMéga8535 étant de 16MHz, l'horloge aura une période: $T = \frac{1}{f} = \frac{1}{16.10^6} = 62{,}5\,ns$

Comme nous devons compter millième par millième de seconde, le timer devra s'exécuter toutes les 1ms. Il faudra donc compter: $\frac{1.10^{-3}}{62{,}5.10^{-9}} = 16\,000\ périodes\,d'horloge.$

Mais on sait que le timer0 ne peut compter que de 0 à 255, car c'est un timer 8 bits, pour compter 16 000 périodes d'horloge il faut donc effectuer un prédivision de la fréquence par 64. La fréquence passe donc à 250KHz.

Ce qui donne une période de:

$$T = \frac{1}{250.10^3} = 4\,\mu s$$

Il faut donc compter : $\frac{1.10^{-3}}{4.10^{-6}} = 250\ périodes\,d'horloge.$

Il faut donc avoir une valeur de comparaison de 250, qui correspond à 0xFA en hexadécimal, ce qui se règle dans le registre de comparaison du timer0 OCR0. On aura donc OCR0 = 0xFA.

2.3.2. Registre TCCR0

Le registre TCCR0 de contrôle nous servira pour autoriser ou non le timer mais aussi pour diviser la fréquence du quartz par 64 comme nous l'avons vu précédemment.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Illustration 4: Configuration du registre TCCR0[2]

Le bit 7 permettrait de ne pas réinitialiser le timer lorsqu'il a atteint sa valeur de comparaison en mode CTC[1]. Or on veut qu'il le fasse donc on laisse le bit à 0.

1 CTC : Clear Timer on Compare match: remettre le timer à 0 dès qu'il atteint une valeur de comparaison.

Les bits 6 et 3 permettent de choisir le mode d'opération du timer:

Table 39. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Illustration 5: Sélection du mode d'opération du timer[2]

On souhaite avoir le mode d'opération CTC, on met donc le bit 6 à 0 et le bit 3 à 1.

Les bits 5 et 4 permettent de définir l'action qui doit se produire sur la sortie OC0 lorsque les valeurs présentes dans le registre de comparaison OCR0 et le compteur TCNT0 sont égales. Pour notre projet nous n'avons pas besoin de cette action, les 2 bits seront donc laissés à 0.

Enfin les bits 2, 1 et 0 permettent d'autoriser ou non le timer et de choisir la prédivision de la fréquence.

Table 43. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Illustration 6: Autorisation et prédivision du timer[2]

Nous avons besoin d'une prédivision de la fréquence de l'oscillateur par 64, ce qui donne les bits 1 et 0 à 1 et le bit 2 à 0. De plus si ces 3 bits sont mis à 0 le timer n'est pas autorisé.

Finalement, pour autoriser le timer il faudra donc écrire TCCR0 = 0x0B et pour ne pas l'autoriser TCCR0 = 0x08.

2.3.3. Registre TCNT0

Le registre TCNT0 est le compteur du timer. Comme on commence initialement à 0 secondes, alors on met le compteur à 0 ce qui donne : TCNT0 = 0x00.

2.3.4. Registre TIMSK

Le registre de validation des interruptions permet d'autoriser la génération d'interruptions du timer.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Illustration 7: Configuration des interruptions[2]

Ainsi il suffit de mettre le bit 1 à 1 pour autoriser, lorsqu'il y a égalité entre le compteur et le registre de comparaison, l'exécution de la routine d'interruption correspondante. C'est donc cette interruption qui s'exécutera toutes les millisecondes et nous permettra de mesurer le temps mis par le kart entre les 2 capteurs.

Ce qui donne TIMSK = 0x02. Cependant, comme nous avons déjà l'autorisation de la routine d'interruption du premier timer pour la mesure du temps de parcours qui est active sur le bit 4, alors si on ajoute les 2 autorisations on obtient : TIMSK = 0x12.

Ce qui donne en C la configuration suivante:

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Normal top=FFh
// OC0 output: Toggle on compare match
TCCR0=0x08;
TCNT0=0x00;
OCR0=0xFA;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x12;
```

Illustration 8: Configuration du timer en C[4]

3. Programmation

3.1. Présentation du microcontrôleur AtMéga8535

Le microcontrôleur AtMéga8535 est un circuit programmable produit par ATMEL[2]. Ce microcontrôleur de type 8 bits, intègre de nombreux périphériques, ainsi que différents types de mémoire. Nous utiliserons le microcontrôleur de type DIL[3] dont le brochage est le suivant:

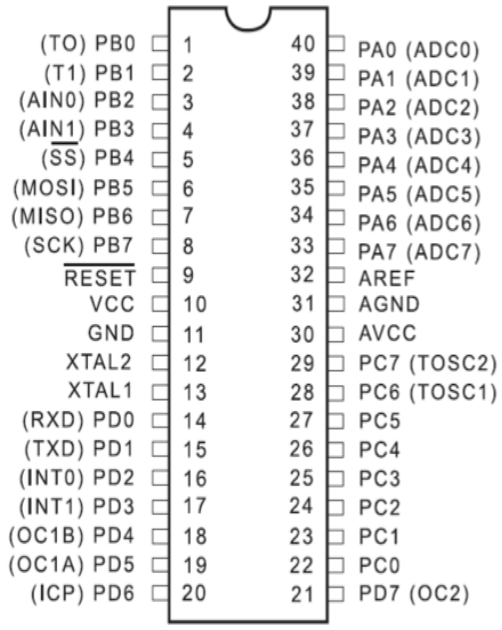


Illustration 9: Brochage de l'AtMéga8535[2]

Il est constitué des fonctions suivantes:

- CPU 8 Bits capable d'exécuter une instruction par cycle d'horloge
- 8 Ko de mémoire programme EEPROM FLASH[4] programmable
- 512 octets d' EEPROM
- 512 octets de RAM statique
- Convertisseur Analogique Numérique 10 bits à 8 entrées multiplexées
- Liaisons séries synchrone (SPI) et asynchrone (SCI)
- 2 TIMERS 8 bits
- 1 TIMER 16 bits
- 1 Comparateur de tensions analogiques
- 2 entrées d'interruptions externes et une entrée de RESET
- 4 Ports d'entrées/sorties 8 bits

2 ATMEL : Fabricant mondial de composants à semi-conducteur.
3 DIL : Dual In Line.
4 EEPROM FLASH : type de mémoire morte qui permet la programmation de plusieurs espaces mémoires simultanément.

Ce microcontrôleur possède 4 ports d'Entrée/Sortie (A, B, C et D) numériques de 8 bits chacun. Chaque bit de ces 4 ports peut être configuré soit en entrée soit en sortie, et la configuration des registres internes qui leur sont associés permet de mettre en œuvre certaines fonctions secondaires propres à un bit. Dans notre projet par exemple, on utilise les fonctions secondaires RxD et TxD associées respectivement aux bits PD0 et PD1.

Le microcontrôleur AtMéga8535 possède 3 timers (0, 1 et 2). Les timers 0 et 2 sont des compteurs 8 bits pouvant compter de 0 à 255 alors que le timer 1 est un compteur 16 bits comptant de 0 à 65 535. Nous utiliserons donc le timer 1 car nous avons besoin de compter pendant 60 secondes au centième de seconde près, soit de 0 à 6 000 centièmes de seconde.

Il faudra aussi régler et configurer les registres de comparaison du timer ainsi que sa fréquence, définie à partir d'une horloge interne ou externe au timer.

3.2. Programmation de la mesure de vitesse

3.2.1. La fonction d'interruption du timer

La fonction d'interruption du timer est la fonction qui s'exécutera tout les millièmes de seconde grâce à la configuration que nous avons effectué. Étant donné que cette fonction doit nous permettre de connaître le temps que met le kart pour aller du premier capteur au deuxième capteur de la borne d'arrivée, nous avons repris la structure de la fonction que nous avions réalisé au cours du semestre précédent et qui nous permettait de connaître le temps de parcours du kart sur le 50 mètres.

Cependant cette fonction dépendait d'un autre timer et s'exécutait tout les centièmes de seconde, il nous a donc fallu l'adapter à notre situation. Pour ce faire, le minimum est devenu 1 milliseconde au lieu de 1 centième de seconde et le maximum 10 secondes au lieu de 1 minute.

La fonction que le timer exécutera tout les millièmes de seconde est la suivante:

```
int secondeV, dixiemeV, centiemeV, milliemeV;

interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
	milliemeV++;
	if( milliemeV>9)
	{
		centiemeV++;
		milliemeV = 0;
		if( centiemeV>9)
		{
			dixiemeV++;
			centiemeV=0;
			if(dixiemeV>9)
			{
				secondeV++;
				dixiemeV=0;
				if(secondeV>9)
				{
					secondeV= 0;
				}
			}
		}
	}
}
```

Illustration 10: Fonction d'interruption du timer 0[4]

Grâce à cette fonction nous pouvons donc récupérer le temps mis par le kart entre les 2 capteurs de la borne d'arrivée.

L'autorisation, ou non, du fonctionnement du timer sera programmé dans la fonction principale. Nous autoriserons le timer, et donc l'exécution de sa fonction d'interruption tout les millièmes de seconde, lorsque le kart passera devant le premier capteur en écrivant TCCR0 = 0x0B. Puis lorsque le kart passera devant le deuxième capteur nous n'autoriserons plus le timer à fonctionner en écrivant TCCR0 = 0x08.

Ce qui s'écrira dans notre programme de la façon suivante:

```
if( CaptA == 1)                 //Passage devant le premier capteur
{
	TCCR0=0x0B;             //Autorisation
}
else if( CaptA == 0)            //Passage devant le deuxième capteur
{
	TCCR0=0x08;             //Arrêt
}
```

Illustration 11: Autorisation du timer[4]

Il ne nous reste plus maintenant qu'à réaliser le calcul de la vitesse et à l'afficher, ce que nous faisons dans la fonction principale.

3.2.2. Calcul et affichage de la vitesse

Le calcul et l'affichage de la vitesse se feront à partir du moment où le kart aura franchi le deuxième capteur, ce qui correspond dans notre fonction principale au « else if » cité dans l'illustration 10. Ce qui donnera:

```
else if( CaptA == 0)
{
        TCCR0=0x08;
        sprintf(tampon,"%d.%d%d%d", secondeV, dixiemeV, centiemeV, milliemeV);
        Temps = atof(tampon);
        vitesse=1.44/ Temps;
        EntiereV=(int)vitesse;
        DecimalV=(int)vitesse%10;
        UniteV = EntiereV - ((EntiereV / 10) * 10);
        DizaineV = EntiereV / 10 - ((EntiereV / 100) * 10);
        CentaineV = EntiereV/100;
        for(i=0;i<5;i++)
        {
                Afficheur(0,CentaineV,0);
                Afficheur(1,DizaineV,0);
                Afficheur(2,UniteV,1);
                Afficheur(3,DecimalV,0);
                delay_ms(2000);
                Afficheur(0,dizaine,0);
                Afficheur(1,unite,1);
                Afficheur(2,dizieme,0);
                Afficheur(3,centieme,0);
                delay_ms(2000);
        }
        flag2=0;
}
```

Illustration 12: Calcul et affichage de la vitesse[4]

Pour calculer la vitesse, nous commençons par créer une chaîne de caractères « tampon » qui contiendra le temps mis par le kart entre les 2 capteurs, en secondes et précis au millième de seconde, grâce à la fonction « sprintf ». Puis nous transformons cette chaîne en un réel à l'aide de la fonction « atof ». Pour calculer la vitesse, nous divisons 1,44 par le temps trouvé comme expliqué dans le principe de la mesure de la mesure.

Nous obtenons donc un réel composé de 4 chiffres, contenant une partie entière et une partie décimale. Pour afficher cette vitesse, il nous faudra donc séparer chaque chiffre pour les afficher sur un digit. Nous avons fais le choix d'afficher un nombre avec une partie entière de 3 chiffres et une partie décimale d'un chiffre.

Pour récupérer la partie entière de ce nombre, il suffit de transformer la variable qui était un réel en un entier, ce qui ce fait grâce à l'opérateur « int ». Pour la partie décimale, il faut prendre le modulo 10 du nombre après l'avoir transformé en entier, d'où : DecimalV=(int)vitesse%10.

Grâce à des formules mathématiques, nous pouvons ensuite récupérer un à un la centaine, la dizaine et l'unité de la partie entière du nombre. La partie décimale ne contenant qu'un chiffre peut être directement affichée.

Enfin, pour réaliser l'alternance souhaité de l'affichage entre le temps et la vitesse, nous avons créé une boucle « for » qui se répètera 5 fois et dans laquelle s'affichera d'abord la vitesse pendant 2 secondes puis le temps pendant 2 secondes. A la fin de cette alternance, l'affichage se figera sur le temps du kart jusqu'à ce qu'un nouveau kart se présente au départ.

4. Améliorations du projet

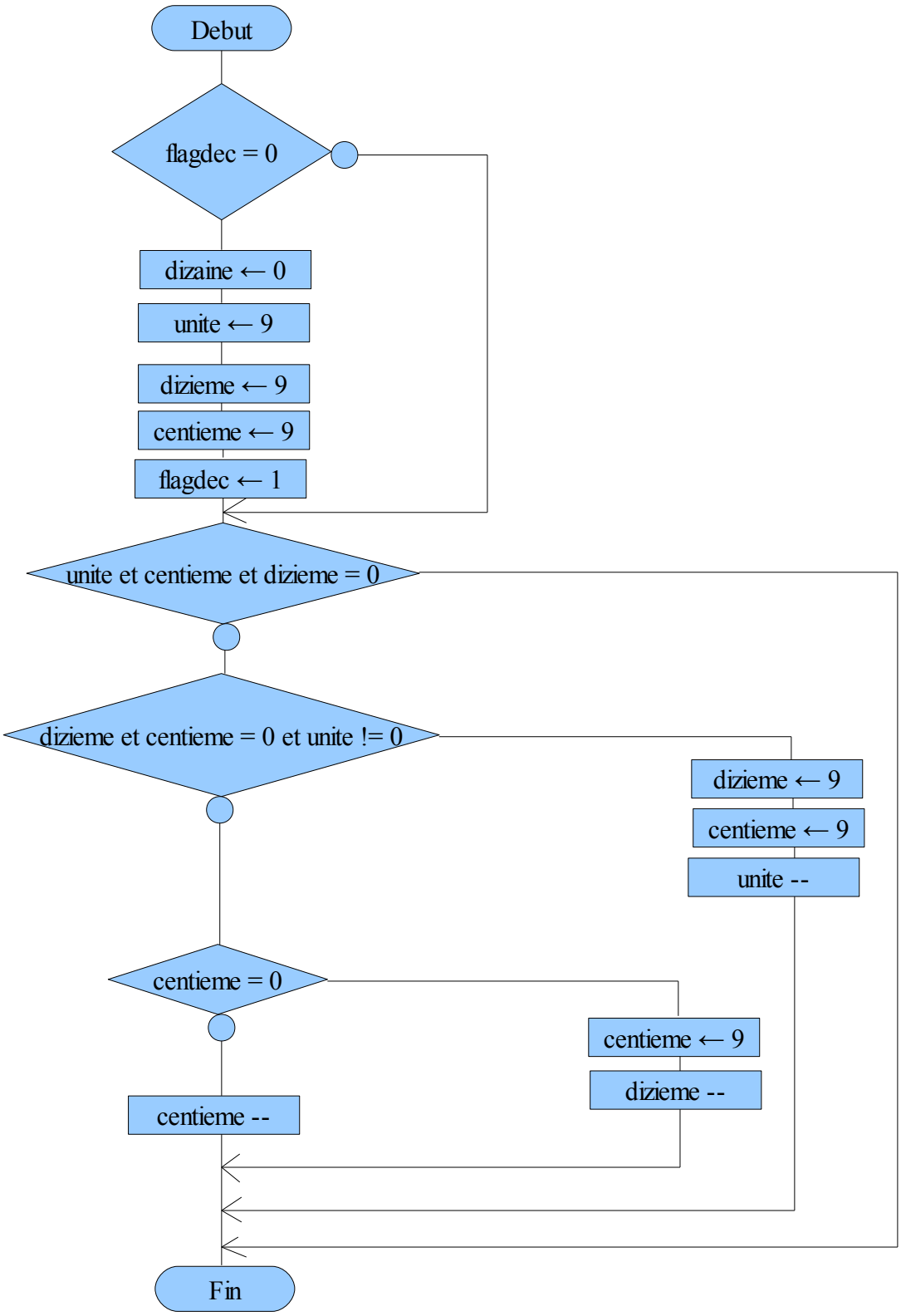
Comme nous avons répondu à l'ensemble du cahier des charges de notre projet en moins de temps que prévu, nous avons décidé d'utiliser ce temps supplémentaire pour apporter différentes améliorations au projet.

4.1. Attente de 10 secondes au départ

Le protocole retenu pour le départ d'un kart sur l'épreuve du 50 mètres départ arrêté est qu'il doit attendre 10 secondes devant la borne de départ avant de pouvoir s'élancer. Lors de notre projet du semestre 3, nous avions réalisé un compteur qui allait de 0 à 10 secondes pour afficher cette attente.

Cependant nous nous sommes rendus compte qu'il était plus judicieux de faire un affichage de 10 à 0 secondes pour marquer le départ.

Ainsi l'ordinogramme de la fonction réalisant ce décompte serait le suivant.



Comme notre timer, qui s'exécute toutes les 10ms, nous sert à la fois à mesurer le temps de parcours du kart sur le 50 mètres et à effectuer le décompte de 10 à 0 secondes, on a dû créer une variable drapeau qui nous permet, lorsque l'on rentre dans la routine d'interruption du timer, de connaître l'action qu'on souhaite exécuter.

Si la variable drapeau est à 1 on effectue le décompte de 10 à 0 secondes et si elle est à 0 alors on effectue le chronométrage normal du kart sur le 50 mètres.

La fonction correspondant à l'ordinogramme ci-dessus est la suivante:

```
if( flagdec == 0)
{
        dizaine=0;
        unite=9;
        dizieme=9;
        centieme=9;
        flagdec=1;
}
if( unite == 0 && dizieme == 0 && centieme == 0)
{
}
else if( dizieme == 0 && centieme == 0 && unite !=0)
{
        dizieme=9;
        centieme=9;
        unite--;
}
else if( centieme == 0)
{
        centieme=9;
        dizieme--;
}
else
        centieme--;
```

Illustration 13: Fonction de décompte de 10 à 0 secondes[4]

4.2. Feux tricolores

Pour améliorer la visibilité de la procédure de départ, Monsieur LEQUEU nous a proposé de rajouter deux feux tricolores sur la borne de départ. Nous avons tout d'abord réalisé la partie mécanique, en fixant les ampoules sur le support. Puis nous avons effectué la partie électrique en câblant les ampoules à l'aide de 2 fils, un relié à l'alimentation qui se fait par le biais d'une carte connectée à notre microcontrôleur afin de pouvoir alimenter ou non l'ampoule. L'autre fil est connecté à la masse commune de toutes les ampoules.

Les feux réalisés seront connectés à la borne de départ de la même manière que l'afficheur est relié à la borne d'arrivée, à l'aide d'une nappe[5].

5 Nappe: câble plat utilisé en informatique pour relier des cartes électroniques.

Voici un exemple de nappe:

Illustration 14: Nappe ou câble plat

Ainsi, nous pouvons réutiliser la même fonction que celle utilisée pour l'afficheur afin de contrôler l'allumage des feux. Cette fonction est la suivante:

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

void afficheur1(unsigned char adresse, unsigned char caractere, unsigned char point)
{
        if ( point == 1 )
        {
                PORTA = (caractere | 0x01);      //Un point est rajouté après le caractère
        }
        else
        {
                PORTA=caractere;        //Un caractère tout seul
        }
        PORTB=(0b00010000|adresse_constant[adresse & 0x0F]);
        PORTB.4=1;
        PORTB.4=0;
        PORTB.4=1;
        PORTA=0x00;
}
```

Illustration 15: Fonction Afficheur utilisé pour les feux[1]

Dans la fonction principale, pour autoriser le contrôle des feux et donc leur allumage, il suffit d'écrire la ligne de code suivante: ENABLE=0. De même, pour ne pas l'autoriser et donc les éteindre il faut écrire dans notre programme: ENABLE=1.

Pour pouvoir allumer un feu la ligne de code est la suivante: afficheur1(0, 0x.. ,0).

Le deuxième paramètre passé à cette fonction « 0x.. » est le paramètre qui nous permet de choisir le feu ou les feux que l'on souhaite allumer. Après avoir étudié la documentation technique de la carte reliant les feux au microcontrôleur[6], voici un tableau résumant le code à passer en paramètre pour chaque cas de figure:

6 Lien pour retrouver cette documentation technique: http://www.thierry-lequeu.fr/data/DISPLAY2.pdf

Feu(x) à allumer	*Code*
Rouge Gauche	0x01
Orange Gauche	0x02
Vert Gauche	0x04
Rouge Droite	0x08
Orange Droite	0x10
Vert Droite	0x20
Les 2 Rouges	0x09
Les 2 Oranges	0x12
Les 2 Rouges	0x24
Tous les feux	0xFF

Il nous a ensuite fallu établir un protocole, en accord avec Monsieur LEQUEU, pour le contrôle des feux. Voici le protocole qui a été retenu:

- En attente d'un kart devant la borne de départ : feux oranges clignotants
- Attente des 10 secondes pour le kart devant la borne avant le départ: feux rouges fixes et oranges clignotants
- Fin des 10 secondes, départ du kart possible : feux verts fixes
- Juste après le départ du kart et tant qu'il n'a pas fini l'épreuve ou en cas de faux départ: feux rouges clignotants
- Arrivée du kart, alternance de l'affichage du temps et de la vitesse: feux rouges clignotants mais à intervalle différent que précédemment
- Fin de l'alternance, affichage du temps fixe et attente d'un nouveau kart: feux oranges clignotants

Voici une photo des 2 feux tricolores lorsqu'ils sont tous allumés :

Illustration 16: Feux tricolores pour la borne de départ[4]

4.3. Cartes RS232

Au cours du semestre 3, nous avions pu constater que les cartes RS232 déjà réalisées avaient un défaut dans leur typon, car certaines connexions avaient été inversées. Pour résoudre ce problème nous avions, par manque de temps, coupé au cutter les pistes du circuit imprimé qui étaient mal reliées puis nous avions effectués des vias[7] afin de croiser les 2 pistes et ainsi rétablir une connexion correcte.

Nous avons donc décidé au cours de ce projet de refaire de nouvelles cartes avec une connexion correcte. Grâce à Monsieur LEQUEU, nous avons été beaucoup plus rapide que prévu car il a lui même réalisé le typon, la gravure et le perçage des cartes. Il nous restait juste à placer et souder les différents composant sur les cartes.

Nous n'avons pas encore terminé de réaliser ces cartes, nous effectuerons donc leurs tests lors de la dernière séance.

7 Vias: fil souple connecté entre 2 points du circuit imprimé.

Conclusion

Ce projet nous a permis de mettre en application toutes nos connaissances en informatique, ainsi qu'une partie de nos connaissances mécaniques et électriques lors de la réalisation des feux. Nous avons aussi appris à étudier des documentations techniques fournies par le constructeur entièrement en anglais lors de l'utilisation et de la configuration du timer, ce qui demandait un bon niveau en anglais technique et une connaissance poussée du microcontrôleur AtMéga8535 et de ses fonctionnalités.

Pour la programmation du microcontrôleur, nous avons utilisé le logiciel CodeVisionAVR que nous avions déjà utilisé durant le semestre 3. Nous avons ainsi eu une certaine aisance car nous connaissons bien son interface, ce qui nous a permis de résoudre rapidement les différents problèmes rencontrés.

Nous avons donc réussi la partie principale de notre projet, et ce en moins de temps que prévu. Cela nous a donc permis d'apporter quelques améliorations supplémentaires au projet entier, tels que 2 feux tricolores présents au départ du 50 mètres et un décompte de 10 à 0 secondes lors du protocole de départ. Pour la dernière semaine, il ne nous reste plus qu'à réaliser les cartes RS232 et à les tester en les intégrant dans notre projet.

Résumé

Dans le cadre du semestre 4, nous avons fait le choix du projet «Balise de mesure de vitesse pour l'épreuve de 50m départ arrêté» en Études et Réalisation. Nous avons tous les 2 choisis ce projet car il était centré surtout autour de l'informatique, et le fait d'avoir un « réel » client, le challenge e-Kart, rajoutait encore plus de réalité et d'envie de mener à bien notre projet afin de le voir fonctionnait en situation réelle durant le challenge.

De plus nous avions déjà réalisé au cours du semestre précédent un projet portant sur le même thème, et comme on avait vraiment aimé travailler sur ce projet et cette thématique du challenge e-Kart, nous avons logiquement décidé de poursuivre notre travail tout en restant dans ce même domaine.

Nous avons donc, en plus des balises de mesure de temps précédemment réalisées, rajouté des balises de mesure de la vitesse du kart lors de son passage devant la borne d'arrivée. Cela nous a donc amené sur l'étude d'un nouveau timer, du microcontrôleur AtMéga8535, et de sa configuration à effectuer, ce qui a été la plus longue partie de notre projet.

Nous avons terminé cette partie de mesure de la vitesse en avance sur notre planning, nous avons donc fait le choix d'utiliser ce temps supplémentaire afin d'ajouter encore quelques améliorations à notre projet, par la création et la programmation de deux feux tricolores ainsi qu'une procédure de décompte de 10 à 0 plutôt que d'aller de 0 à 10 secondes avant le départ du kart.

Finalement, notre dernière séance sera consacrée à la réalisation de nouvelles cartes RS232 et à leurs tests. Il ne nous restera plus désormais qu'a tester cette communication avec une liaison filaire sur plus de 50 mètres afin d'être certains qu'elle ne sera pas parasitée par l'environnement l'entourant sur cette distance, et qu'elle sera donc utilisable durant une épreuve 50 mètres départ arrêté.

317 mots

Index des illustrations

Bibliographie

[1] **LEQUEU Thierry,** *La documentation de Thierry sur OVH,* (page consulte e en 2011) < http://thierry-lequeu.fr/ >

[2]**LEQUEU Thierry**, *Documentation technique ATMEL,* (page consultée en 2012) <http://www.thierry-lequeu.fr/data/AT-MEGA-8535L.pdf>

[3] **BESSE Aure lie, POTELLE Je ro me,** *Balise de chronome trage pour l'e preuve de 50m de part arre te,* 2007/2009

[4] **Les Auteurs,** 2011/2012

Annexes

Annexe n°1 : Programme complet du microcontrôleur relié à la borne de départ

```
unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & 0x80) ) // Test de RXC bit7
;
/* Get and return received data from buffer */
return UDR;
}

void afficheur1(unsigned char adresse,unsigned char caractere, unsigned char point)
{
 if (point == 1) {
     PORTA=(caractere | 0x01); // Un caractère avec le point !
    }
    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;
}

void Feu( int flag)
{
        if( flag == 1)
        {
                afficheur1(0,0x12,0);
                delay_ms(500);                  //Feux oranges clignotants
                afficheur1(0,0x00,0);
                delay_ms(500);
        }
        if( flag == 2)
        {
                afficheur1(0,0x00,0);
                delay_ms(500);                  //Feux rouges clignotants
                afficheur1(0,0x09,0);
                delay_ms(500);
        }
}
void main(void)
{
// 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=In Func3=In Func2=In Func1=Out Func0=In
// State7=1 State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x80;
DDRD=0x82;

// 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
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E;
OCR1AL=0x20;
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;

// 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;

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

// LCD module initialization

while (1)
      {
      int i,flag;
       ENABLE=0;
       if( car == 'M')
            carRecu=USART_Receive();
       if(flag!=2 || carRecu == 'F' || flag == 1)
            flag=1;                                          // En attente qu'un kart se presente
            if( CaptD == 0 && CaptA == 1)
            {
                        car = 'D';
                        USART_Transmit(car);
                        carRecu=USART_Receive();
                        if(carRecu == 'C')
                        {
                        for(i=0;i<10;i++)
                        {
                                    if( CaptD == 0 && CaptA == 1)
                                    {
                                                car = 'P';
                                                USART_Transmit(car);
                                                afficheur1(0,0x09,0);
                                                delay_ms(500);                  //Procédure des 10s d'attente
                                                afficheur1(0,0x1B,0);           //Rouges fixes et oranges clignotants
                                                delay_ms(500);
                                    }
                                    else
                                    {
                                                car = 'A';
                                                USART_Transmit(car);
                                                i=10;                           //Faux départ
                                                flag=2;
                                    }
```

```
                    }
                }
            }
            if( car == 'P' && CaptD == 0 && CaptA == 1)
            {
                car = 'R';
                afficheur1(0,0x24,0);                    //Kart autorisé a partir, feux verts fixes
                flag=2;
                USART_Transmit(car);
                do
                {
                    if( CaptA == 0)
                    {
                        car = 'M';
                        USART_Transmit(car);
                    }
                }
                while( CaptD == 0 && car != 'M');
            }
            if( car == 'M')
            {
                do
                {
                        car='x';                                    //Kart en course , feux rouges clignotants
                        Feu(flag);
                        carRecu=USART_Receive();
                }
                while( carRecu == 'C' );
                flag=1;
            }
            Feu(flag);
        }

}
```

Annexe n°2 : Programme complet du microcontrôleur relié à la borne d'arrivée

```
/*****************************************************
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 :
Version :
Date    : 03/10/2011
Author  : F4CG
Company : F4CG
Comments:


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

#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#include <stdlib.h>
#include <math.h>

#define Vert PIND.6
#define Rouge PIND.7
#define Bleu PINB.5
#define CaptD PIND.4
#define CaptA PIND.5
#define ENABLE PORTD.7

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

// Declare your global variables here
unsigned char car, CarEmis;

int flagdec,drapeau,dizaine, unite, dizieme, centieme, secondeV, dixiemeV, centiemeV, milliemeV;

char tampon[16],TamponV[16];
```

```
flash const unsigned char valeur_constant[]=
{
0xEE,0x82,0xDC,0xD6,0xB2,0x76,0x7E,0xC2,0xFE,0xF6,
0xFA,0xFE,0x6C,0xEE,0x7C,0xF8,0x7E,0xBA,0x82,0x86,
0x3C,0x2C,0xEA,0x1A,0xEE,0xF8,0xF2,0xFA,0x76,0x3C,
0x0E,0xAE,0xDC};

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

void Afficheur(unsigned char adresse, unsigned char caractere, unsigned char point)
{                                               // Fonction d'affichage classique( temps, chrono, vitesse)
        if ( point == 1 )
        {
                PORTA = (valeur_constant[caractere] | 0x01);  //Un point est rajouté après le caractère
        }
        else
        {
                PORTA=valeur_constant[caractere]; //Un caractere tout seul
        }
        PORTB=(0b00010000|adresse_constant[adresse & 0x0F]);
        PORTB.4=1;
        PORTB.4=0;
        PORTB.4=1;
        PORTA=0x00;
}

void Afficheur2(unsigned char adresse, unsigned char caractere, unsigned char point)
{                                                       //Fonction d'affichage du message F A U H
        if ( point == 1 )
        {
                PORTA = (caractere | 0x01);             //Un point est rajouté après le caractère
        }
        else
        {
                PORTA=caractere;                        //Un caractere tout seul
        }
        PORTB=(0b00010000|adresse_constant[adresse & 0x0F]);
        PORTB.4=1;
        PORTB.4=0;
        PORTB.4=1;
        PORTA=0x00;
}

unsigned char USART_Receive( void )                     // Fonction de rececption de caractère RS232
{
/* Wait for data to be received */
while ( !(UCSRA & 0x80) ) // Test de RXC bit7
;
/* Get and return received data from buffer */
return UDR;
}
```

```
void USART_Transmit( unsigned char data )          // Fonction de transmission de caractère RS232
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (0x20)) ) // Test de UDRE bit 5
;
/* Put data into buffer, sends the data */
UDR = data;
}

interrupt [TIM1_COMPA] void timer1_compa_isr(void)          //Fonction d'interruption du Timer 1
{
      if(drapeau == 1)
          {
               if( flagdec == 0)
               {
                        dizaine=0;
                        unite=9;
                        dizieme=9;
                        centieme=9;
                        flagdec=1;
               }
              if( unite == 0 && dizieme == 0 && centieme == 0)
               {                                                    //Fonction de décompte au départ : 10s à 0
               }
               else if( dizieme == 0 && centieme == 0 && unite !=0)
               {
                           dizieme=9;
                           centieme=9;
                           unite--;
               }
               else if( centieme == 0)
               {
                        centieme=9;
                        dizieme--;
               }
               else
                     centieme--;
          }
          else
          {
           centieme++;
           if( centieme>9)
           {
                 dizieme++;
                 centieme = 0;
                 if( dizieme>9)                          //Fonction réalisant le chronométrage du temps
                 {
                       unite++;
                       dizieme=0;
                       if(unite>9)
                       {
                            dizaine++;
                            unite=0;
                            if(dizaine == 6)
                            {
                                  dizaine = 0;
                            }
```

```
                        }
                    }
                }
        }
        Afficheur(0,dizaine,0);
        Afficheur(1,unite,1);
        Afficheur(2,dizieme,0);
        Afficheur(3,centieme,0);
}

interrupt [TIM0_COMP] void timer0_comp_isr(void)                    //Fonction d'interruption du Timer0
{
        milliemeV++;
        if( milliemeV>9)
        {
                centiemeV++;
                milliemeV = 0;
                if( centiemeV>9)
                {                                                   //Fonction permettant le calcul du temps effectué par le kart
                        dixiemeV++;                                 //pour en déduire la vitesse
                        centiemeV=0;
                        if(dixiemeV>9)
                        {
                                secondeV++;
                                dixiemeV=0;
                                if(secondeV>9)
                                {
                                        secondeV= 0;
                                }
                        }
                }
        }
}

void Chronometre(int Flag )
{
        if( Flag == 1)
        {
                ENABLE=0;                                           //Autorisation de l'affichage
                TCCR1B = 0x0A;                                      //Autorisation du Timer
        }
        else if( Flag == 0)
        {
                        TCCR1B = 0x08;                              //Arrêt du Timer
                        ENABLE = 0;                                 //Affichage du message:
                        Afficheur2(0,0x78,1);                               //F
                        Afficheur2(1,0xFB,1);                               //A
                        Afficheur2(2,0xAE,1);                               //U
                        Afficheur2(3,0xBA,1);                               //H
        }
```

```
        else if (Flag == 2)
        {
                TCCR1B = 0x08;
                ENABLE = 0;
                dizaine = 0;                            //Remise a zéro de l'affichage
                unite = 0;
                dizieme = 0;
                centieme = 0;
                Afficheur(0,dizaine,0);
                 Afficheur(1,unite,1);
                 Afficheur(2,dizieme,0);
                Afficheur(3,centieme,0);
        }
        else if(Flag == 4)
        {
                TCCR1B = 0x08;
                ENABLE = 0;
                dizaine = 1;
                unite = 0;    //Mise a 10.00 de l'affichage avant la procédure de décompte pour  le départ
                dizieme = 0;
                centieme = 0;
                Afficheur(0,dizaine,0);
                Afficheur(1,unite,1);
                 Afficheur(2,dizieme,0);
                 Afficheur(3,centieme,0);
      }
}

void main(void)
{
// 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=In Func3=In Func2=In Func1=Out Func0=In
// State7=1 State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x80;
DDRD=0xC2;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Normal top=FFh
// OC0 output: Toggle on compare match
TCCR0=0x08;
TCNT0=0x00;
OCR0=0xFA;

// 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
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E;
OCR1AL=0x20;
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=0x12;

// 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;
```

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

// LCD module initialization
lcd_init(16);

#asm("sei")

   while (1)
   {

      int flag,DecimalV,EntiereV,CentaineV,UniteV,DizaineV,i,flag2;
      float vitesse, Temps;
          while( CaptD == 1)
          {
                    secondeV=0;
                    dixiemeV=0;
                    centiemeV=0;
                    milliemeV=0;
                    if( flag2 != 1)
                    {
                              car=USART_Receive();        //Appel de la fonction de reception de caractère
                    }
                    if (car == 'R')                     //Remise a zéro du chronomètre
                    {
                          flag = 2;
                          drapeau=0;
                          flagdec=0;
                     }                                  //Départ du kart sur l'épreuve
                    if (car == 'M')
                    {
                          flag = 1;
                          flag2=1;
                    }
                    if (car == 'A')                     //Faux départ
                    {
                          flag = 0;
                          flagdec=0;
                    }
                    if(car == 'D')                      //Kart se présentant au départ
                     {
                           flag=4;
                           CarEmis = 'C';
                            USART_Transmit(CarEmis);
                     }
                    if(car == 'P')                      //Procédure d'attente des 10s au départ
                    {
                          drapeau=1;
                          flag=1;
                      }
```

```
                Chronometre(flag);
                CarEmis = 'C';
                USART_Transmit(CarEmis);
                lcd_gotoxy(0,0);
                    sprintf(TamponV," %c  %c", CarEmis, car);
                    lcd_puts(TamponV);
        }
        TCCR1B = 0x08;
        ENABLE=0;
        if( CaptA == 1)
        {
                TCCR0=0x0B;
        }
        else if( CaptA == 0)
        {
                TCCR0=0x08;
                sprintf(tampon,"%d.%d%d%d", secondeV, dixiemeV, centiemeV, milliemeV);
                Temps = atof(tampon);
                vitesse=1.44/ Temps;
                EntiereV=(int)vitesse;
                DecimalV=(int)vitesse%10;                       //Calcul de la vitesse du kart
                lcd_gotoxy(0,2);
                lcd_puts(tampon);
                UniteV = EntiereV - ((EntiereV / 10) * 10);
                DizaineV = EntiereV / 10 - ((EntiereV / 100) * 10);
                CentaineV = EntiereV/100;
                for(i=0;i<5;i++)                        //Affichage de l'alternance Temps-Vitesse 5 fois
                {                                       //avec une alternance toutes les 2s
                        Afficheur(0,CentaineV,0);
                        Afficheur(1,DizaineV,0);
                        Afficheur(2,UniteV,1);
                        Afficheur(3,DecimalV,0);
                        CarEmis = 'C';
                        USART_Transmit(CarEmis);
                        delay_ms(2000);
                        Afficheur(0,dizaine,0);
                        Afficheur(1,unite,1);
                        Afficheur(2,dizieme,0);
                        Afficheur(3,centieme,0);
                        delay_ms(2000);
                        CarEmis = 'C';
                        USART_Transmit(CarEmis);
                }
                CarEmis = 'F';
                USART_Transmit(CarEmis);
                flag2=0;
                lcd_gotoxy(0,0);
                sprintf(TamponV," %c  %c", CarEmis, car);
                lcd_puts(TamponV);
        }
    }
}
```