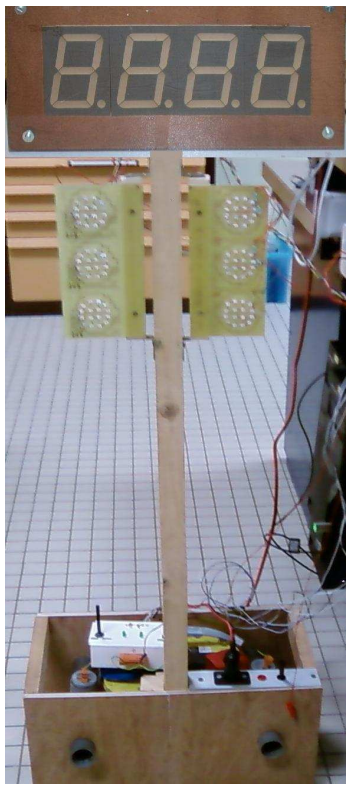
Université François Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle



Étude et Réalisation 2ème année
Balise de chronométrage pour l'épreuve de 50 m départ arrêté

Aurélie BESSE
Jérôme POTELLE
Groupe Q2
2007/2009

Enseignant
Thierry LEQUEU
Bernard GLIKSOHN

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

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

Étude et Réalisation 2ème année

Balise de chronométrage pour l'épreuve de 50 m départ arrêté

Aurélie BESSE
Jérôme POTELLE
Groupe Q2
2007/2009

Enseignant
Thierry LEQUEU
Bernard GLIKSOHN

Sommaire

Introduction....5

1. Présentation du dispositif de chronométrage....6

1.1. L'épreuve 50 mètres départ arrêté....6

1.2. Cahier des charges....6

1.3. Description de la borne de départ....9

2. Le microcontrôleur AT8535....12

2.1. Généralités....12

2.2. Les E/S numériques....13

2.3. Les timers....13

3. Programmation....14

3.1. Configuration des ports d'E/S....14

3.2. Configuration du Timer....16

3.3. Programme principal....19

Conclusion....23

Résumé....24

Index des illustrations....25

Bibliographie....26

Annexes....27

Introduction

Notre travail d'étude et réalisation du semestre 4 consiste à développer la partie technique de chronométrage d'une épreuve de 50 mètres départ arrêté d'une course de karts électriques. Deux binômes ont travaillé sur ce projet : l'un à l'étude de la borne de départ (notre projet), le second étant chargé de celle d'arrivée.

Après un descriptif du dispositif mis en place, nous présenterons le microcontrôleur ATmega 8535 puis l'étude du programme de chronométrage.

1. Présentation du dispositif de chronométrage

1.1. L'épreuve 50 mètres départ arrêté

Cette épreuve se déroule sur 50 mètres départ arrêté avec un ou deux karts. Le but est de parcourir cette distance en un minimum de temps et de couper les faisceaux de la borne d'arrivée le plus rapidement possible. Afin d'optimiser le chronométrage de cette épreuve, il a été convenu d'installer une borne de départ et une borne d'arrivée.

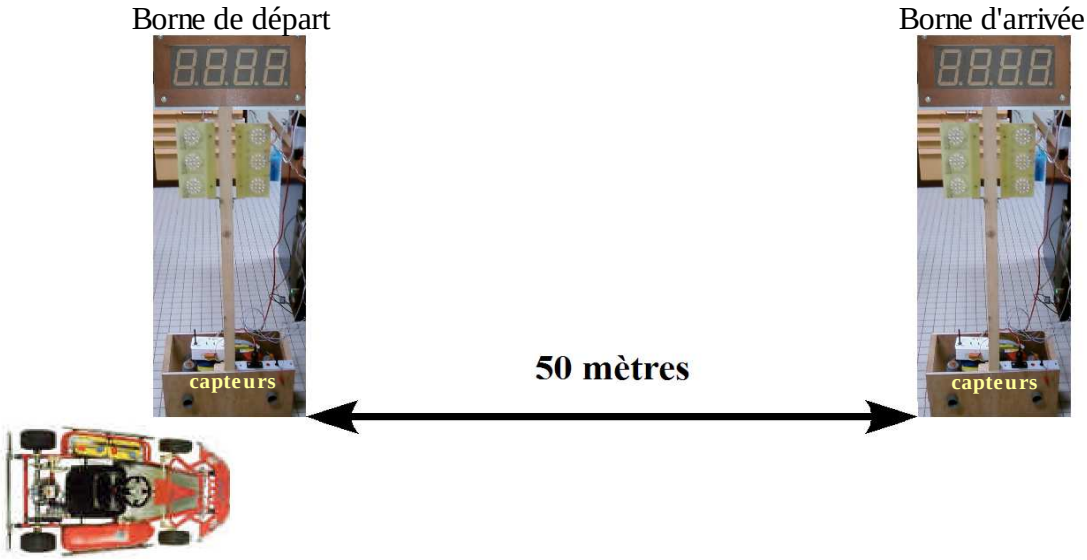


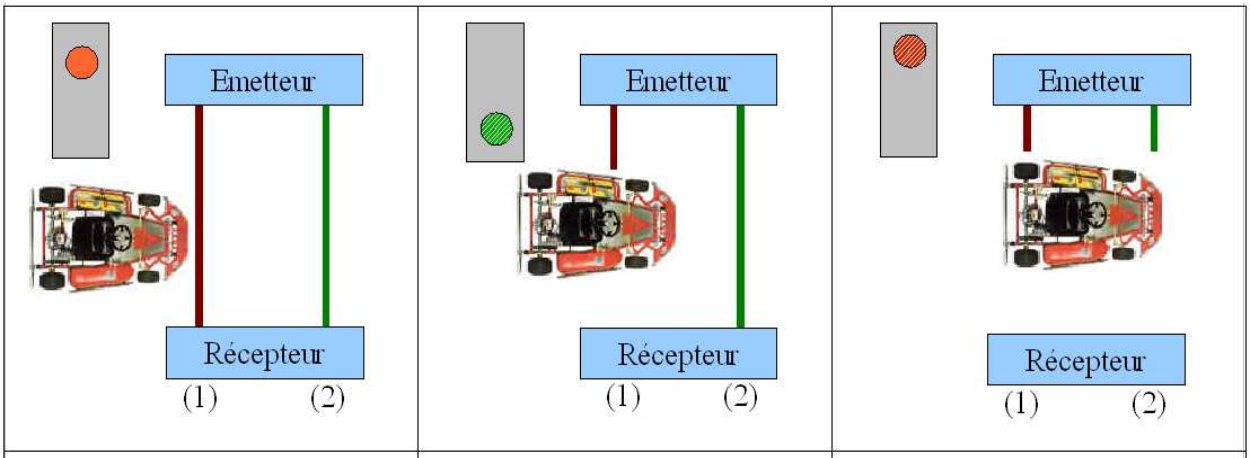
Illustration 1: Présentation de l'épreuve

1.2. Cahier des charges

Pour faciliter le travail et le réaliser en un minimum de temps, nous serons deux binômes à travailler sur ce projet. Ce dossier ne traitera donc que la borne de départ.

Illustration 2: Synoptique de premier niveau

La fonction première de cette borne est de détecter le départ des karts. Il y a donc un organe qui détecte le passage du kart appelé « faisceau de détection ». On peut s'en servir aussi pour positionner les karts. À l'aide de 2 faisceaux de détection (1) et (2), il est possible d'indiquer la position du kart sur la ligne de départ.



Cas N° 1 :	**Cas N° 2 :**	**Cas N° 3 :**
Il n'y a pas de kart au départ.	Le kart est en position correct de départ.	Le kart est en position de faux départ.
Le feu de signalisation est en position fixe orange.	Le feu de signalisation clignote en vert	Le feu de signalisation clignote en rouge.

Illustration 3: Schéma de principe de la mesure du temps pour l'épreuve de 50 mètres départ arrêté

La borne de départ dispose d'un système de signalisation (feux rouge, orange et vert) pour donner le départ qui doit être synchronisé et/ou commandé par le commissaire de piste qui tient le drapeau.

Le commissaire de départ peut garder le privilège du déclenchement du départ et donner l'autorisation du départ à l'aide de son drapeau. Dans ce cas le chronomètre devra se déclencher lors du passage du kart devant le capteur (2). Le système peut être également automatique à partir du moment où les karts sont en position correcte pendant un temps donné (30 secondes par exemple), la séquence de départ déroule alors les feux rouge, orange et vert. Les karts ont alors une durée donnée (10 secondes) pour démarrer. Le système se réinitialise si aucun départ n'est détecté au delà de 10 secondes.

Pour ce projet nous disposons d'une borne de départ comprenant :

- une carte microcontrôleur ATMega8535,
- 4 capteurs de gestion du départ,
- 4 lampes de gestion du départ,
- 1 feu de signalisation,
- un afficheur 4 digits à Leds géré par une liaison série.

Pour la transmission des informations entre la borne de départ et celle d'arrivée, nous utiliserons une liaison UHF (Ultra Haute Fréquence) à 433MHz qui pourra être remplacée par une liaison filaire en particulier pour la mise au point du programme.

Afin d'organiser correctement le travail dans le temps qui nous est imparti, nous avons effectué le planning suivant :

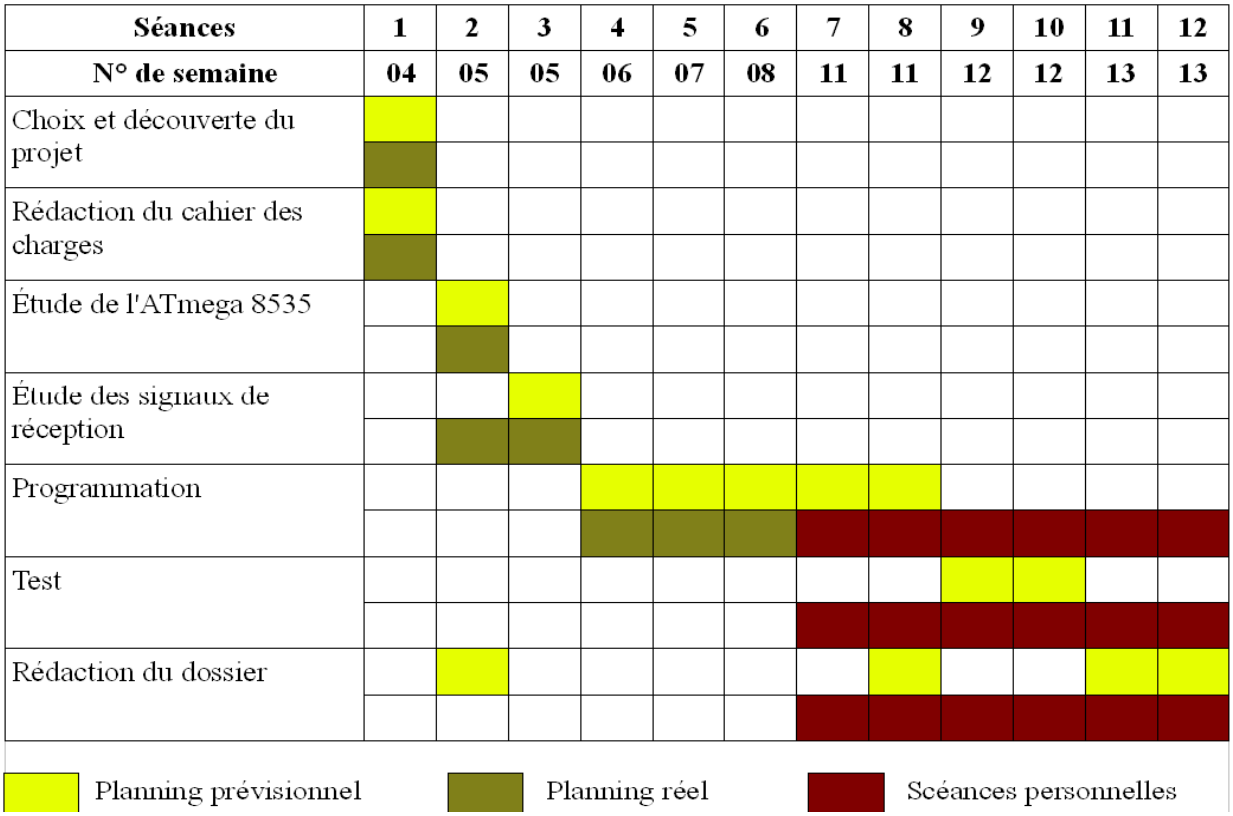


Illustration 4: Planning prévisionnel et réel

Nous remarquons sur l'illustration 4 que la première partie du planning réel est en accord avec celui prévisionnel malgré un travail supplémentaire a effectuer. En effet, après quelques séances nous nous sommes aperçu que la carte microcontrôleur ne fonctionnait pas correctement. Il a fallu, à l'aide des schémas électriques de la carte, percer et souder une nouvelle carte.

Ensuite, diverses pannes sont survenues au cours des séances notamment des pannes électriques (court-circuits, mauvais câblage, composants défectueux).

Un feu tricolore manquait à la borne de départ et il a donc fallu en réaliser un sachant que les schémas électriques étaient fournis. Nous avons dû refaire une grande partie des câbles ainsi que redimentionner la carte d'alimentation fournissant du 15V et du 30V à partir de la batterie 12V.

L'affichage fonctionnait à la fin de la 6ème séances mais au retour des vacances et avec le même programme, nous n'arrivions plus à le faire fonctionner. Cela nous a donc empêché de tester notre programme final. De plus des tests ont été effectués à l'extérieur et le système des lampes avec les capteurs ne fonctionnent pas.

Nous avons quand même pu réaliser la majeure partie du projet (programmation du microcontrôleur) malgré des mouvements de grèves les semaines 11 et 12 et avec un travail personnel supplémentaire en dehors des séances d'E&R.

Ce qui reste à faire :

- mettre au propre une partie du câblage,
- résoudre le problème de l'afficheur géant,
- faire fonctionner la liaison série UHF,
- réécrire un programme en tenant compte des différents modes et du nombre de karts,
- Revoir la sensibilité des capteurs.

1.3. Description de la borne de départ

La borne de départ est gérée par un microcontrôleur qui reçoit :

- les informations de départ des deux karts,
- les tops d'arrivée par liaison UHF.

Ce microcontrôleur pilote l'afficheur LCD[1], l'afficheur géant ainsi que les feux de départ.

L'illustration 5 présente les interconnexions des différents modules.

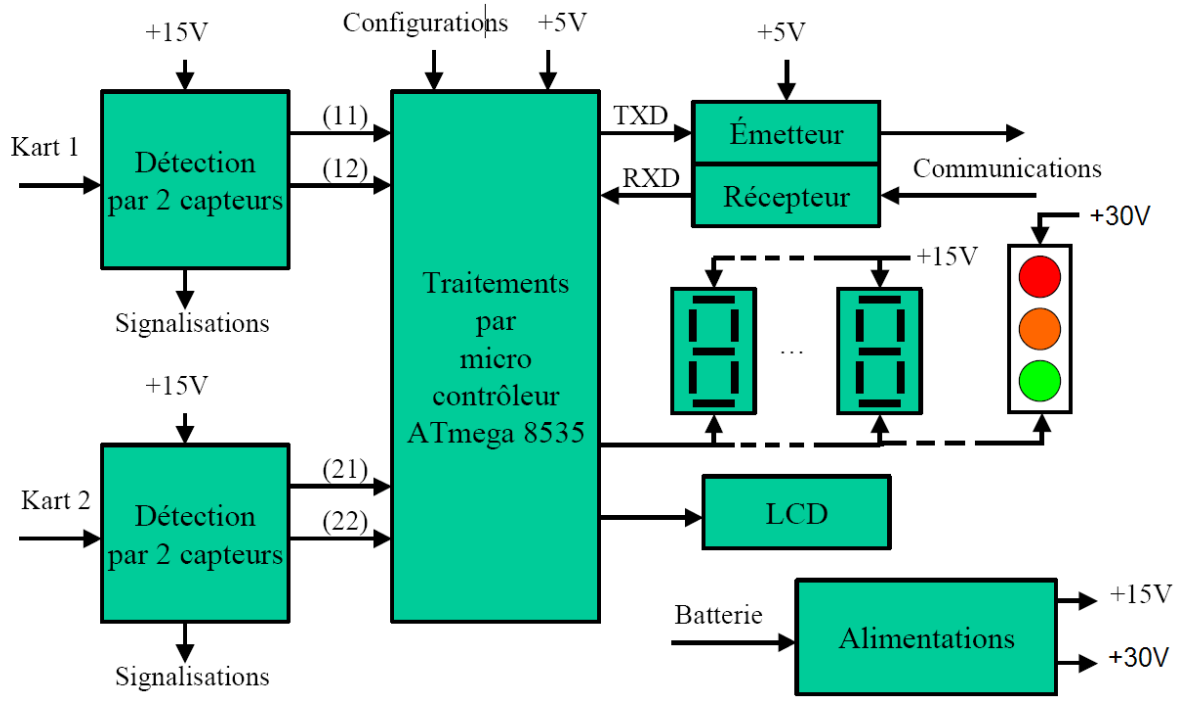


Illustration 5: Synoptique de niveau 3 de la borne [3]

1 LCD : Liquid Crystal Display

L'illustration ci-dessous montre la composition de la borne ainsi que la disposition des différents éléments.

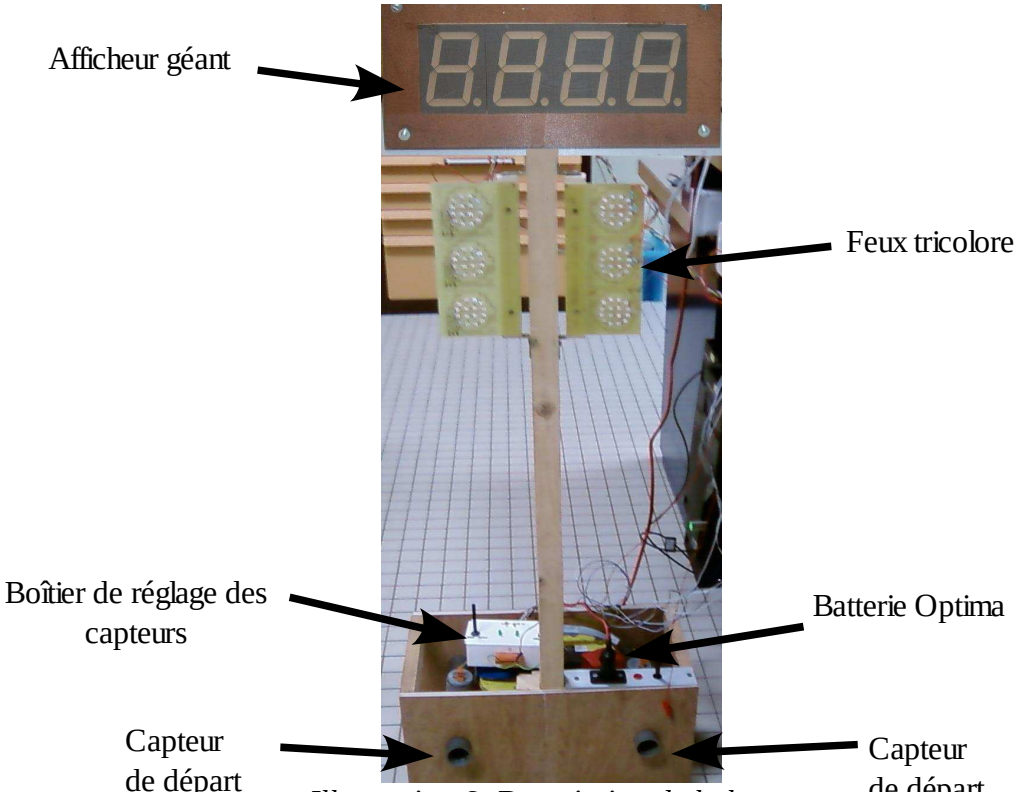


Illustration 6: Description de la borne

L'afficheur géant représente le chronomètre.

Les feux tricolore indiquent au pilote s'il est en bonne position ainsi que le départ.

Le boîtier électronique est composé de potentiomètres permettant de régler la sensibilité des capteurs en fonction de la luminosité ambiante. Les capteurs sont des photo-résistances.

La batterie Optima nous fournit l'énergie nécessaire l'autonomie de la borne.

L'illustration suivante montre les deux faces de la carte microcontrôleur. Les boutons poussoirs nous ont servi dans un premier temps pour simuler les capteurs. Nous en avions donc défini un pour simuler le start et l'autre pour le stop. Dans le programme final, ils serviront à sélectionner les différents modes de fonctionnement sachant qu'un bouton servira toujours au lancement du chronomètre.

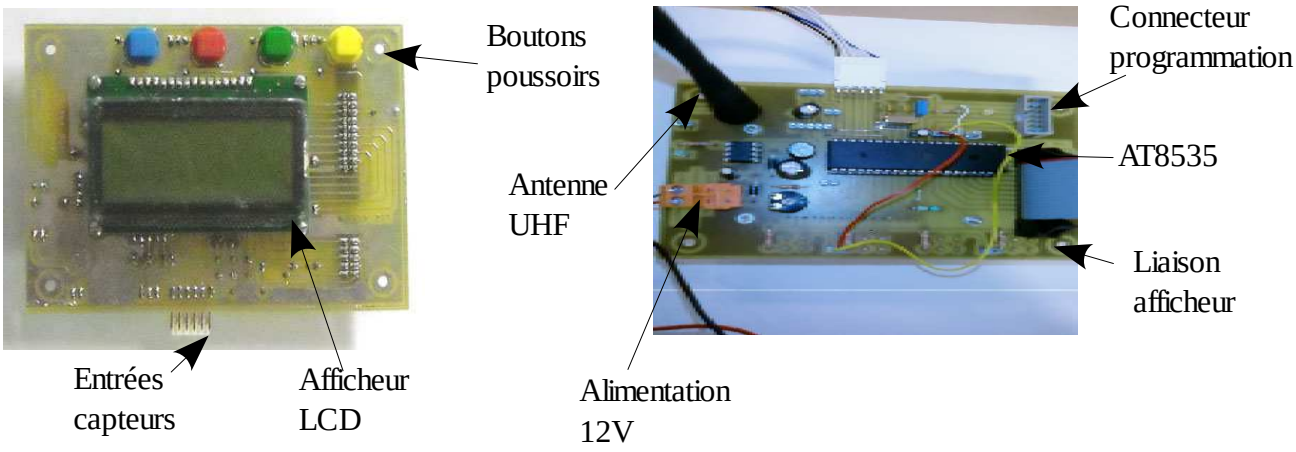


Illustration 7: Carte microcontrôleur

L'afficheur LCD indique le chronomètre en fonction de l'état des entrées. La carte est alimentée par la tension 12V de la batterie.

La carte d'alimentation reçoit en entrée la tension de la batterie 12V et fournie en sortie 30V et 15V. La tension de 30V alimente les feux tricolores alors que le 15V l'afficheur géant ainsi que le boîtier de réglage.

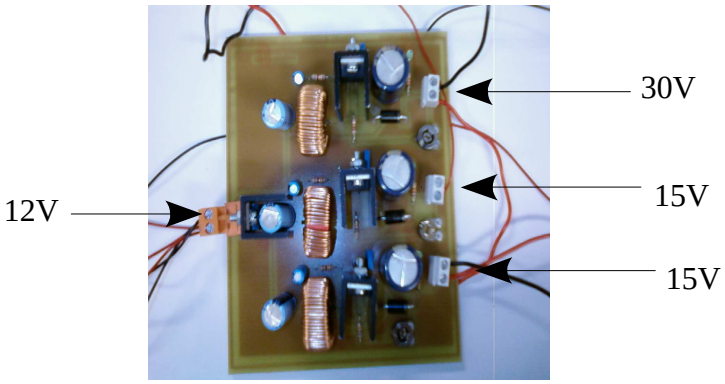


Illustration 8: Carte d'alimentation

La carte de l'illustration 9 commande les couleurs des feux tricolores. Elle est reliée par un câble en nappe à la carte microcontrôleur. Le pilotage d'un feu s'effectue par un niveau logique bas.

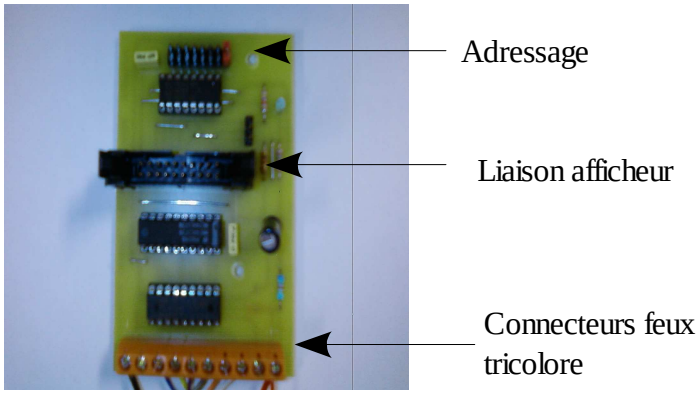


Illustration 9: Carte feux tricolore

L'objectif de notre projet étant la programmation du microcontrôleur ATMéga 8535 qui gére l'ensemble du dispositif, nous allons décrire briévement ce microcontrôleur en présentant plus particulièrement les fonctions mises en oeuvre.

2. Le microcontrôleur AT8535

2.1. Généralités

Le microcontrôleur AT8535 est un circuit programmable capable d'exécuter un programme et qui possède des circuits d'interface intégrés avec le monde extérieur. Nous utiliserons le modèle de type DIL[1] qui possède le brochage suivant :

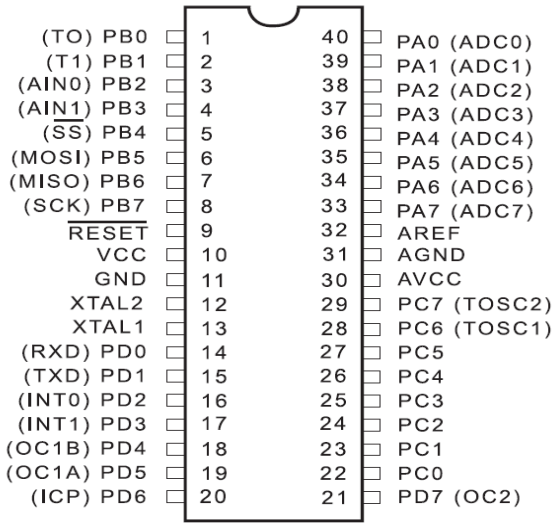


Illustration 10: Brochage de AT90LS8535 [3]

Il est constitué de différentes fonctions :

- CPU 8 Bits capable d'exécuter une instruction par cycle d'horloge,
- 8 Ko de mémoire programme EEPROM FLASH programmable,
- 512 Octets d'EEPROM (Stockage de données non volatiles),
- 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 (dont 1 utilisable en RTC a l'aide d'un oscillateur externe),
- 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.

Les avantages :

- diminution de l'encombrement du matériel et du circuit imprimé,
- simplification du tracé du circuit imprimé (plus besoin de tracer de bus),
- augmentation de la fiabilité du système. Nombre de composants connexions composants/supports et composants circuit imprimé réduit,

1 LID : Dual In Line

- intégration en technologie MOS[1], CMOS[2], ou HCMOS[3] (donc diminution de la consommation),
- le microcontrôleur contribue à réduire les coûts à plusieurs niveaux: moins cher que les composants qu'il remplace et diminution des coûts de main d'oeuvre (conception et montage).

Les inconvénients :

- investissement dans les outils de développement,
- nécessite un technicien informaticien pour l'écriture du programme.

2.2. Les E/S numériques

Il possède 4 ports d'E/S numériques de 8 bits chacun. Chaque bit peut être configuré en entrée ou en sortie et possède des fonctions secondaires mises en oeuvre par programmation des registres internes associés.

Nous les utiliserons pour les liaisons avec les capteurs et les afficheurs (4 digits et feux), de notre application.

2.3. Les timers

Le microcontrôleur AT8535 possède 3 timers (0, 1 et 2) dont nous utiliserons le timer 1 pour notre chronomètre. Les timers 0 et 2 sont des compteurs 8 bits alors que le timer 1 travaille sur 16 bits. Un timer est composé principalement d'un compteur qui évolue en permanence lorsque le timer est actif. Le comptage peut se faire à différentes fréquences et à partir d'une horloge interne ou externe. Il possède un ou plusieurs registres de comparaison et une ou plusieurs broches d'entrée/sortie à configurer.

Les timers 8 bits peuvent compter de 0 à 255 alors que le timer 16 bits compte de 0 à 65 535.

Un registre de comparaison permet de comparer en permanence la valeur du compteur à celle que l'on a chargée lors de la configuration. Lorsqu'il y a égalité entre les 2 valeurs, alors l'interruption du timer s'exécute si celle-ci est autorisée.

Le schéma fonctionnel du timer 1 en annexe 2 nous décrit plus précisément son fonctionnement.

1 MOS : Metal Oxide Semiconductor
2 CMOS : Complementary Metal Oxide Semiconductor
3 HCMOS : Hight speed CMOS

3. Programmation

Le schéma électrique de la carte du microcontrôleur nous étant fourni, il nous a fallu l'analyser afin de découvrir la connectique des entrées/sorties et nous permettre de configurer les ports et le timer.

Le tableau ce-dessous résume l'affectation des bits des ports.

Rang	7	6	5	4	3	2	1	0
Port A	D7	D6	D5	D4	D3	D2	D1	D0
	Données d'affichage et feux							
Port B	BP2	BP1	BP0	CS	A0	A1	A2	A3
				Validation digits afficheurs et feux				
Port C	LCD DB7	LCD DB6	LCD DB5	LCD DB4	NU	LCDE	LCD R/W	LCD RS
Port D	NU	NU	NU	NU	NU	NU	NU	RXD

NU : Non Utilisé RXD : ligne de réception série de la liaison UHF

Tableau 1: Affectation des bits

3.1. Configuration des ports d'E/S

Du tableau d'affectation des bits des ports, on en déduit la configuration des bits soit en entrée, soit en sortie .

	7	6	5	4	3	2	1	0
Port A	S	S	S	S	S	S	S	S
Port B	E	E	E	S	S	S	S	S
Port C	S	S	S	S	S	S	S	S
Port D	E	E	E	E	E	E	E	E

S : Sortie E : Entrée

Tableau 2: Configuration des E/S

Pour configurer un port nous disposons d'un registre 8 bits de direction : DDRx (x pour A, B, C ou D). Chaque bit peut être configuré en entrée (0) ou en sorie (1).

Prenons l'exemple du Port B :

PIN	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Configuration	0	0	0	1	1	1	1	1

Tableau 3: Résumé de la configuration du Port B

Ce qui donne DDRB = 0x1F;

La configuration des ports fait partie de la fonction BrdInit( ) que l'on appelle au début du programme et est de la forme suivant :

```
// 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=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
PORTC=0x00;
DDRC=0xFF;

// Port D 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
PORTD=0x00;
DDRD=0x00;
```

3.2. Configuration du Timer

Comme nous l'avons vu précédemment, le microcontrôleur est composé de 3 Timers. Pour réaliser notre chronomètre à la centième de seconde sur 1 minute, il nous faut prendre, en considération la fréquence de l'oscillateur de la carte ATmega (16 MHz), les valeurs maximales de comptage des timers ainsi que les facteurs possibles de prédivision de l'horloge (/8, /64,...).

Il nous faut compter au maximum 5999 centième de seconde. La fréquence de l'horloge étant de 16 Mhz, nous avons une période : $T = 1 / f = 1 / 16.10^6 = 62,5$ ηs.

Nous devons compter $10.10^{-3} / 62,5.10^{-9} = 160\ 000$ périodes d'horloge. Sachant que le timer 1 ne peut compter que de 0 à 65 535, il nous faut effectuer une prédivision de la fréquence par 8. Cela nous donne une fréquence de 2 MHz. La période est donc de : $T = 1 / 2.10^6 = 0,5$ µs. Il nous faut compter au final $10.10^{-3} / 0,5.10^{-6}$ soit 20 000 périodes. Toutes les 20 000 périodes l'interruption liée au timer 1 sera exécutée et incrémentera la variable de chronométrage qui gère les digits d'affichage.

Pour mettre en oeuvre ce timer nous devons programmer différents registres mentionnés dans le schéma fonctionnel (annexe 2).

Registre de contrôle (TCCR1A[1]) :

Le registre TCCR1A est une importante partie du timer. Il va gérer tous les autres registres du timer et aiguiller les informations suivant la configuration donnée.

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Illustration 11: Description du registre de contrôle A

1 TCCR1A : Timer/Counter1 Control Register A

Les bits COMX1[2] et COMX0 permettent de définir l'action qui doit se produire sur la sortie OC1X lorsque les valeurs présentes dans le registre de comparaison OCR1X[3] et le compteur TCNT1[4] sont égales. Il faut donc regarder le tableau suivant pour savoir quelles valeurs donner à ces bits.

COM1X1	COM1X0	Description
0	0	Timer/Counter1 disconnected from output pin OC1X
0	1	Toggle the OC1X output line.
1	0	Clear the OC1X output line (to zero).
1	1	Set the OC1X output line (to one).

Note: X = A or B.

Tableau 4: Compare 1 du mode Selec

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

Tableau 5: PWM mode Selec

On s'aperçoit que OC1A[1] est déjà relié à un capteur. Nous ne pourrons donc pas nous en servir en sortie ce qui donne : `TCCR1A = 0x40 ;`

Le registre TCCR1B de contrôle nous servira pour autoriser ou non le timer mais aussi pour diviser la fréquence du quartz par 8 comme nous l'avons vu précédemment.

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	–	–	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Illustration 12: Description du registre de contrôle B

Le bit 7 permet de s'affranchir des parasites. Nous l'avons mis à 0 car il ne nous est d'aucune utilité.

Le bit 6 permet d'enregistrer l'entrée sélectionnée or cette fonction n'est pas utile dans le cadre de notre projet. Il sera donc à 0.

Les bits 4 et 5 sont réservé d'après la documentation du constructeur.

2 COMx1 : Compare Out Mode
3 OCR1x : Output Compare Register
4 TCNT1 : Timer Counter
1 OC1A : Output Compare match A output

Le bit 3 CTC1[2] indique que le compteur est remis à 0 dès qu'il atteint la valeur du comparateur A (OCR1A) c'est-à-dire que le compteur recommence son cycle. Il nous est très utile et doit être mis à 1.

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

Illustration 13: Description des bits 0, 1 et 2

Nous avons vu auparavant qu'il nous faut effectuer une division par 8 de l'horloge. Il nous faut donc mettre les bits de rang 1 et 3 à 1. D'où `TCCR1B = 0x0A ;`

Configuration des compteurs du Timer (TCNT1) :

TCNT1L représente le poids faible du compteur du Timer 1 alors que TCNT1H représente son poids fort. Ils seront initialement mis à 0 ce qui donne : `TCNT1H = 0x00 ; TCNT1L = 0x00 ;`

Configuration du registre de capture du Timer (ICR1[1]) :

Lors d'un front actif sur ICP[2], le contenu du compteur libre TCNT1 est recopié dans le registre ICR1 de plus, le bit ICF1[3] du registre TIFR est positionné à 1. ICR1L représente le poids faible du compteur du Timer 1 alors que ICR1H représente son poids fort. Pour notre projet nous n'aurons pas besoin de cette fonction de capture ce qui donne : `ICR1H = 0x00 ; ICR1L = 0x00 ;`

Configuration des registre de comparaison du Timer 1 (OCR1) :

Les registres de comparaison contiennent la valeur qui est comparée en permanence avec le contenu du compteur libre TCNT1. Lorsque la valeur contenue dans le registre OCR1x est égale à celle présente dans TCNT1 l'action programmée sur OC1x est exécutée et le bit OCF1x du registre TIFR[4] est positionné à 1. Or nous avons vu précédemment que la valeur de comparaison devra être 20 000 ce qui correspond en hexadécimal à 4E20. Donc `OCR1AH = 0x4E ; OCR1AL = 0x20 ;`

Le registre B n'étant pas utilisé, ses comparateurs seront mis à 0 : `OCR1BH = 0x00 ; OCR1BL = 0x00 ;`

2 CTC1 : Clear Timer/Counter1 on Compare Match
1 ICR1 : Input Capture Rgister
2 ICP : Input Capture Pin
3 ICF : Input Capture Flag
4 TIFR : Timer Interrupt Flag Register

Configuration du registre de validation (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	
$39 ($59)	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Illustration 14: Description du registre de validation

Lorsque l'on active le bit 4, on autorise, lorsqu'il y a égalité entre le compteur et le registre de comparaison, l'exécution de la routine d'interruption correspondante. C'est dans cette interruption que sera gérée la valeur des différents digits du chronomètre.

Cette configuration sera elle-aussi dans la fonction BrdInit() qui s'effectue au début du programme c'est-à-dire lors de la mise en route de la borne.

3.3. Programme principal

Le programme principal a été réalisé en simplifiant le cahier des charges à cause du retard pris à cause des mouvements de grève. Nous avons donc décidé de partir du principe qu'il n'y aura qu'un seul kart et que le départ pourra être manuel ou automatique.

L'organigramme qui suit présente de façon claire le déroulement du programme principal dans lequel on a mentionné :

Rouge = feux en rouge fixe

Orange = feux en orange fixe

Vert = feux en vert fixe

VertClig = feux en vert clignotant

RougeClig = feux en rouge clignotant

Capt1 = capteur n°1

Capt2 = capteur n°2

BpStart = bouton start

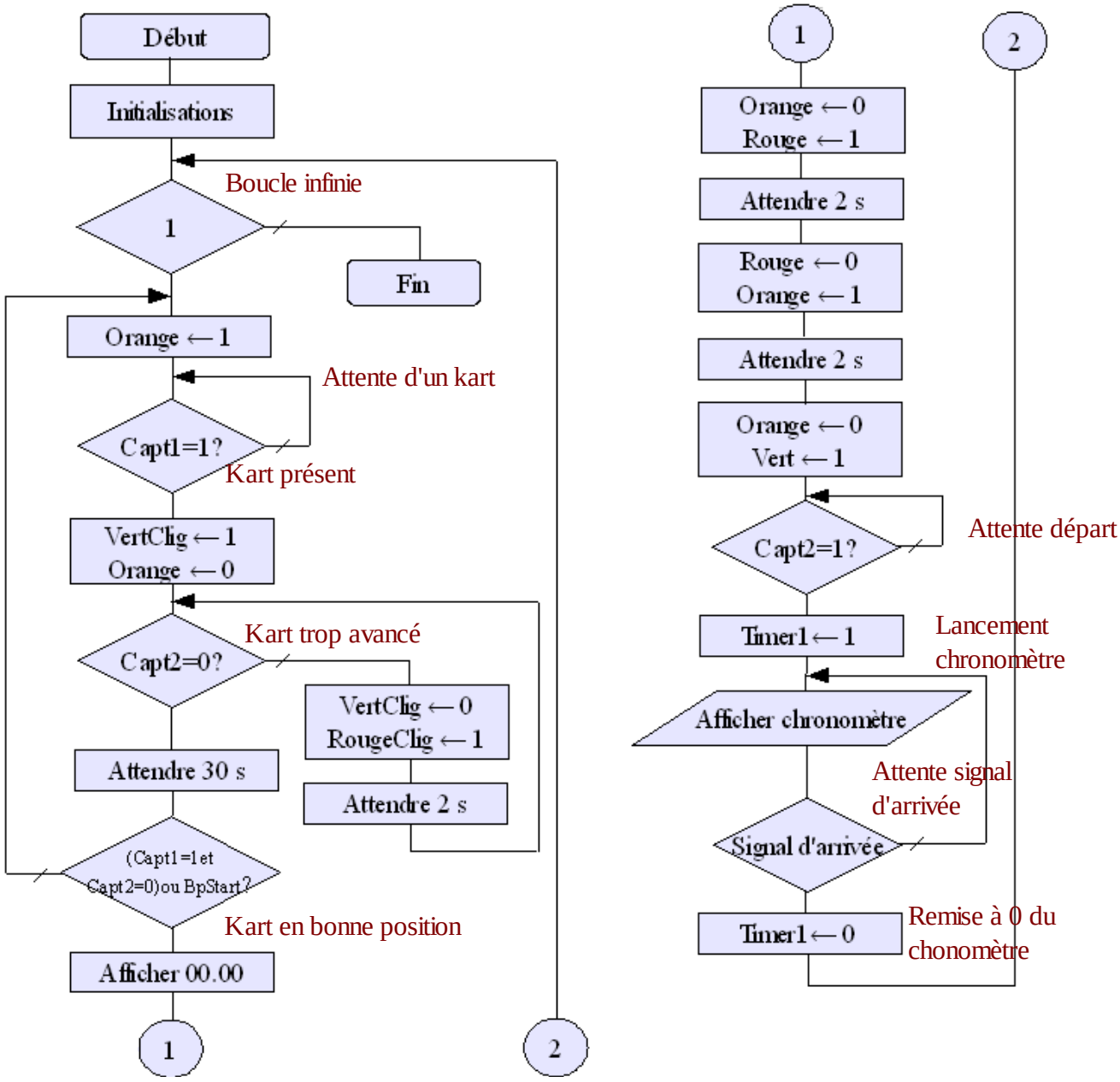


Illustration 15: Organigramme du programme principal

La boucle infinie permet de faire en sorte que le programme s'exécute sans cesse. Ensuite, on fait allume le feu orange et on attend qu'un kart se présente. Lorsque le kart coupe le premier faisceau, on fait clignoter le feu vert. Puis on test s'il est en bonne position. Si le kart est trop avancé on fait clignoter le feu rouge. Une fois le kart dans la bonne position, on attend 30 seconde puis on retest sa position. Le kart coupant les bons faisceaux, il reste plus qu'à actionner les différents feux (du rouge fixe au vert fixe). Une fois le feu au vert, on attend que le kart parte c'est-à-dire qu'il passe devant le deuxième capteur. Le chronomètre se déclenche et on l'affiche. Lorsque l'on reçoit le signal d'arrivée, on arrête le chronomètre et on remet les compteurs du timer à 0 tout en laissant le temps final inscrit sur l'afficheur géant et l'écran LCD.

Il ne reste plus qu'à traduire cet organigramme en langage C++ pour le logiciel CodeVisionAVR.

```
void main (void)
{
	unsigned char phrase[16];		//déclaration d'un tableau pour l'afficheur LCD
	brdInit();				//initialisations
	while(1)	//boucle infinie
	{
		do
		{
			afficheur1(7,Orange,0);		//commande feu orange
			if(capt1==1);			//un kart se présente
			afficheur1(7,VertClig,0);
			if(capt2==0)			//le kart ne coupe pas le deuxième faisceau
			{
				delay(30000);		//attendre 30 000ms soit 30s
			}
			else				//kart en position de faux départ
			{
				afficheur1(7,RougeClig,0);
				delay(2000);
			}
		}
		while ((cap1==1)&&(capt2==0)||(BpStart==0));	/*kart en bonne position ou appui sur le bouton start*/
		unite = 0;
		dizeme = 0;
		dixieme = 0;
		centieme = 0;
		afficheur1(0,dizaine,0);		//commande afficheur géant : dizaine
		afficheur1(1,unite,1);			//commande afficheur géant : unité
		afficheur1(2,dixieme,0);		//commande afficheur géant : dixième
		afficheur1(3,centieme,0);		//commande afficheur géant : centième
		sprintf(phrase,"%d %d %d %d  ",dizaine,unite,dizieme,centieme);
```

```
            lcd_gotoxy(1,1);            //emplacement de l'affichage de la phrase
            lcd_puts(phrase);           //affichage
            afficheur1(7,Rouge,0);
            delay(2000);
            afficheur1(7,Orange,0);
            delay(2000);
            afficheur1(7,Vert,0);
            while(capt2==1);
            TCCR1B = 0x0A;              //déclenchement du chronomètre
            do
            {
                  afficheur1(0,dizaine,0);
                  afficheur1(1,unite,1);
                  afficheur1(2,dixieme,0);
                  afficheur1(3,centieme,0);
            }
            while(reception=='s');      //attente du signal d'arrivée
            TCCR1B = oxoo;              //arrêt du chronomètre
      }
}
```

Vous retrouverez en annexe 3 le programme complet ce-dessus avec les déclarations et la partie des initialisations.

Conclusion

Ce projet nous a apporté de nouvelles connaissances en informatique et en électronique. En effet, nous avons dû étudier la borne de départ à l'aide des documents fournis par un binôme de notre promotion ainsi que ceux donnés par Thierry Lequeu. Une fois cette partie effectuée, nous nous sommes penchés sur la partie informatique afin d'écrire le programme.

La programmation était en partie nouvelle pour nous car seulement l'un d'entre nous connaissait le logiciel utilisé. Cela nous a appris à vraiment travail en équipe en s'aidant mutuellement et en s'échangeant nos connaissances.

Progressivement et en testant régulièrement me programme, nous avons rempli une grande partie de notre cahier des charges. Ce projet aurait pu être mené à bien sans les mouvements de grèves que nous avons eu. Il reste encore du travail surtout pour la communication entre les deux bornes et la prise en compte des différents modes de fonctionnement.

Résumé

Nous avons choisi ce projet car il nous semblait très complet et nous permettait d'approfondir nos connaissances dans deux principaux domaines : l'électronique et l'informatique.

Après une étude des différents éléments qui composent la borne de départ mise à notre disposition, nous nous sommes penchés sur la carte du microcontrôleur. Cette carte est le cœur de notre projet car elle est composée de l'ATmega 8535 qui va nous permettre de gérer les feux tricolores, l'afficheur 4 digits ainsi que les capteurs. Nous avons donc passé beaucoup de temps dessus.

Une séance a été consacrée à son étude et à celle du document constructeur. Nous avons ensuite configuré les ports ainsi que le timer choisi en fonction de l'application à savoir le chronométrage. Pour cela, nous avons répertorié par port les bits en entrée et ceux en sortie.

Nous avons réalisé de nombreux petits programmes afin de tester le fonctionnement du protocole pour les afficheurs mais aussi pour vérifier que l'information des capteurs arrivait sur notre carte. Le programme principal a été réalisé à la fin en regroupant ces programmes et en essayant de faire en sorte qu'il soit compréhensible pour tous et que la borne soit facile d'utilisation.

197 mots

Index des illustrations

Illustration 1: Présentation de l'épreuve.....6

Illustration 2: Synoptique de premier niveau.....6

Illustration 3: Schéma de principe de la mesure du temps pour l’épreuve de 50 mètres départ arrêté 7

Illustration 4: Planning prévisionnel et réel.....8

Illustration 5: Synoptique de niveau 3 de la borne [3].....9

Illustration 6: Description de la borne.....10

Illustration 7: Carte microcontrôleur.....10

Illustration 8: Carte d'alimentation.....11

Illustration 9: Carte feux tricolore.....11

Illustration 10: Brochage de AT90LS8535 [3].....12

Illustration 11: Description du registre de contrôle A.....16

Illustration 12: Description du registre de contrôle B.....17

Illustration 13: Description des bits 0, 1 et 2.....18

Illustration 14: Description du registre de validation.....19

Illustration 15: Organigramme du programme principal.....20

Bibliographie

[1] **VICENTE Jérôme**, *Les microcontrôleur*, 2006 (page consultée en 2009)
< http://iusti.polytech.univ-mrs.fr/~vicente/supportcours/cours_microcontrol_2005.pdf >

[2] **SEGURET J-M,** *Le module timer du uc AT90S8535*, 2002 (page consultée en 2009)
< http://sti.tice.ac-orleans-tours.fr/spip//IMG/pdf/Timer_AT8535_V2.pdf >

[3] **SEGURET J-M**, *Présentation du microcontrôleur AT90S8535*, 2002 (page consultée en 2009) < http://sti.tice.ac-orleans-tours.fr/spip//IMG/pdf/Presentation_AT8535-2.pdf >

[4] **LEQUEU Thierry,** *La documentation de Thierry sur OVH*, (page consultée en 2009)
< http://thierry-lequeu.fr/ >

Annexes

Annexe 1 : Schéma électrique de la carte microcontrôleur.....28

Annexe 2 : Schéma fonctionnel du timer 1.....29

Annexe 3 : Programme complet.....30