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

Projet Tutoré – Étude et Réalisation

Mesure de temps de parcours par RFID

BINTI ABDULLAH Akmal – SAPIENS Corentin
Année 2011 – Groupe Q2
Promotion 2009 - 2011

Enseignants : M. LEQUEU
M. PAPAZIAN

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

Projet Tutoré – Étude et Réalisation

Mesure de temps de parcours par RFID

BINTI ABDULLAH Akmal – SAPIENS Corentin
Année 2011 – Groupe Q2
Promotion 2009 - 2011

Enseignants : M. LEQUEU
M. PAPAZIAN

Sommaire

Introduction....4

1.Présentation du sujet....5
1.1.Cahier des charges....5
1.2.Qu'est ce que la RFID ?....6
1.3.Planning....7

2.Études du système....9
2.1.Solutions envisagées....9
2.2.Choix des composants....12
2.3.Les principaux composants....14
2.4.Initialisation de CodeVision AVR....16

3.Réalisation....19
3.1.Routage sur Orcad....19
3.2.Programmation de la partie écriture....25
3.3.Programmation de la partie lecture....31
3.4.Problèmes rencontrés....33

Conclusion....35

Résumé....36

Annexes....39

Introduction

Dans le cadre d'un 3ème semestre d'IUT, nous devons réaliser un projet en Études de Réalisations. Nous avons choisi le projet de mesure du temps de parcours par RFID (Radio Frequency Identification), 2 groupes d'étudiants des années précédentes ont travaillé sur ce projet avant nous : Vivien MARTINEZ dans un premier temps, puis HOGUET Romain et MARCHAND Rémi par la suite.

La partie de notre projet est divisée en deux : l'émission et la réception. Nous allons utiliser la technologie des puces RFID, afin de calculer des temps de parcours lors d'une course à pied. La plus grande partie de notre travail portera sur la programmation, en utilisant le logiciel Code Vision AVR. Comme les typons et les prototypes ont déjà été réalises, nous n'avons pas eu à nous occuper de la partie électronique.

Ce rapport commence par la présentation du sujet, suivi par l'étude technologiques du système et enfin par la méthode de réalisation des cartes de lecture et d'écriture.

1. Présentation du sujet

Préalablement conçu pour des raids ou des cross, le chronométrage sans fil est généralement loué pour des manifestations sportives regroupant de nombreux concurrents. Lors d'une course à pied par exemple, les participants se verront remettre avant le départ, un badge personnel, En passant à travers des portiques disséminés sur le parcours, le temps de passage sera automatiquement inscrit dans les badges. Il suffira de lire les données des badges à la fin de la course pour déterminer le temps de parcours de chacun.

Illustration 1: exemple d'émetteur RFID[1]

1.1. Cahier des charges

Ces dispositifs coûtent très chêr, c'est pourquoi l'association Tours'n Aventure avait demandé à Vivien MARTINEZ de refaire de toute pièce, les boîtiers de mesure. Notre rôle est de terminer ce projet. Le principe sera quelque peu simplifié. Les portiques seront remplacés par des bornes.

Chaque concurrent devra présenter son badge devant chacune d'entre elles. Avant la manifestation, une personne de l'organisation doit affecter une fonction à chaque boitier (borne de départ, intermédiaire, d'arrivée) en attribuant un numéro et en synchronisant l'horloge interne.

Pour permettre une intervention rapide et sans difficulté de l'opérateur, les boitiers devront s'ouvrir facilement et l'accès aux boutons de réglages doivent être aisé. L'affichage du numéro de borne et de l'horloge seront placés en façade. Chaque boitier devra être facilement transportable et stable. De plus, pour éviter les court-circuit, les bornes devront être étanches et solides.

L'autonomie électrique de chaque borne se fera à une batterie qui doit assurer le fonctionnement pendant une centaine d'heures. Le boitier a été choisi et possède les dimensions suivante : 160x98x68mm. Chaque transmission se fera par la technologie RFID.

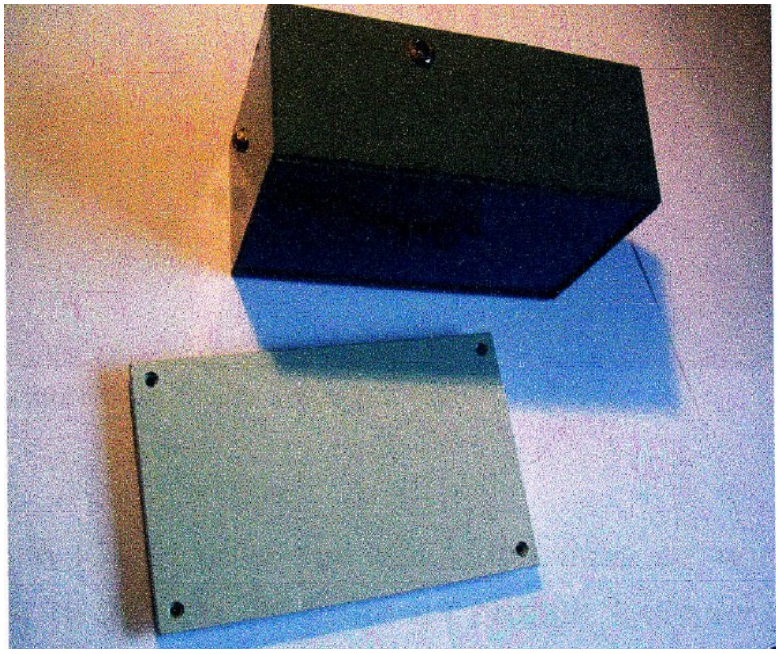

Illustration 2: Le boitier[2]

L'association à qui bénéficie le projet, a choisi que le budget total doit être inférieur à 400€ pour l'ensemble des bornes et de 200€ pour les badges individuels.

1.2. Qu'est ce que la RFID ?

La RFID est une technologie analogue au code barre. Contrairement à celui-ci, l'étiquette RFID, aussi appelée TAG, n'a pas besoin de lecteur optique pour être lu. Elle contient un système électronique implanté dans du plastique sous forme de carte de crédit, de pièce ou de badge. Elle n'a pas besoin d'une source d'alimentation externe. La lecture se fait donc à distance pouvant aller d'une dizaine de centimètres à quelques mètres. Un module distant, composé d'un circuit imprimé et d'une antenne, interroge le TAG qui envoie ensuite ses données. Un TAG peut être incorporé à un produit ou bien porté sur soi. Grâce à son numéro d'identification unique, il est nominatif. Cette technologie est principalement utilisée pour stocker et récupérer des données à distance. De plus, il existe une procédure de sécurité pour certains modèles.

La technologie RFID trouve son utilisation dans de nombreux domaines, dans lesquels il est nécessaire de suivre un produit à distance ou d'authentifier une personne : industries, portes d'immeubles, protections antivol dans les grands magasins...

Illustration 3: Badge RFID[3]

1.3. Planning

Pour répondre au cahier des charges, nous avons bénéficié de onze séances de projet et de deux séances libres. Un cours a été consacré à la formation du logiciel Orcad permettant de réaliser des schémas électriques et des typons, mais dans notre cas, inutile.

Le projet possédait déjà deux prototypes, ce qui nous a permis de ne pas avoir à en réaliser et de passer directement à la partie programmation. C'est pourquoi le planning final n'est pas en adéquation avec le prévisionnel. Cependant, nous nous sommes heurtés à plusieurs problèmes techniques à cause du « va-et-vient » du matériel entre les projets.

La réalisation des prototypes ayant déjà été réalisés, nous avons pu nous consacrer pleinement à la partie programmation. Le rapport des semaines figure en annexe 5 :

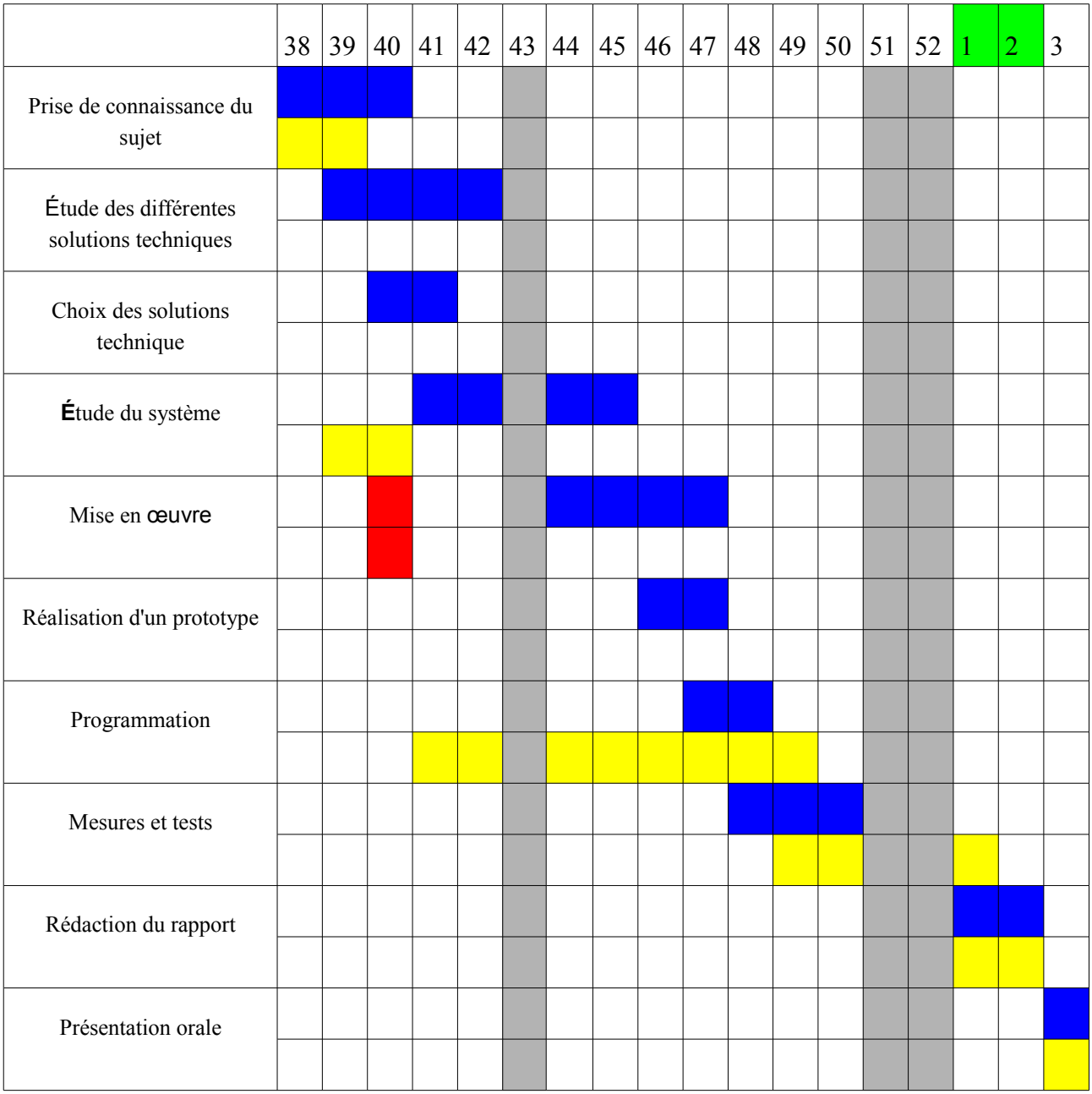

Tableau 1 : Planning prévisionnel et final[4]

	Planning prévisionnel
	Planning réel
	Séance libre
	Apprentissage Orcad

2. Études du système

2.1. Solutions envisagées

Vivien MARTINEZ avait envisagé de faire une carte, qui pouvait recevoir deux types de modules RFID. Le premier (à gauche sur l'image) ayant une fréquence de 125 KHz et le second de 13,56 MHz (à droite). En effet, plus on augmente la fréquence, plus le TAG peut être lu à une distance éloignée. Or, la distance du module longue portée n'est que d'une dizaine de centimètres. Il n'était donc pas question de prendre le module de 125 KHz. C'est pourquoi la carte a été faite pour recevoir seulement le module haute fréquence (13,56 MHz).

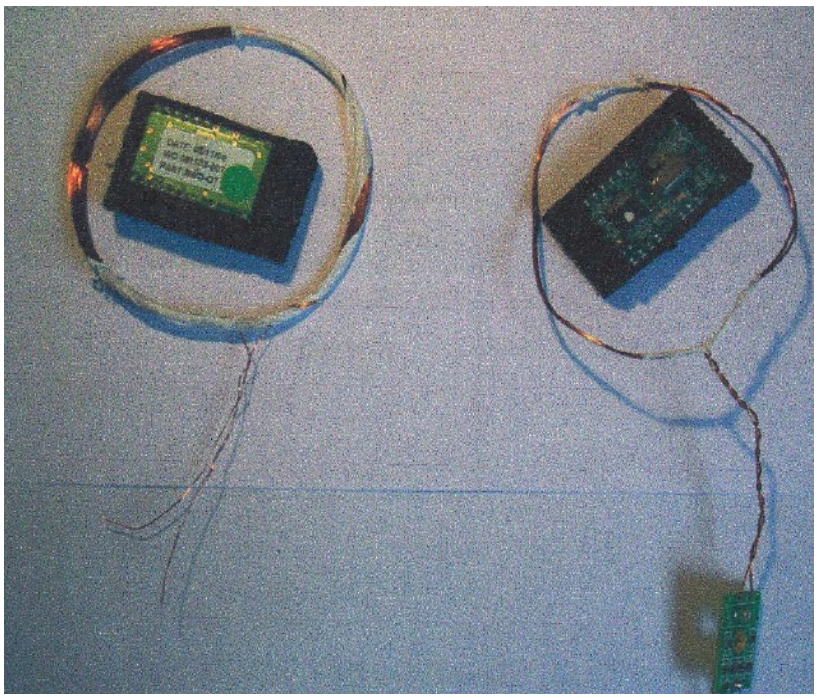

Illustration 4: Modules RFID et leurs antennes[2]

En revanche, le choix du micro-contrôleur ATMEGA 8535 et du régulateur de tension LM2574 a été conservé.

Voici les schémas fonctionnels du projet :

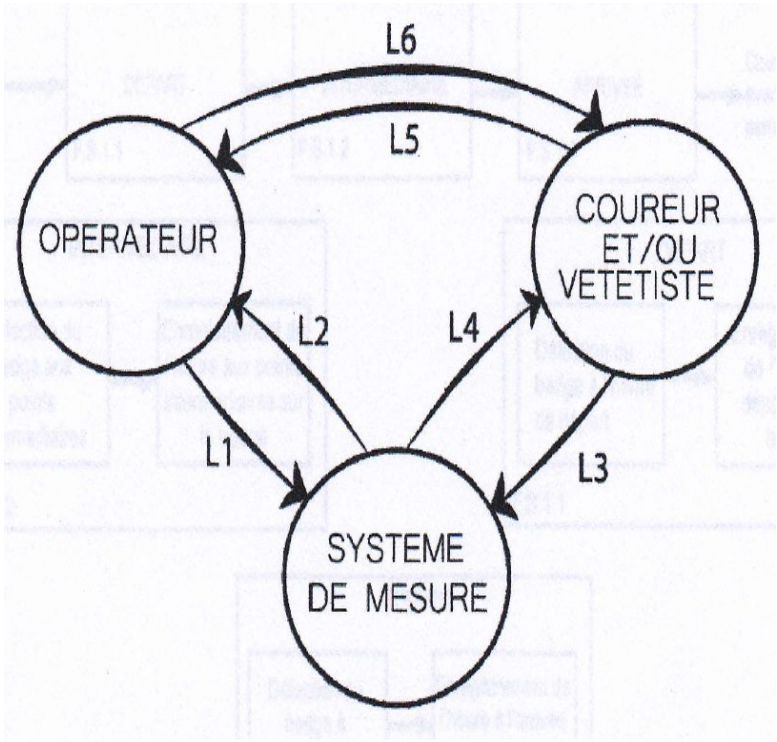

Illustration 5: Schéma global du projet[5]

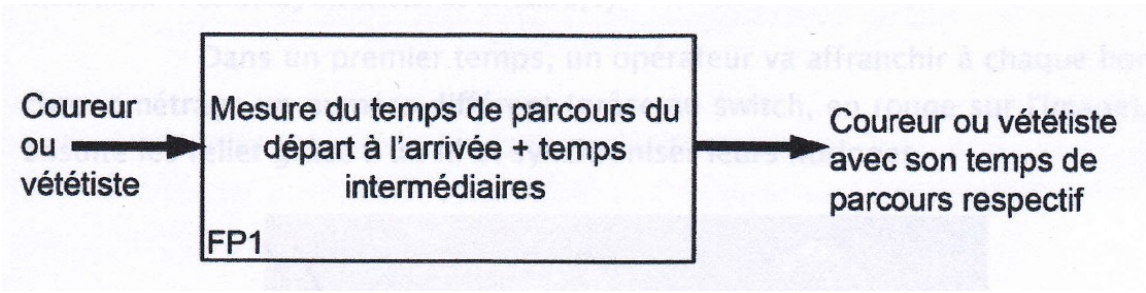

Illustration 6: Schéma fonctionnel de niveau 1 [5]

On mesure donc le temps de parcours du coureur. Chaque badge va être analysé par les bornes qui vont ensuite écrire le temps de passage.

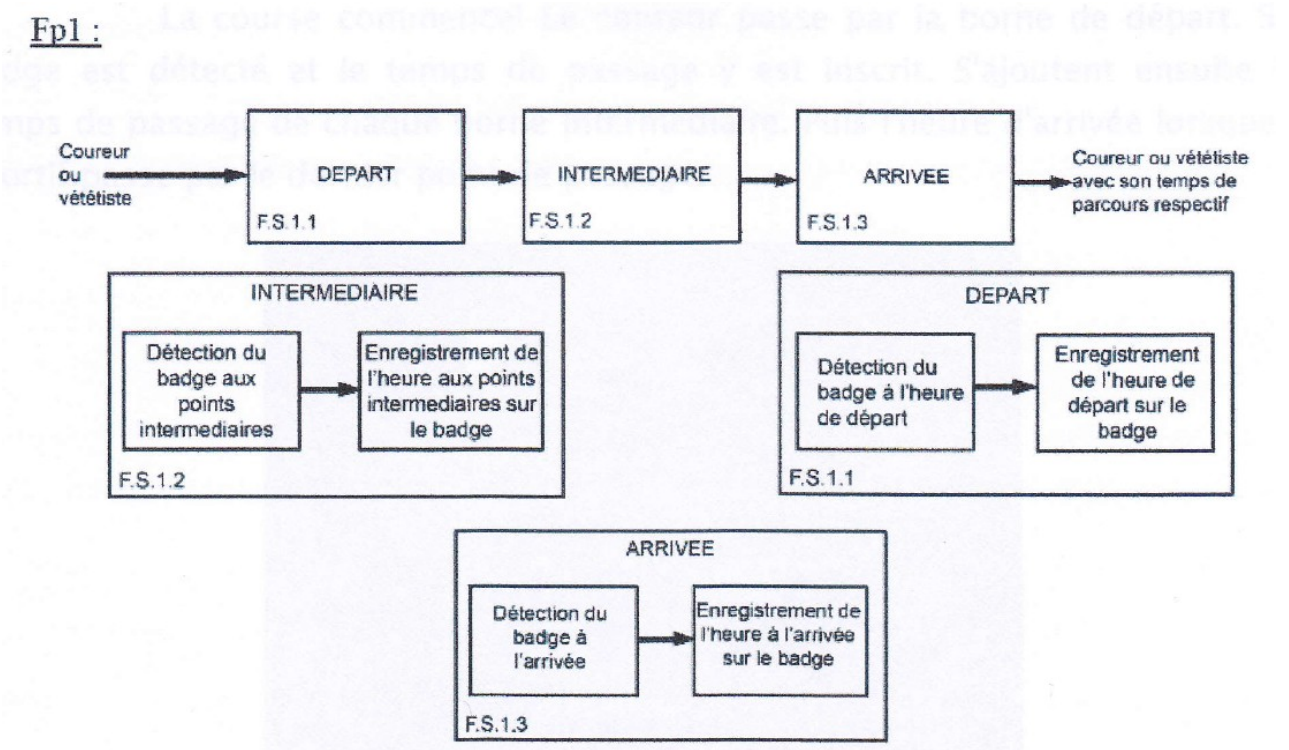

Illustration 7: Schéma fonctionnel de niveau 2 [5]

Dans un premier temps, un opérateur va affranchir à chaque borne de chronométrage un numéro différent (grâce au switch, en rouge sur l'image), pour ensuite les relier grâce à un fil et synchroniser leurs horloges.

Illustration 8: La carte finie[3]

Ensuite, les responsables de la manifestation vont affecter à chaque participant leur badge (ou TAG). Grâce à son identifiant unique, il va être nominatif et permettra de faire en sorte que chaque personne ait son numéro comme avec un dossard.

La course commence! Le coureur passe par la borne de départ. Son badge est détecté et le temps de passage y est inscrit. S'ajoutent ensuite les temps de passage de chaque borne intermédiaire. Puis l'heure d'arrivé lorsque le sportif passe par le dernier point de passage.

Enfin chaque participant remet aux responsables son badge, qui va être décodé par une borne spécifique à la lecture. Les informations sont alors stockées dans une base de données et le classement est établi.

2.2. Choix des composants

Pour respecter le budget, les composants les moins chers ont été choisi. Voici leur nomenclature et le prix de chacun:

Fournisseur	Nature du composant	Référence	Prix unitaire (HT),€	Quantité	Prix Total (HT),€	Notes
Radiospares	Micro-contrôleur	ATMEGA 8535	6,17	1	6,17	
Radiospares	Régulateur de tension	LM2574N-5	2,83	1	2,83	
Radiospares	Module RFID	RWD MIFARE	38,75	1	38,75	
Radiospares	Antenne RFID	ANT 1356M	14,23	1	14,23	
Radiospares	Afficheur LCD	16x4 caractères	7,5	1	7,5	
Radiospares	Condensateur	100µF 63V	0,28	1	0,28	Vendu par 5
Radiospares	Condensateur	470µF 6,3V	0,272	1	0,272	Vendu par 5
Radiospares	Condensateur	10µF 6,3V	0,908	2	1,816	Vendu par 5
Radiospares	Condensateur	100nF	0,102	2	0,204	Vendu par 5
Radiospares	Condensateur	22pF	0,0244	2	0,0488	Vendu par 25
Radiospares	Diode	1N4007	0,18	1	0,18	Vendu par 5
Radiospares	Diode	1N5819	0,24	1	0,24	Vendu par 5
Radiospares	DEL	2mA 3mm Rouge	0,156	1	0,156	Vendu par 50
Radiospares	DEL	2mA 3mm	0,075	1	0,075	Vendu par 50

		Orange				
Radiospares	DEL	2mA 3mm Verte	0,166	2	0,332	Vendu par 50
	Batterie		15	1	15	Non commandé
Radiospares	Buzzer	KPEG220A	2,81	1	2,81	
Radiospares	Fusible		0,66	1	0,66	Vendu par 5
Radiospares	Porte-Fusible		0,93	1	0,93	Vendu par 5
Radiospares	Switch	Glissière 4 voies	1,37	1	1,37	
Radiospares	Quartz	16Mhz	2,4	1	2,4	Vendu par 10
Radiospares	Inductance	10µH	0,976	1	0,976	Vendu par 5
Radiospares	Inductance	47µH	0,975	1	0,975	Vendu par 5
Radiospares	Bornier	2 fils	0,65	3	1,95	Vendu par 5
Radiospares	Bouton Poussoir	4 broches	0,75	1	0,75	Vendu par 5
Radiospares	Potentiomètre	10kΩ	0,0432	1	0,0432	Vendu par 5
Radiospares	Résistance	1,5kΩ	0,0616	1	0,0616	Vendu par 5
Radiospares	Résistance	4,7kΩ	0,0636	2	0,1272	Vendu par 5
Radiospares	Résistance	10kΩ	0,25	2	0,50	Vendu par 5
Radiospares	Réseau résistance	4x4,7kΩ	1,054	1	1,054	Vendu par 5
Radiospares	Interrupteur	2 positions	1,21	1	1,21	
Radiospares	Support ATMEGA	Tulipe	4,12	1	4,12	
Radiospares	Support LM et Switch	Tulipe	0,91	2	1,82	
Radiospares	Plaque cuivrée	160x100mm	3,24	1	3,24	
Radiospares	Boitier	160x98x68mm				
			Total (HT)	44	113,08 €	

Le prix d'une borne est donc d'environ 135€ TTC. Il faut savoir que plus les composants sont commandés en quantité, plus le coût total diminue. C'est pourquoi, construire les bornes à l'IUT a été moins onéreux.

2.3. Les principaux composants

2.3.1. Le micro-contrôleur ATMEGA 8535

L'ATMEGA 8535 est un micro-contrôleur très utilisé à l'IUT. C'est pourquoi il a été choisi, car il est facilement disponible au magasin. Il n'y avait donc pas eu de besoin de créer de commande.

Ce composant permet après programmation de gérer des entrées/sorties en autonomie. C'est le cerveau de la borne. Il fait non seulement le lien entre le module RFID et sa mémoire, mais il écrit aussi sur l'afficheur. Pour fonctionner, il a besoin d'un montage annexe composé d'un quartz et de condensateurs. Celui-ci va permettre de donner une vitesse de calcul comme dans le cas d'un ordinateur.

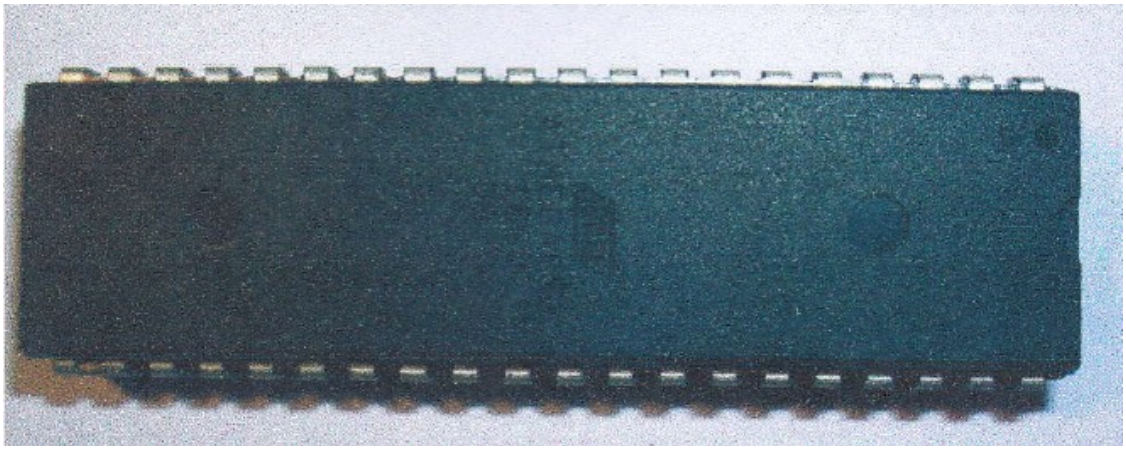

Illustration 9: ATMEGA 8535[2]

Afin de programmer l'ATMEGA 8535, on utilise le logiciel CodeVision AVR. La programmation se fait en C, en utilisant les fonctions basiques de ce langage.

2.3.2. Le régulateur de tension LM2574N-5

Le LM2574N-5 est un régulateur de tension, capable de supporter une source de tension comprise entre 7 et 40 volts continu. Il va faire en sorte d'obtenir 5 volts en sortie en toutes circonstances. C'est pourquoi lors des essais, le montage a été alimenté avec une alimentation continue de 12 volts et la version autonome sera alimentée avec une batterie de 9 volts. Le bon fonctionnement du régulateur est indiqué par un led verte positionnée à coté de l'afficheur.

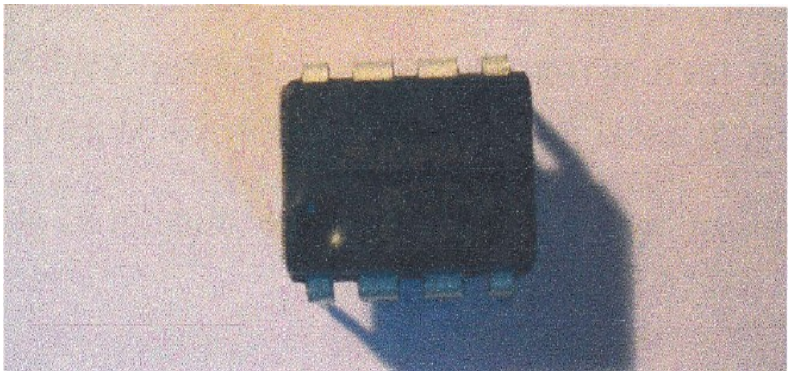

Illustration 10: Régulateur de tension LM2574N-5 [2]

2.3.3. Le module RFID RWD MIFARE

Comme vu précédemment, le module RFID fait la liaison entre le TAG et le micro-contrôleur par le biais de son antenne. Sa fréquence est de 13,56 Mhz. Il dispose d'un programme interne qui gère des trames de données. Ainsi l'ATMEGA peut par exemple envoyer une demande de lecture du code d'identification unique par liaison série RS232.

L'information arrive au module qui avec un autre langage, va demander des informations sur l'IUD au TAG pour les réceptionner et les renvoyer au micro-contrôleur.

Il faut faire attention de bien gérer la sortie CTS « Ready To Receive ». Celle-ci envoie une impulsion qui montre que le module est prêt à réceptionner les informations; Dans le cas contraire, les données sont perdues et le module ne fonctionne pas.

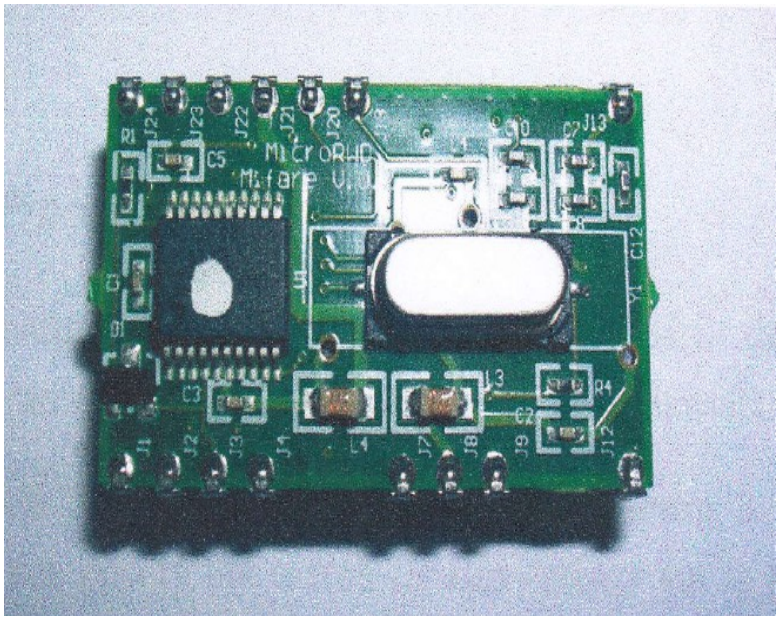

Illustration 11: Module CMS RFID 13,56 MHz [2]

La détection de la carte est gérée par deux voyants: une led verte et une led rouge. Si le TAG est bien reconnu et qu'aucune erreur n'est détectée lors de la transmission de la trame, la led verte s'allume. Sinon, c'est la led rouge qui s'allume de façon continue.

Le datasheet de ce module et son implantation est disponible en annexe 1 et 2.

2.3.4. L'afficheur 16x4 caractères

L'afficheur LCD choisi, dispose de 16 caractères sur 4 lignes. L'écran est choisi rétro-éclairé ce qui permet un meilleur confort lorsqu'il fait sombre ou même nuit. Il affiche grâce aux données transmises par l'ATMEGA, des informations pratiques pour l'utilisateur.

2.3.4.1. En écriture

On va alors faire figurer dans un premier temps le numéro de la borne, puis l'IUD de la carte lorsqu'elle est lue et reconnue, le statut de la carte et enfin l'horloge interne du dispositif. La largeur de l'écran permet d'afficher sur une même ligne, le temps avec ses heures, ses minutes et ses secondes.

2.3.4.2. En lecture

En ce qui concerne le programme de la lecture des temps, nous affichons d'abord la statut de la carte ainsi que le numéro de la borne que l'on choisit de lire, puis nous affichons ensuite l'IUD de la carte et enfin nous affichons les données correspondant aux temps réalisé par les coureurs.

2.4. Initialisation de CodeVision AVR

CodeVision AVR est un logiciel de programmation en C qui permet d'utiliser toute les fonctions d'un micro-contrôleur, c'est pourquoi nous l'avons choisi. Lors de la création du projet sur CodeVision, il nous suffit d'insérer les différentes données (écran LCD, interruptions...)

Une fois CodeVision AVR ouvert, il faut cliquer sur « Fichier » → « New », puis sélectionner « Poject ». Une page s'affichera alors pour demander si on veut utiliser CodeWizartAVR, il faut répondre « Yes ».

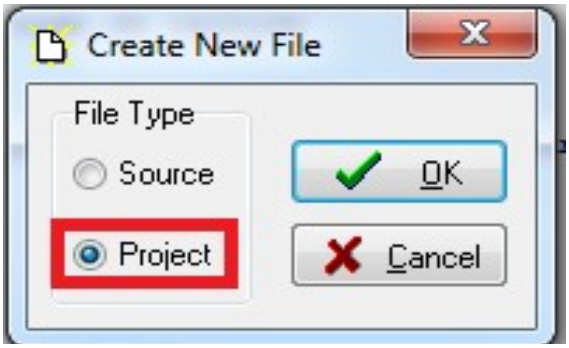

Illustration 12: CodeVision Initialisation 1 [3]

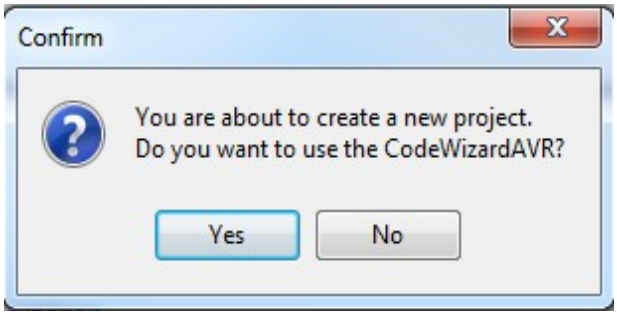

Illustration 13: CodeVision Initialisation 2 [3]

Une nouvelle fenêtre va s'ouvrir et va demander de choisir tous les composants qui seront connectés au micro-contrôleur.

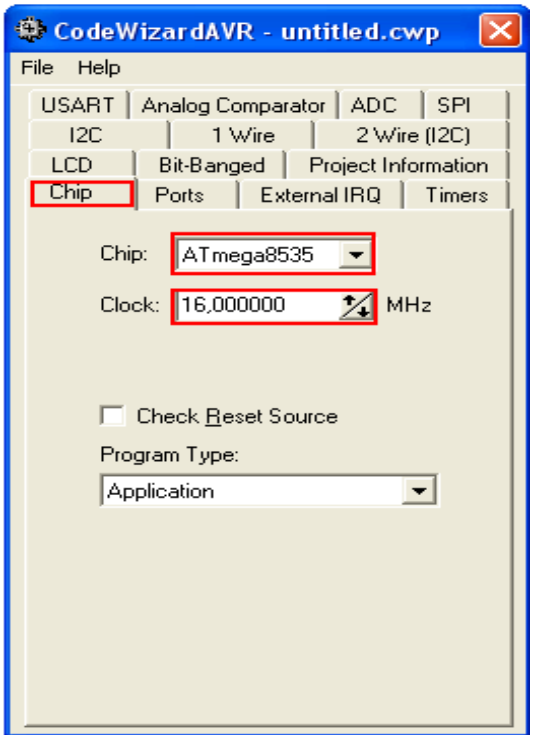

Illustration 14: CodeVision Initialisation 3 [3]

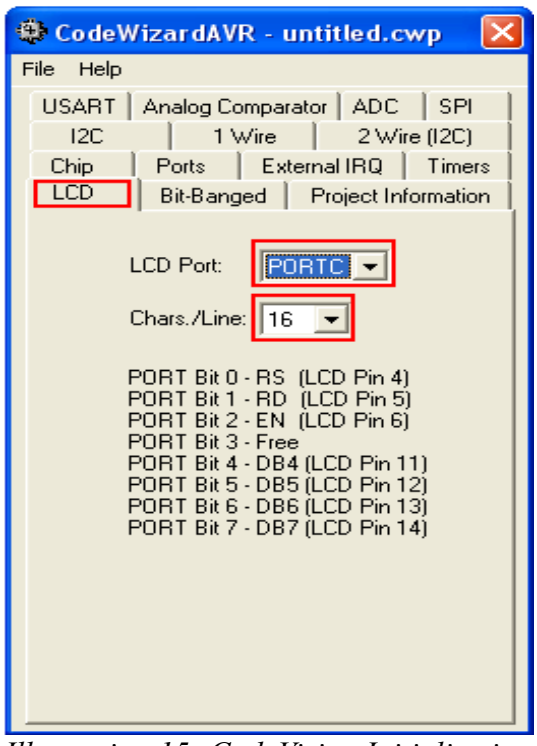

Illustration 15: CodeVision Initialisation 4 [3]

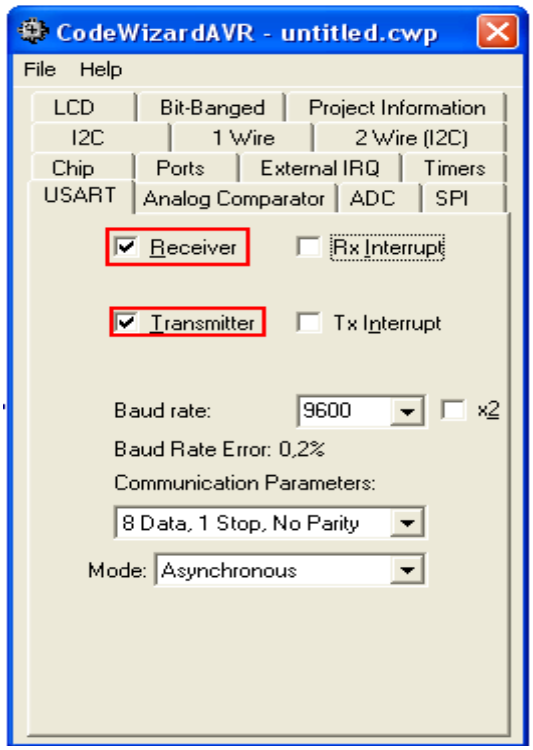

Illustration 16: CodeVision Initialisation 5 [3]

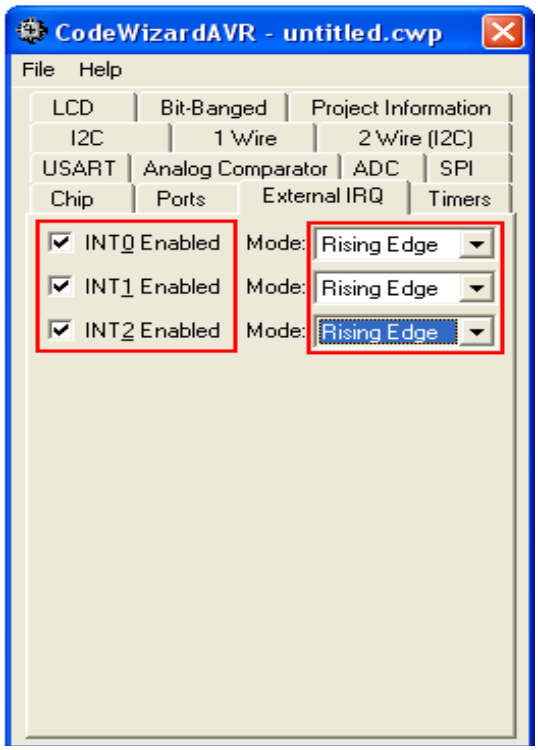

Illustration 17: CodeVision Initialisation 6 [3]

Nous avons choisi un ATMEGA 8535 avec une horloge 16 MHz, un écran LCD que nous avons connecté sur le PORTC. Il faut faire attention à bien régler l'écriture et la lecture entre l'ordinateur et le micro-contrôleur. Enfin nous utiliserons les interruptions uniquement pour la partie d'écriture, pour l'horloge et pour sa remise à zéro. Ça n'est d'aucune utilité pour la partie lecture.

Il faut demander à CodeVision de générer le code, pour cela « File » → « Generate, Save and Exit ».

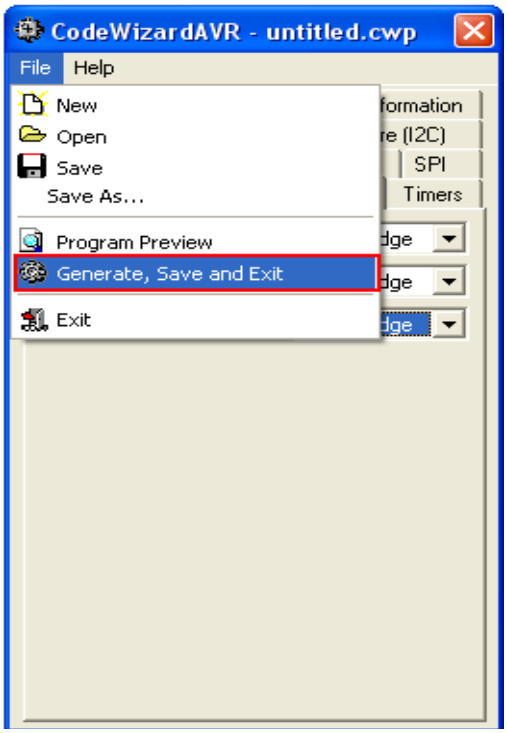

Illustration 18: CodeVision Initialisation 7 [3]

Il y a alors 3 fichiers à sauvegarder, le .c, le .prj et le .cwp. Afin d'éviter tout problème de compilation avec CodeVision, les 3 fichiers doivent être sauvegardés sous des noms différents.

On peut également sélectionner « Program Preview » qui permet une visualisation du code d'initialisation sans effacer notre code actuel, si on rajoute un composant par exemple.

3. Réalisation

La programmation de l'ATMEGA 8535 est faite à partir de CodeVision AVR, un logiciel de programmation en C payant.

3.1. Routage sur Orcad

Le prototype réalisé par Vivien MARTINEZ a permis d'effectuer une série de tests au groupe de l'an dernier et ainsi de créer des versions finales fiables. Les librairies et les empreintes des composants utilisés ont été fabriqués par M, LEQUEU et M, BRAULT.

3.1.1. Schéma d'alimentation

Voici donc le schéma électrique de l'alimentation qui permet d'obtenir la tension nécessaire à la carte 5 volts. Pour protéger le circuit, un fusible 100mA a été ajouté.

La diode verte (notée D3) montre le bon fonctionnement de celle-ci.

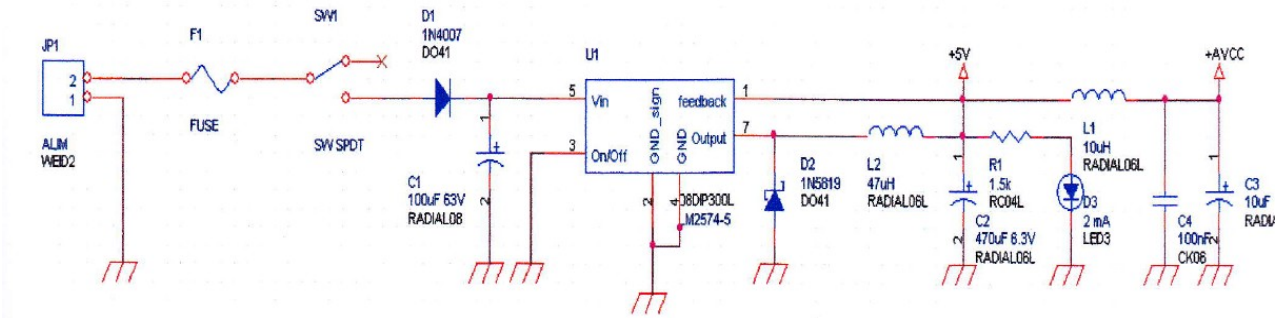

Illustration 19: Schéma électrique du régulateur de tension[6]

3.1.2. La partie ATMEGA 8535 et sélection de borne

Le micro-contrôleur possède le schéma électrique le plus complexe. En effet, il est le centre de la carte et gère tous les composants.

En bas à droite du schéma, le switch (sélection de la borne) permet de sélectionner le numéro de la borne. Il est relié au quatre premiers bits du port A de l'ATMEGA. Entre les deux, le réseau de résistances permet de définir le niveau de détection des bits. Ici l'état 1.

Juste au-dessus, un buzzer qui est connecté au port A. Grâce à son signal sonore, il définit la fin de la détection.

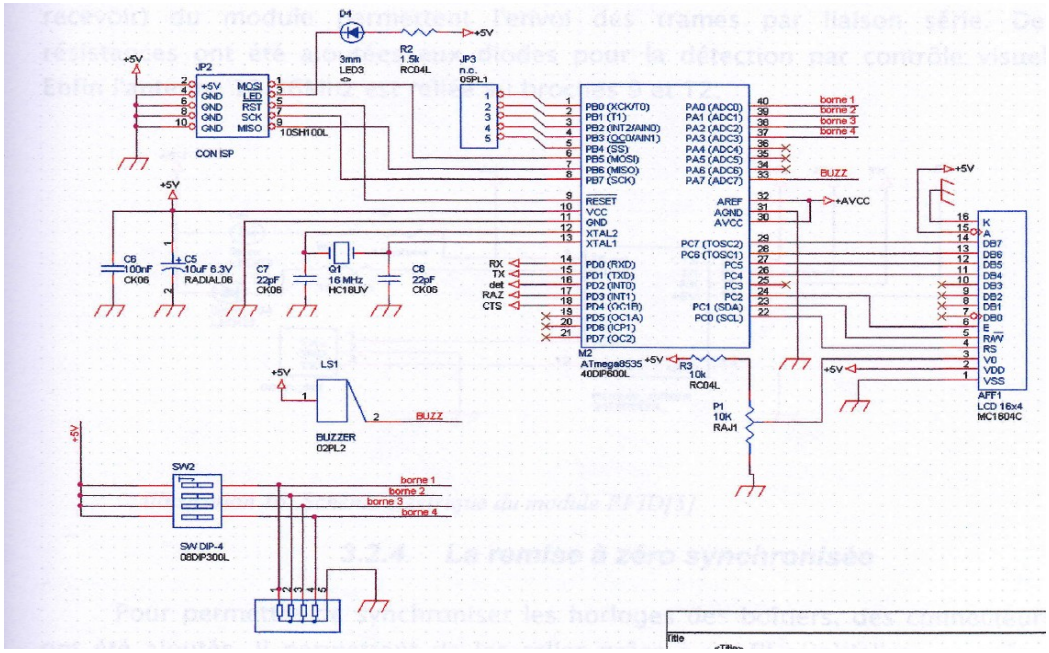

Illustration 20: Schéma électrique de l'ATMEGA 8535[6]

3.1.3. Le schéma du RFID

Les broches 22 à 24 (Rx : Réception, Tx: Transmission et CTS : prêt à recevoir) du module permettent l'envoi des trames par liaison série. Des résistances ont été ajoutées aux diodes pour la détection par contrôle visuel. Enfin l'antenne 13,56 MHz est reliée aux broches 9 et 12.

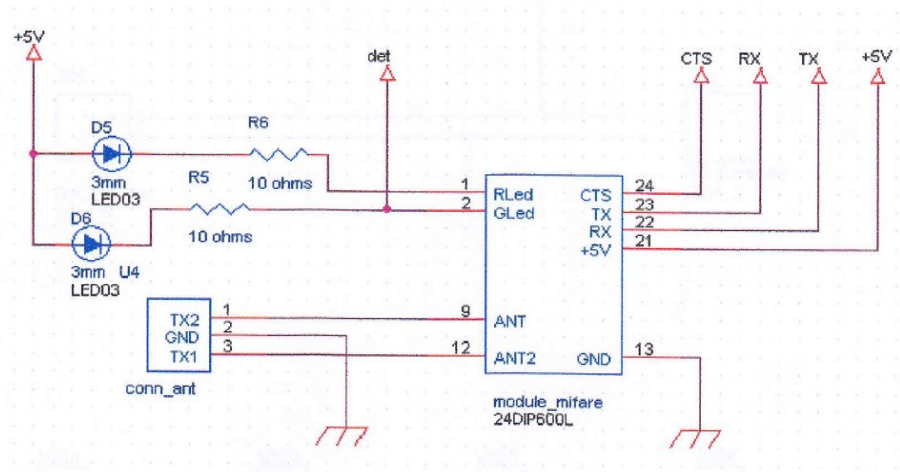

Illustration 21: Schéma électrique du module RFID[6]

3.1.4. La remise à zéro synchronisé

Pour permettre de synchroniser les horloges des boîtiers, des connecteurs ont été ajoutés. Ils permettent de les relier grâce à un fil réinitialiser en même temps que leurs horloges en appuyant sur le bouton poussoir (belge sur nos bornes). Le signale noté RAZ sur le schéma est connecté à l'entrée d'interruption INT1 de l'ATMEGA (voir programmation).

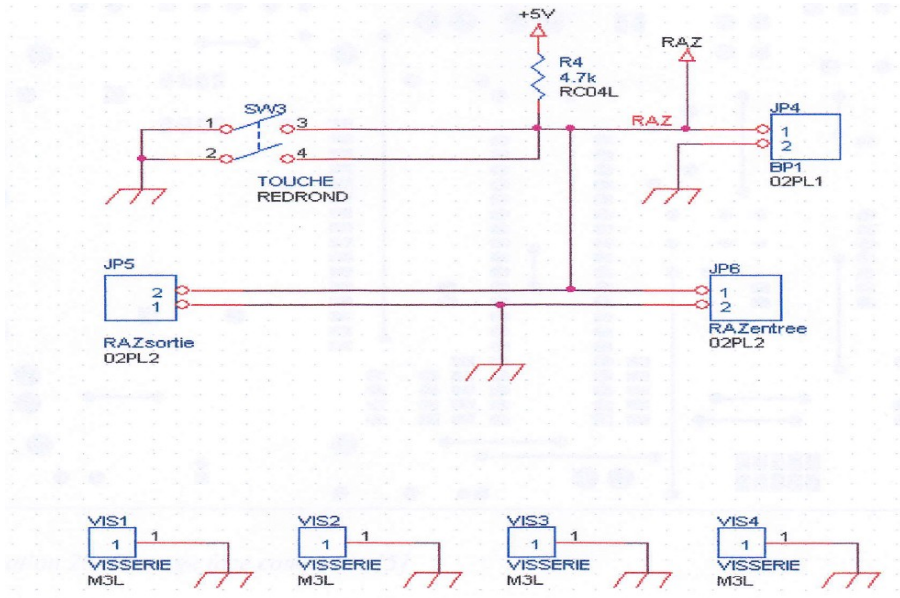

Illustration 22: Schéma électrique de la remise à zéro synchronisée[6]

3.1.5. Le typon

Après avoir effectué différents réglages, ils obtiennent le typon final. Pour l'obtenir, il a fallut créer une nouvelle empreinte pour le module RFID car ses caractéristiques sont spécifiques.

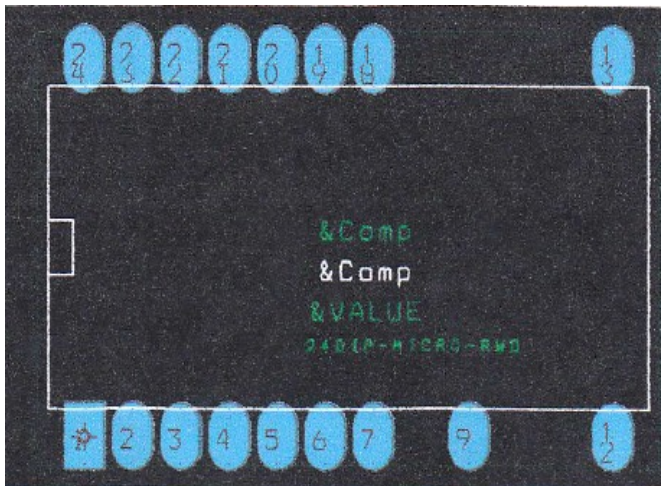

Illustration 23: Empreinte du module RFID[0]

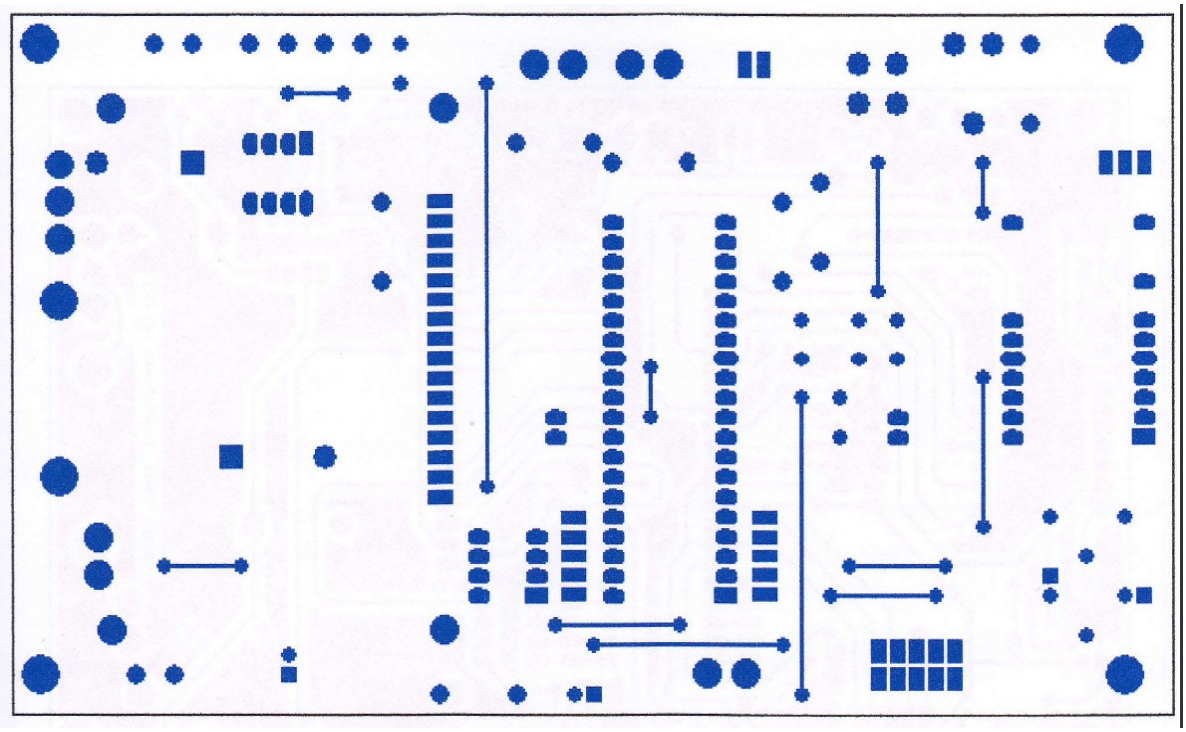

Illustration 24: Routage face composant[6]

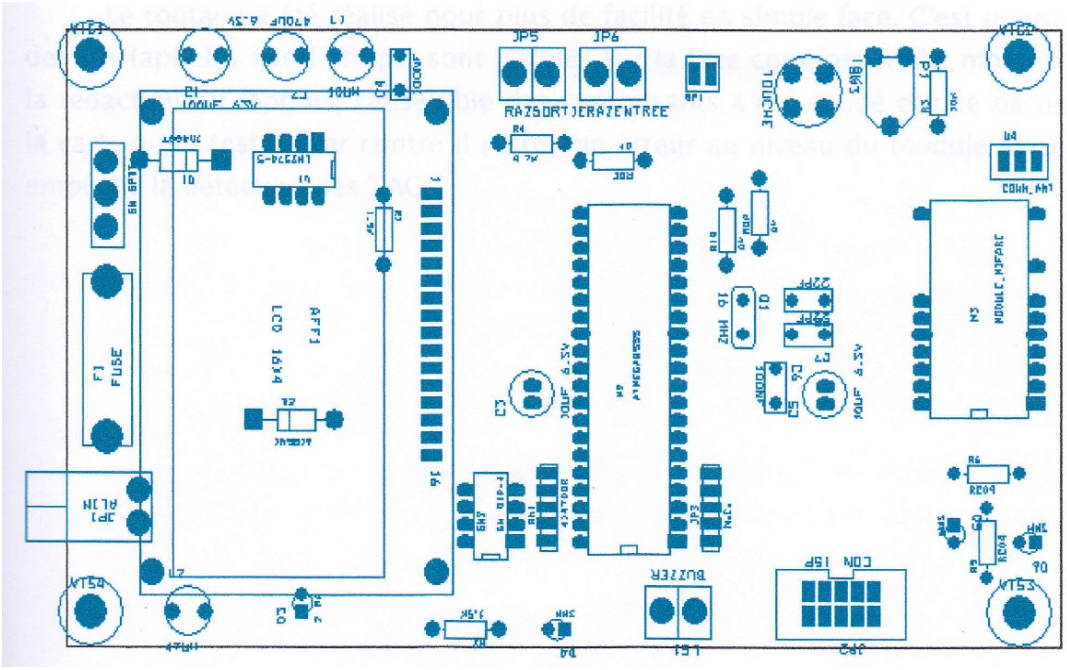

Illustration 25: Routage côté composant[6]

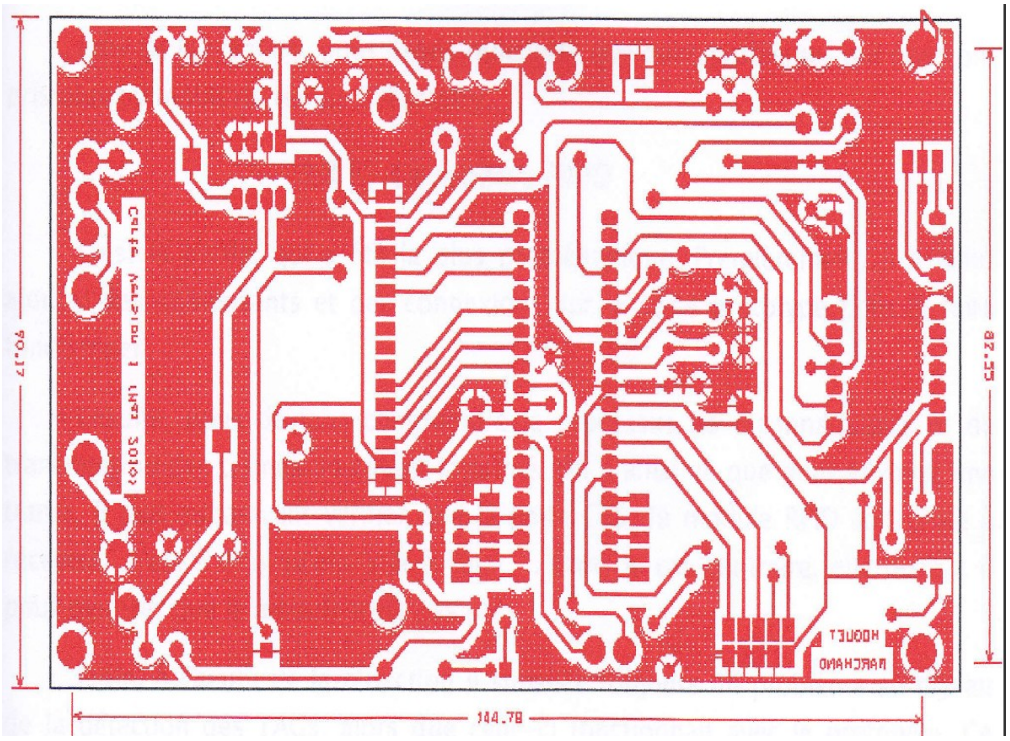

Illustration 26: Routage côté cuivre[6]

3.2. Programmation de la partie écriture

L'intégrité des programmes est disponible en annexe 3

3.2.1. L'afficheur du numéro de bornes

On déclare dans un premier temps les variables permettant de gérer l'affichage du numéro de bornes. Ces variables sont définies en « variables locales » c'est à dire qu'elles ne peuvent être utilisées seulement dans la fonction où elles sont utilisées.

unsigned char NumeroBorneHexa;

int Dizaine, Unite, NumeroBorne;

int PresenceCarte, z;

La variable *« z »* étant utilisé afin de réaliser une boucle qui nous permettra l'affichage de l'IUD.

On récupère ensuite les informations envoyées par l'interrupteur à 4 positions. Il a fallu créer un masque pour ne lire que les entrées intéressantes.

NumeroBorneHexa = (PINA&0x0f);

On procède ensuite à la conversion du numéro de borne en décimal

Dizaine=NumeroBorneHexa/10;

Unite=((int)NumeroBorneHexa)%10;

*NumeroBorne=Dizaine*10+Unite;*

Enfin, on écrit ce numéro de borne en décimal sur la première ligne de l'afficheur.

sprintf(tampon,"N borne : 0x%x", NumeroBorneHexa);

lcd_gotoxy(0,0);

lcd_puts(tampon);

On peut alors écrire sur l'afficheur une valeur de 1 à 15 permettant de différencier 15 bornes.

3.2.2. Mise en place de l'horloge interne

Pour mettre en place l'horloge, il faut configurer une interruption interne. Pour la déclarer, on écrit les lignes suivantes dans la fonction principale (fonction main( )):

// 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; // Configuration de la valeur de comparaison

TCCR1B=0x0A;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x4E; // Sélectionne la base de temps sur une fréquence de 2 MHz, soit un top d'horloge toutes les 0,5 us.

OCR1AL=0x20; // Une interruption quand on arrive à 20 000 top d'horloge (0x4E20 soit 10 ms)

OCR1BH=0x00;

OCR1BL=0x00;

Attention de ne pas oublier ces déclarations :

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x10;

et

// Global enable interrupts

#asm("sei")

On déclare enfin la fonction interruption permettant de compter les centièmes de secondes (variables « Temps »), les secondes, les minutes et les heures.

// Mise en place de l'interruption interne 2000 KHz permettant d'établir le temps

```
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
    Temps++;
    if (Temps>=100)
    {
        Temps=0;
        Seconde++;
        if(Seconde>=10)
        {
            Seconde=0;
            Minute++;
            if(Minute>=3)
            {
                Minute=0;
                Heure++;
                if(Heure>=99)
                {
                    Heure=0;
                };
            };
        };
    };
}
```

Attention, pour pouvoir être récupérées facilement, les variables Temps, Seconde, Minute, Heure sont déclarées en tout début de programme en tant que variables globales.

```
// Declare your global variables here
unsigned char Temps, Seconde, Minute, Heure;
```

3.2.3. Vérification du statut de la carte

On décide de vérifier que la carte RFID communique correctement avec l'ATMEGA avant d'y envoyer des informations. On affiche ensuite sur l'écran LCD si la carte est correctement reconnue ou non. L'envoi ou la réception d'informations réalisées sans connexion avec la puce RFID provoque un blocage du programme.

```
int VerifStatut (void)
{
    int Presence;
    while(CTS==1);     //Attente que l'entrée CTS soit active (à 0)
    trame='S';       //ou 0x53 : permet de vérifier le statut
    USART_Transmit(trame);
    Statut = USART_Receive();

    if(Statut==0x96)      //Bit d'identification de la carte
    {
        Presence=1;
        sprintf(tampon,"Lecture  ");
        lcd_gotoxy(0,2);
        lcd_puts(tampon);
    }
    else
    {
        Presence=0;
        sprintf(tampon,"Pas Carte");
        lcd_gotoxy(0,2);
        lcd_puts(tampon);
    }
    return Presence;
}
```

On vérifie la bonne communication entre la carte et le micro-contrôleur avant d'autoriser l'envoi des informations vers la carte afin d'éviter tout « plantage » de l'émetteur.

3.2.4. Écriture du temps de passage dans le TAG

Il faut faire attention à ce que le module RFID soir prêt à recevoir les données envoyées. C'est pourquoi avant chaque envoi, on attend que l'entrée « CTS » soit à 1. Il faut également faire attention à bien envoyer 19 bits au total (18 arguments plus la commande). On récupère au préalable le numéro de borne dans la fonction principale.

```
void EcritureTemps(int Borne)
{
    int z;

    while(CTS==1);
    trame='W';     //ou 0x57 : permet de lire l'UID
    USART_Transmit(trame);

    while(CTS==1);
    trame=Borne*18;     //Écriture à l'adresse du numéro de borne
    USART_Transmit(trame);

    while(CTS==1);
     trame=0x00;     //Clé de sécurité à choisir entre 0 et 31 sur les 5 premiers bits de l'octet
    USART_Transmit(trame);

    //Écriture de l'identifiant
    for(z=0; z<4;z++)
    {
        trame = Identifiant[z];
        USART_Transmit(trame);
    }
```

```
        //Écriture du temps de passage
        while(CTS==1);
        trame=Temps;
        USART_Transmit(trame);

        while(CTS==1);
        trame=Seconde;
        USART_Transmit(trame);

        while(CTS==1);
        trame=Minute;
        USART_Transmit(trame);

        while(CTS==1);
        trame=Heure;
        USART_Transmit(trame);

        //Écriture dans les derniers bits de la carte RFID
        sprintf(trame, "");      //Initialisation de la variable trame
        for(z=0;z<8;z++)
        {
                while(CTS==1);
                USART_Transmit(trame);
         }
        sprintf(Information, "");        //Initialisation de la variable Information
        Information = USART_Receive();    //Récupération de l'acknowledge afin de vérifier que
                                          l'écriture s'est faite correctement
}
```

Une donnée sera stockée sur chaque octet. Nous enregistrons donc chaque temps de borne toutes les 18 adresses, correspondant aux 18 bits envoyés. Si nous ne faisons pas cela, les temps s'effaceront au fur et à mesure que nous écriront sur la carte.

Le numéro de borne va sélectionner la partie de la carte où écrire le temps de passage. Ainsi, la borne d'arrivée n'écrasera pas par exemple, les informations fournies par la borne de départ.

3.3. Programmation de la partie lecture

3.3.1. Affichage des données

Comme pour la partie écriture, on décide d'afficher en premier lieu le statut de la carte ainsi que la borne. Puis on écrit les 8 premiers bits lu sur la carte: les 4 premiers correspondront à l'UID de la carte, les 4 suivants correspondront aux temps (Temps, Seconde, Minute, Heure).

```
for(z=0;z<4;z++)              //Lecture des 8 bits affiché sur 2 lignes
{
        sprintf(tampon, "%3d", Temps[z]);   //4 premiers bits = UID
        lcd_gotoxy(z*4, 1);
        lcd_puts(tampon);

        sprintf(tampon, "%3d", Temps[z+4]);         //bits qui nous intéresse
        lcd_gotoxy(z*4, 2);
        lcd_puts(tampon);
}
```

Sur l'affichage de l'écran LCD, on multiplie le variable de l'axe horizontal par 4, ce qui correspond aux nombre de cases qu'on réserve pour afficher la valeur, afin d'éviter les soucis de rafraichissement de l'écran. Par exemple pour l'affichage de l'UID, on prend une carte dont la première valeur est 128, on n'aura aucun souci. Mais si on prend ensuite une deuxième carte dont le premier bit de l'UID est 64, alors, si on ne multiplie pas par 4, on verra s'afficher 164 sur l'écran LCD.

3.3.2. Lecture du temps de passage

Comme pour l'émission, il faire attention à ce que le module RFID soit prêt à recevoir les données envoyées, c'est à dire que l'entrée « CTS » soit à 1. On place un masque afin de ne récupérer de l'acknowledge que les informations qui nous intéressent.

```
void Recoit_Info (int Borne)
{
    int i;
    unsigned char car;
    i=0;
    while(CTS==1);

    trame='R';                          //permet de lire la trame
    USART_Transmit(trame);              //Transmission de 'R' ou 0x52
    USART_Transmit(Borne*18);           //Transmission de l'adresse de location
    USART_Transmit(0x00);               //Transmission du code de lecture
    Information = USART_Receive();

    Information = (Information & 0xCF); //Mise en place d'un masque pour lecture
                                        nécessaire du Acknowledge
    if(Information == 0x86)
    {
        //Temps[1] = USART_Receive();

        while(i<=15)                    //16 bits à récupérer
        {
            car = USART_Receive();
            Temps[i] = car;
            i++;
        }
    }
```

```
    //lcd_clear();  //Effacer l'écran LCD
}
```

Lors de la réception, nous envoyons 2 bits correspondant à l'adresse et à son code associé afin d'y récupérer nos 16 bits de données présent sur la carte.

3.4. Problèmes rencontrés

Lors de notre projet, nous avons rencontré quelques problèmes qui nous ont pris plus ou moins de temps à résoudre.

3.4.1. Le prototype

C'est ce qui nous a fait perdre le plus de temps. Certains composants comme l'antenne ou le module RFID sont des éléments très fragiles. Une autre personne de licence pro travaillait également sur la RFID, les prototypes voyageaient donc beaucoup entre les salles. De ce fait, plusieurs composants sont devenus défectueux et des soudures étaient à refaire régulièrement. Nous avons changé beaucoup de composants, au fil des semaines.

Finalement, une fois les cartes fonctionnelles avec les programmes transférés dans les micro-contrôleurs, nous avons rangé et fixé les prototypes dans des boitiers afin d'éviter ce genre de problème à l'avenir.

3.4.2. La programmation

C'était la première fois que nous programmions sur le logiciel CodeVision AVR. Il nous a fallu un peu de temps pour le prendre en main.

Nous avions récupérer le programme du binôme de l'an dernier ce qui nous avez un peu aidé. Nous avons eu également quelques difficulté à comprendre le principe du CTS et des flags afin d'envoyer, de recevoir et de savoir si tout cela s'est bien fait correctement.

Conclusion

Lors de ce projet, nous avons mis en application toutes les connaissance acquises durant les 3 semestres pendant nos cours d'informatique industrielle. Il nous a également fallu comprendre le principe de la RFID, et récupérer les typons et cartes des anciens binômes.

Le projet a été début par Vivien MARTINEZ, puis continué par le binôme HOGUET Romain et MARCHAND Rémi. Ce sont eux qui ont effectué l'étude électronique et ont réalisé des prototypes fonctionnels.

Ce projet nécessite une compréhension sur le fonctionnement de chaque composant afin de pouvoir les programmer. L'étude des schémas électriques, la réalisation des typons et enfin, la partie programmation en utilisant la logiciel Code Vision AVR nous ont posé beaucoup de problèmes au cours des premières séances. C'est pour cela que la lecture du datasheets et les essais pratiques sont indispensables pour y résoudre.

Ce projet nous a permis de mettre en application de façon concrète et utile ce que nous avons appris. Il nous a permis également à apprendre à récupérer des sujets laissés par autrui. C'est ce qui se rapproche, pour nous, d'une véritable expérience professionnelle.

Résumé

La technologie RFID est une technologie analogue au code barre. Elle permet dans notre cas, d'établir un chronométrage et un classement de chaque participant lors d'une manifestation sportive. Un badge nominatif sera présenté par le couruer à des points stratégiques (départ, arrivée,...), devant une borne d'écriture.

Le but a donc été de créer et de programmer ces boîtiers capable de communiquer les temps de passage à distance par la liaison RFID. Dans un premier temps, le prototype de HOGUET Romain et MARCHAND Rémi a été récupéré pour analyse et programmation. Nos séances avaient pour but de faire fonctionner une horloge, d'écrire et de lire dans les badges RFID et de différencier les bornes de départ, d'intermédiaire et d'arrivée.

Les cartes électroniques de chaque borne sont conçues et programmées de la même façon. Chacune possède un régulateur de tension, un module RFID, un micro-contrôleur et un switch qui permettra de différencier les bornes en leur attribuant une valeur en 1 et 15.

Après avoir résolu les problèmes auxquels nous avons été confrontés, nous avons établi nos propres programmes d'écriture et de lecture des badges. A ce jour, les cartes et les programmes sont fonctionnels et prête à utiliser par l'association sportive.

231 mots

Index des illustrations

Illustration 1: exemple d'émetteur RFID[1].....5
Illustration 2: Le boitier[2].....6
Illustration 3: Badge RFID[3].....7
Illustration 4: Modules RFID et leurs antennes[2].....9
Illustration 5: Schéma global du projet[5].....10
Illustration 6: Schéma fonctionnel de niveau 1 [5].....10
Illustration 7: Schéma fonctionnel de niveau 2 [5].....11
Illustration 8: La carte finie[3].....12
Illustration 9: ATMEGA 8535[2].....15
Illustration 10: Régulateur de tension LM2574N-5 [2].....15
Illustration 11: Module CMS RFID 13,56 MHz [2].....16
Illustration 12: CodeVision Initialisation 1 [3].....17
Illustration 13: CodeVision Initialisation 2 [3].....17
Illustration 14: CodeVision Initialisation 3 [3].....18
Illustration 15: CodeVision Initialisation 4 [3].....18
Illustration 16: CodeVision Initialisation 5 [3].....18
Illustration 17: CodeVision Initialisation 6 [3].....18
Illustration 18: CodeVision Initialisation 7 [3].....19
Illustration 19: Schéma électrique du régulateur de tension[6].....20
Illustration 20: Schéma électrique de l'ATMEGA 8535[6].....21
Illustration 21: Schéma électrique du module RFID[6].....22
Illustration 22: Schéma électrique de la remise à zéro synchronisée[6].....23
Illustration 23: Empreinte du module RFID[0].....24
Illustration 24: Routage face composant[6].....24
Illustration 25: Routage côté composant[6].....25
Illustration 26: Routage côté cuivre[6].....25

Bibliographie

[1] **DAG Europe**. *Chronométrage RFID*, , [En ligne]. (Page consultée le) <http://eu.dag-system.com/Product.aspx?Id=12>
[2] **HOGUET/MARCHAND**, "*Photographies*", , 2010.
[3] **ABDULLAH/SAPIENS**, "*Photos personnelles*", , 2011.
[4] **ABDULLAH/SAPIENS**, "*Tableau personnel*", , 2011.
[5] **MARTINEZ Vivien**. Le chronométrage RFID. **,** 2010, N°, p. .
[6] **HOGUET/MARCHAND**, "*Schéma*", , 2010.
[0] **LEQUEU Thierry**, "*Empreinte spécifique sous Orcad*", , 2010.

Annexes

Annexe 1 : L'implantation du module RFID

Annexe 2 : Datasheet du module RWD MIFARE

Annexe 3 : Programme complet de gestion des bornes d'écritures

Annexe 4 : Programme complet de lecture des carte RFID

Annexe 5 : Fiche de suivi de projet

Annexe 3 : Programme complet de gestion des bornes d'écritures

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

Project : Mesure du temps de parcours par RFID
Version : 10
Date    : 04/01/2011
Author  : BINTI ABDULLAH Akmal & SAPIENS Corentin
Company : Thierry LEQUEU
Comments: Programmation Borne d'écriture

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>
/* the LCD module is connected to PORTC */
#asm
   .equ __lcd_port=0x15
#endasm
```

```
/* now you can include the LCD Functions */
#include <lcd.h>
#include <delay.h>     // pour la fonction delay_ms();
#include <stdio.h>     // pour la fonction sprintf();

#define CTS     PIND.4
#define BUZZER  PINA.7

// Declare your global variables here

unsigned char tampon[20], Temps1[20], Identifiant[8];
unsigned char trame, Statut, Information;
unsigned char Temps, Seconde, Minute, Heure;


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

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;
}

// Permet de vérifier le Statut de la carte

int VerifStatut (void)
{
        int Presence;
        while(CTS==1);     //Attente que l'entrée CTS soit active (à 0)
      trame='S';       //ou 0x53 : permet de vérifier le statut
      USART_Transmit(trame);
      Statut = USART_Receive();

        if(Statut==0x96)      //Bit d'identification de la carte
      {
        Presence=1;
        sprintf(tampon,"Lecture  ");
        lcd_gotoxy(0,2);
        lcd_puts(tampon);
      }
      else
      {
        Presence=0;
        sprintf(tampon,"Pas Carte");
        lcd_gotoxy(0,2);
        lcd_puts(tampon);
      }
   return Presence;
```

```
}

void Recoit_UID (void)
{
   int i;
   unsigned char car;;
   i=0;
   while(CTS==1);
   trame='U',     //ou 0x55 : permet de lire l'UID
   USART_Transmit(trame);
   Statut = USART_Receive();

   while(i<=6)     // 6 pour les MIFARE 1k/4k card types
   {
      car = USART_Receive();
      Identifiant[i]=car;
      i++;
   }
}

// Mise en place de l'interruption interne 2000 KHz permettant d'établir le temps

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
        Temps++;
        if (Temps>=100)
        {
                Temps=0;
                Seconde++;
```

```
            if(Seconde>=10)
            {
                Seconde=0;
                Minute++;
                if(Minute>=3)
                {
                    Minute=0;
                    Heure++;
                    if(Heure>=99)
                    {
                        Heure=0;
                    };
                };
            };
        };
}

void Horloge(void)
{
   sprintf(Temps1,"%2dh %2dmin %2ds", Heure, Minute, Seconde);
   lcd_gotoxy(0,3);
   lcd_puts(Temps1);
}


//Remise à zéro synchronisée
interrupt [EXT_INT1] void ext_int1_isr(void)
{
   Temps=0;
```

```
   Seconde=0;
   Minute=0;
   Heure=0;
}


//Déclenchement à la detection d'une carte
interrupt [EXT_INT0] void ext_int0_isr(void)
{

}

void EcritureTemps(int Borne)
{
   int z;

   while(CTS==1);
   trame='W';     //ou 0x57 : permet de lire l'UID
   USART_Transmit(trame);

   while(CTS==1);
   trame=Borne*18;     //Ecriture à l'adresse du numéro de borne
   USART_Transmit(trame);

   while(CTS==1);
   trame=0x00;     //Clé de sécurité à choisir entre 0 et 31 sur les 5 premiers bits de l'octet
   USART_Transmit(trame);

   for(z=0; z<4;z++)
```

```
{
        trame = Identifiant[z];
        USART_Transmit(trame);
}

while(CTS==1);
trame=Temps;
USART_Transmit(trame);

while(CTS==1);
trame=Seconde;
USART_Transmit(trame);

while(CTS==1);
trame=Minute;
USART_Transmit(trame);

while(CTS==1);
trame=Heure;
USART_Transmit(trame);

sprintf(trame, "");
for(z=0;z<8;z++)
{
        while(CTS==1);
        USART_Transmit(trame);
}

    sprintf(Information, "");
```

```
        Information = USART_Receive();



}



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

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0xF0;

// 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=0x00;
DDRB=0x00;

// 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;

// Timer/Counter 0 initializatio_n
// 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; // Configuration de la valeur de comparaison
TCCR1B=0x0A;
```

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x4E; // Sélectionne la base de temps sur une fréquence de 2 MHz, soit un top d'horloge toutes les 0,5 us.

OCR1AL=0x20; // Une interruption quand on arrive à 20 000 top d'horloge (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

TCCR2=0x00;

TCNT2=0x00;

OCR2=0x00;

// External Interrupt(s) initialization

// INT0: On

// INT0 Mode: Falling Edge

// INT1: On

// INT1 Mode: Falling Edge

// INT2: Off

GICR|=0xC0;

MCUCR=0x0A;

MCUCSR=0x00;

GIFR=0xC0;

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

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

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

/* initialize the LCD for 2 lines & 16 columns */
lcd_init(16);

/* switch to writing in Display RAM */
//lcd_gotoxy(0,0);
//lcd_putsf("Projet Borne RFID");
```

```
// initialisation des variables globales
Temps=0;
Seconde=0;
Minute=0;
Heure=0;
BUZZER=1;

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



while (1)
{
   //Déclaration des variables locales

   unsigned char NumeroBorneHexa;
   int Dizaine, Unite, NumeroBorne;
   int PresenceCarte, z;

   // Appelle de L'horloge
   Horloge();

   //Changement du numéro de borne de 0 à 9

   NumeroBorneHexa = (PINA&0x0f);
   /*Dizaine=NumeroBorneHexa/10;          //Conversion du numéro de bornes Hexadécimal en
Décimal
   Unite=((int)NumeroBorneHexa)%10;
   NumeroBorne=Dizaine*10+Unite;*/
```

```
sprintf(tampon,"N borne : 0x%x", NumeroBorneHexa);
lcd_gotoxy(0,0);
lcd_puts(tampon);

//Vérification du statut de la carte

PresenceCarte = VerifStatut ();   // Appelle de la fonction permettant de vérifier le statut

if(PresenceCarte==1)
{
   /* Possiblité de lire l'identifiant unique de la carte     */

   Recoit_UID();
   for(z=0;z<4;z++)
   {
      sprintf(tampon,"%3d",Identifiant[z]);
      lcd_gotoxy(z*4,1);
      lcd_puts(tampon);
   }

   // Ecriture du temps de passage sur la carte

   EcritureTemps(NumeroBorneHexa);
   if(Information == 0x96)
      sprintf(tampon,"OK");
   else
      sprintf(tampon, "Erreur");

      lcd_gotoxy(11,2);
```

```
        lcd_puts(tampon);


       //on attend 500 ms et on déclenche le buzzer pour marquer la fin de la lecture il faut faire attention de
     //ne pas figer l'affichage de l'horloge

     BUZZER=0;      //logique inversée

     delay_ms(100);
     Horloge();

     delay_ms(100);
     Horloge();

     delay_ms(100);
     Horloge();

     delay_ms(100);
     Horloge();

     delay_ms(100);

     BUZZER=1;
   }
}
}
```

Annexe 4 : Programme complet de lecture des carte RFID

```
/*****************************************************
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 : Mesure du temps de parcours par RFID
Version : 3
Date    : 04/01/2011
Author  : BINTI ABDULLAH Akmal & SAPIENS Corentin
Company : Thierry LEQUEU
Comments: Programmation Borne de lecture


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>

// Alphanumeric LCD Module functions
#asm
```

```
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <delay.h>     // pour la fonction delay_ms();
#include <stdlib.h>

// Standard Input/Output functions
#include <stdio.h>

#define CTS     PIND.4
#define BUZZER  PINA.7

// Declare your global variables here

unsigned char trame, Statut, Information;
unsigned char Temps[20];
unsigned char tampon[20], Identifiant[10];

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

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;
}

int VerifStatut (void)
{
        int Presence;
        while(CTS==1);     //Attente que l'entrée CTS soit active (à 0)
      trame='S';       //ou 0x53 : permet de vérifier le statut
      USART_Transmit(trame);
      Statut = USART_Receive();

        if(Statut==0x96)      //Bit d'identification de la carte
      {
        Presence=1;
        sprintf(tampon,"Lecture  ");
        lcd_gotoxy(0,0);
        lcd_puts(tampon);
      }
      else
      {
        Presence=0;
        sprintf(tampon,"Pas Carte");
        lcd_gotoxy(0,0);
        lcd_puts(tampon);
      }
   return Presence;
```

```
}

void Recoit_UID (void)
{
   int i, z;

   unsigned char car;
   i=0;
   for(z=0;z<7;z++)
   {
        Identifiant[z] = 0;
   }
   while(CTS==1);
   trame='U',     //ou 0x55 : permet de lire l'UID
   USART_Transmit(trame);
   Statut = USART_Receive();

   if(Statut == 150)
   {
        while(i<=6)     // 6 pour les MIFARE 1k/4k card types
        {
        car = USART_Receive();
        Identifiant[i]=car;
        i++;
        }
   }

}
```

```
void Recoit_Info (int Borne)
{
   int i;

   unsigned char car;
   i=0;
   while(CTS==1);

   trame='R';     //permet de lire la trame
   USART_Transmit(trame);  //Transmission de 'R' ou 0x52
   USART_Transmit(Borne*18);       //Transmission de l'adresse de location
   USART_Transmit(0x00);           //Transmission du code de lecture
   Information = USART_Receive();

   Information = (Information & 0xCF);      //Mise en place d'un masque pour lecture nécessaire du
Acknowledge


   if(Information == 0x86)
   {
        //Temps[1] = USART_Receive();

        while(i<=15)            //16 bits à récupérer
        {
                car = USART_Receive();
                Temps[i] = car;
                i++;
        }
   }
```

```
        //lcd_clear();
}

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;

// 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=0x00;
DDRB=0x00;

// 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=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;

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 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=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;

// 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);

while (1)
{
      unsigned char NumeroBorneHexa;
         int z, PresenceCarte;
         PresenceCarte = VerifStatut ();   // Appelle de la fonction permettant de vérifier le statut

         NumeroBorneHexa = (PINA&0x0f);

                  sprintf(tampon, "%x", NumeroBorneHexa);
                  lcd_gotoxy(12, 0);
                  lcd_puts(tampon);
         if(PresenceCarte==1)
         {
                  // Possiblité de lire l'identifiant unique de la carte
```

```
            //Recoit_UID();


            for(z=0;z<4;z++)
        {
            sprintf(tampon,"%3d",Identifiant[z]);
                    lcd_gotoxy(z*4,1);
                    lcd_puts(tampon);
        }
        Recoit_Info(NumeroBorneHexa);
            /*sprintf(tampon, "0x%x", Information);      //Affichage du Acknowledge de reception
des 16 bits (avec masque)
            lcd_gotoxy(0,3);
            lcd_puts(tampon);*/


            for(z=0;z<4;z++)              //Lecture complète des 16 bits affiché sur 4 lignes
            {
                    sprintf(tampon, "%3d", Temps[z]);   //4 premiers bits = UID
                    lcd_gotoxy(z*4, 1);
                    lcd_puts(tampon);


                    sprintf(tampon, "%3d", Temps[z+4]);         //bits qui nous intéresse
                    lcd_gotoxy((z)*4, 2);
                    lcd_puts(tampon);


                    sprintf(tampon, "%3d", Temps[z+8]);
                    lcd_gotoxy((z)*4, 3);
                    lcd_puts(tampon);
                    /*
                    sprintf(tampon, "%3d", Temps[z+12]);
```

```
                    lcd_gotoxy((z)*4, 3);
                    lcd_puts(tampon);*/
                }


        }


}
}
```

Annexe 5 : Fiche de suivi de projet

Semaine 38 (20/9/2010)

- Explication par M, LEQUEU du déroulement des séances de projet,
- Proposition des sujets par M. LEQUEU. Nous avons choisi de continuer le projet « mesurer le temps de parcours par RFID » commencé par Vivien MARTINEZ, puis continué par le binôme Thomas HOGUET et Rémi MARCHAND.
- Compréhension général du sujet

Semaine 39 (27/9/2010)

- Lecture des différentes datasheets et trouver les modification à apporter

Semaine 40 (4/10/2010)

- Présentation du logiciel de routage du typon Orcad Layout par M.LEQUEU.

Semaine 41 (11/10/2010)

- Vérification des anciennes cartes (l'une de cartes ne fonctionne pas)
- Réalisation des essais avec le programme d'écriture réalisé par l'ancien binôme.

Semaine 42 (18/10/2010)

- Commencer les programmes d'écriture et de lecture

Semaine 45 (12/11/2010)

- Initialisation de AVR
- Définition de bibliothèque
- Définition d'entrée/sortie

Semaine 46 (15/11/2010)

- Traitement de la partie réception

  → On a du revoir la datasheet pour faire la partie réception

  → Problème de masque et status flag

Semaine 47(22/11/2010)

- Continuer à résoudre le problème de masque et de mots de passe

  → Problème rencontrés : erreur sur RS232 et MFRC

- Erreur de communication car on a inversé l'antenne
- Commander les cartes de test

Semaine 48 (29/11/2010)

- On a aperçu que les 4 premiers trames reçu sont l'UID
- Fin de la partie réception
- La partie émission

  → on a le problème du numéro de borne, erreur MFRC et RS 232

- Remarque : Si on ne peut pas transférer le programme, débrancher l'alimentation ou/et la liaison série et réessayer.

Semaine 49(6/12/2010)

- On a résolu le problème du code de sécurité, donc les deux programmes sont terminés

Semaine 50(13/12/2010)

- Réparation de carte :

  → Visualisation d'une tension trop faible en sortie du régulateur de tension car l'ATMEGA provoquait un court-circuit d'où la chute de tension. Donc changement de l'ATMEGA.

→ Changement de l'écran LCD détruit dût au court-circuit de l'ATMEGA.

→ Courant en entrée des cartes : 91mA. Donc, changement des fusibles sur les deux cartes (80mA par 100mA).

Semaine 01(3/1/2011)

- Changement de la puce RFID sur une des cartes, car détruite.
- Dernière vérification du bon fonctionnement des 2 cartes et des 2 programmes.
- Préparation d'un second boitier pour la deuxième carte.
- Rédaction du rapport écrit.

Semaine 02 (10/01/2011)

- Fin de rédaction du rapport écrit,
- Rendu du dossier.