Université François-Rabelais de Tours

Institut Universitaire de Technologie de Tours

Département Génie Électrique et Informatique Industrielle

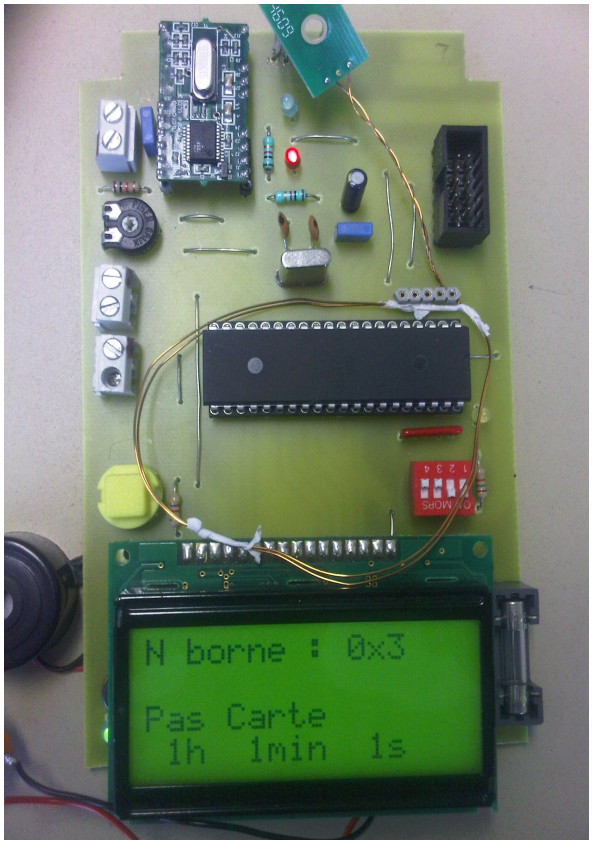
UNIVERSITE FRANCOIS-RABELAIS
TOURS

Institut Universitaire de Technologie

Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Projet Tutoré Étude et Réalisation 2e année

Mesure du temps de parcours pour une course à pied par badge RFID



Giraud Bastien
Bernard Rémy
Groupe Q1
Promotion : 2010-2012

Enseignants :
Lequeu Thierry
Gliksohn Charles

Université François-Rabelais de Tours

Institut Universitaire de Technologie de Tours

Département Génie Électrique et Informatique Industrielle

Projet Tutoré Étude et Réalisation 2e année

Mesure du temps de parcours pour une course à pied par badge RFID

Giraud Bastien
Bernard Rémy
Groupe Q1
Promotion : 2010-2012

Enseignants :
Lequeu Thierry
Gliksohn Charles

Sommaire

Introduction

Au cours du semestre 4 de notre formation en DUT GEII, nous devions réaliser un projet dans le cadre du cours d'Étude et Réalisation. Nous avons choisi en binôme un projet ayant pour thème la conception d'un système permettant la mesure d'un temps de parcours par RFID (Radio Frequency IDentification).

Notre projet a pour but de mesurer le temps de parcours d'un concurrent lors d'une course en utilisant la technologie RFID. Nous avons dû étudier la technologie RFID afin de comprendre son fonctionnement, et pouvoir facilement l'utiliser durant notre projet. Notre travail a consisté principalement à réaliser la programmation des cartes permettant la lecture et l'écriture sur les badges RFID, qui devront permettre de récupérer le temps mis par les concurrents pour réaliser un parcours lors d'événements sportifs tels que les « 10 km de Tours ». En effet les cartes ont été réalisées par des étudiants des années précédentes, ce qui nous a permis de nous concentrer sur la programmation.

Nous verrons donc dans ce rapport, en premier lieu, une présentation de notre projet ainsi que le cahier des charges. Puis nous étudierons la technologie RFID afin de comprendre comment fonctionne le système. Enfin nous présenterons le programme permettant le fonctionnement du projet et les améliorations que nous lui avons apporté.

1. Présentation du projet

1.1 Cahier des charges

Ce projet doit permettre de mesurer un temps de parcours réalisé par des participants lors d'une course comme celle des « 10 km de Tours » par exemple. C'est pourquoi le système à besoin de répondre à un cahier des charges précis afin d'avoir une utilisation optimale dans les conditions d'utilisation.

Détection du badge RFID

Le badge RFID devra être détecté par la borne d'arrivée, de départ et intermédiaire. La détection, dans le cadre d'une course devra se faire soit quand le coureur passe sous une arche (représentant l'antenne RFID), soit en s'arrêtant et en présentant le badge RFID à une borne qui enregistrera le passage de chaque participant.

Le badge

Le badge RFID devra contenir l'UID : l'identifiant qui va permettre de reconnaître chaque coureur. Ainsi qu'une balise temporelle pour se repérer dans le temps entre la borne d'arrivée et de départ. On marquera simplement l'heure de passage devant une borne intermédiaire, l'heure de départ ainsi que l'heure d'arrivée afin de calculer le temps total à la fin du parcours.

Le matériel

Pour réaliser ce projet, nous avons à disposition deux borniers similaires qui, grâce à un switch nous permettent de définir des bornes intermédiaires. Les borniers sont principalement composés de : l'antenne RFID, le décodeur/encodeur RFID, un régulateur de tension permettant l'alimentation de différents composants de la carte et un microcontrôleur ATmega8535. Ce dernier permettra de programmer la borne et ainsi définir ce qui sera écrit sur le badge RFID et le traitement à effectuer sur les données lues.

Synoptique

Au départ de la course, les concurrents doivent passer leur badge RFID devant la borne de départ, ce qui va permettre de récupérer l'heure de départ sur la carte. Lorsque les concurrents arriveront devant une borne intermédiaire, ils devront à nouveau passer le badge RFID devant la borne, et obtiendront ainsi le temps écoulé depuis le départ.

A l’arrivée, les concurrents vont passer une dernière fois leur badge, cette fois-ci devant la borne d’arrivée et ainsi obtenir le temps total qui leur a été nécessaire pour réaliser le parcours et l’heure d’arrivée.

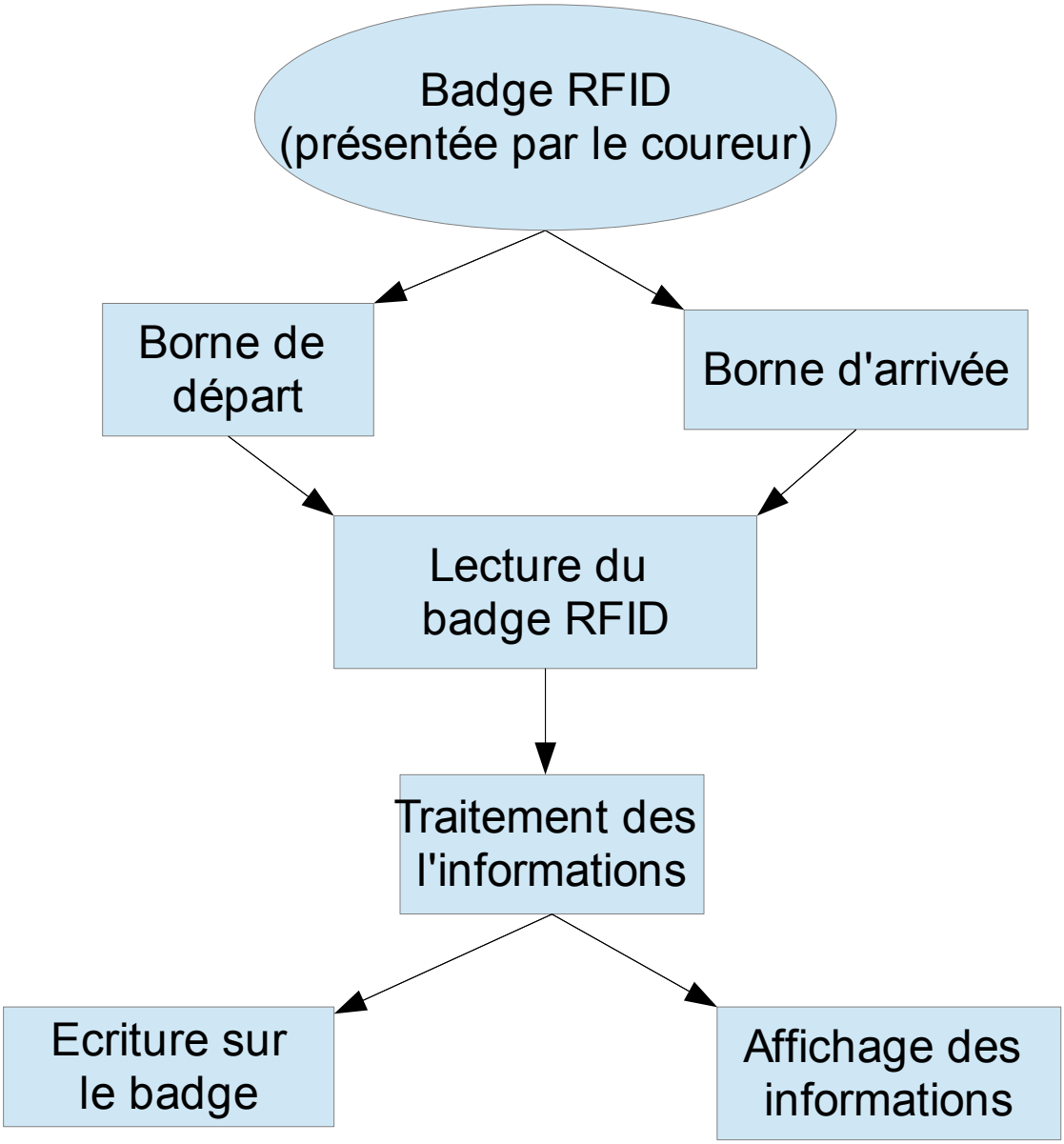
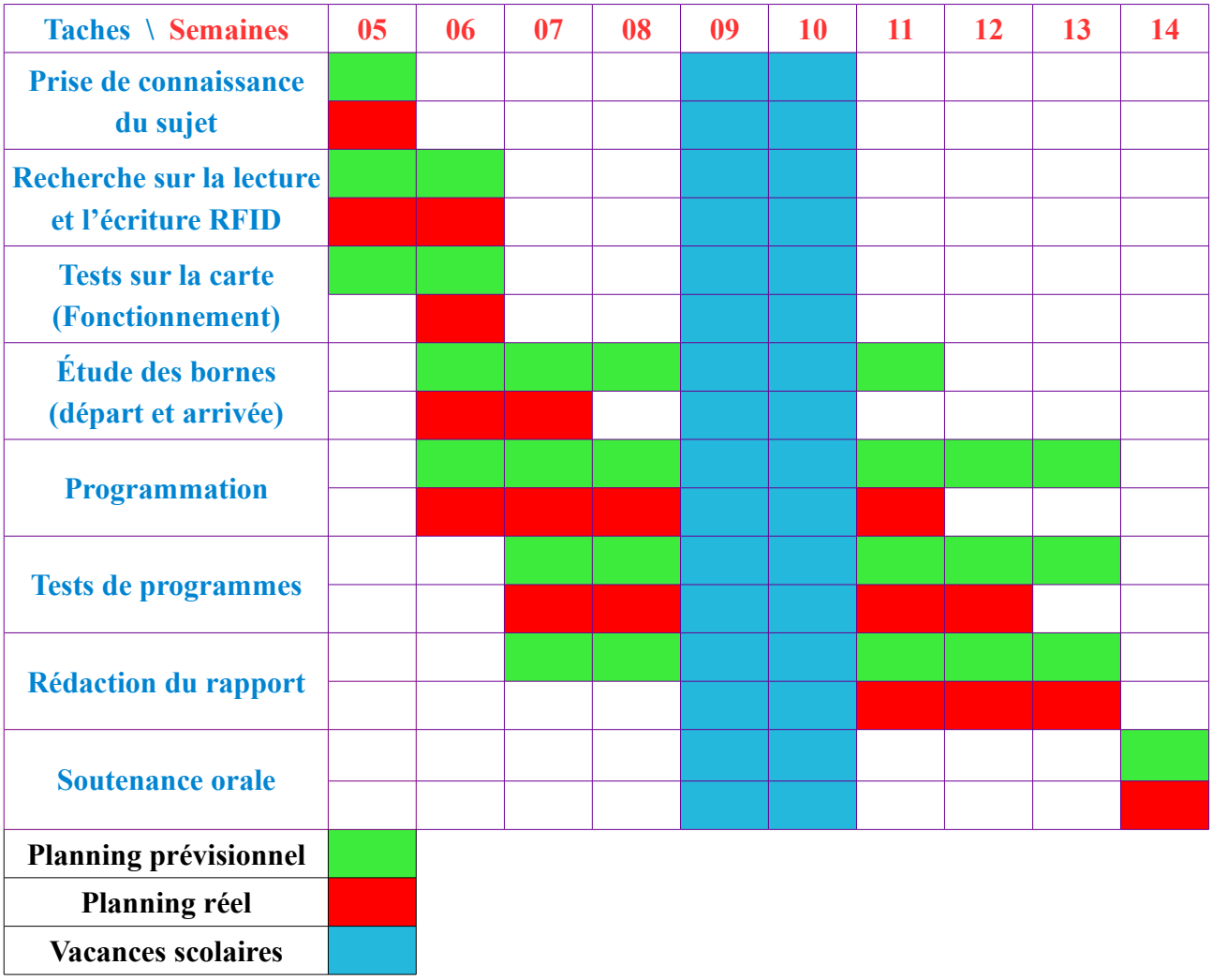


Illustration 1: Schéma synoptique

1.2 Planning prévisionnel

Pour réaliser ce projet durant le semestre 4, nous avons établi un planning prévisionnel afin de respecter le délai imposé. Les cartes permettant l'écriture et la lecture du badge RFID étant déjà réalisée, nous avons pu nous concentrer principalement sur la programmation et donc réaliser un programme permettant de répondre au cahier des charges.



2. Étude du projet

Dans cette partie, nous allons tout d'abord étudier la technologie RFID, afin de comprendre le fonctionnement du système. Nous étudierons ensuite la carte qui a été mise à notre disposition pour réaliser notre projet et qui va nous permettre d'écrire et lire sur les badges RFID.

2.1 La technologie RFID

La radio-identification, appelée communément RFID (Radio Frequency IDentification) est une technologie sans fil qui permet de stocker et de récupérer des données à distance. Elle est aujourd'hui utilisée dans de très nombreux domaines, notamment pour les passeports biométriques ou les cartes d'identifications comme les cartes d'abonnements de bus, etc.

La technologie RFID est composée de radio-étiquettes, comprenant une antenne et une puce électronique qui permettent de recevoir et de répondre à des requêtes radio émises depuis un émetteur-récepteur.

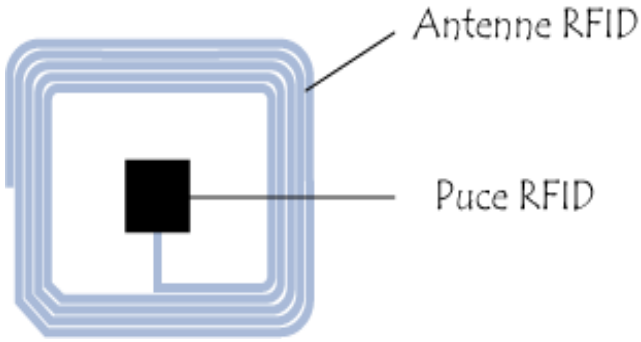


Illustration 2: Schéma RFID

Les radio-étiquettes sont des dispositifs passifs, ce qui veut dire qu'ils n'ont pas besoin de source d'énergie pour fonctionner, en dehors de celle fournie par les lecteurs lorsqu'ils envoient une requête. L'antenne RFID permet de communiquer avec des émetteurs de radiofréquences, sur des distances plutôt courtes, de l'ordre du mètre. La puce RFID contient les informations écrites sur le badge. Il est possible de lire les informations contenues dans la puce mais aussi d'en écrire de nouvelles.

Cette technologie est en compétition avec la technologie des codes barres, qui ne permettent, eux, uniquement une lecture et doit avoir une vision dégagée sur le code. La RFID, elle, n'a pas besoin de ligne de vue, possède une portée plus grande et permet également d'écrire sur le badge. La technologie RFID est donc supérieure sur de nombreux points à la technologie des codes barre, hormis le prix. La RFID coûte effectivement plus cher, car elle nécessite plus de matériel.

2.2 Étude de la carte

Comme nous l'avons précisé précédemment, les cartes jouant le rôle des borniers et permettant de réaliser la lecture et l'écriture des badges RFID nous ont été fournies, nous n'avons pas eu besoin de les réaliser.

Nous avons tout de même étudier la carte fournie, afin de comprendre le fonctionnement de notre système complet, et être capable de l'expliquer.

La carte se compose donc :

- d'un micro-contrôleur ATMega8535
- d'un régulateur de tension alimentant les différents composants
- d'un décodeur/encodeur RFID
- d'une antenne RFID
- d'un écran LCD

L'ATMEGA 8535 est un microcontrôleur très utilisé et disponible au magasin de l'IUT. Ce composant permet de gérer des entrées/sorties en le programmant préalablement en utilisant un logiciel comme CodeVision AVR.

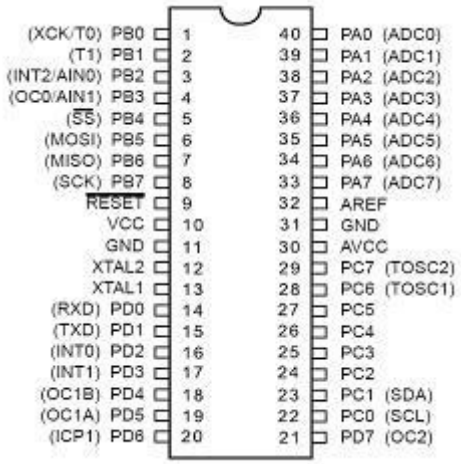


Illustration 3: Brochage ATMEGA

C'est lui qui va piloter la carte, il devra gérer l'affichage sur l'écran LCD, et faire le lien entre le module RFID et la carte.

Le LM2574N-5 est un régulateur de tension, capable de supporter une source de tension comprise entre 7 et 40 volts continus. C'est lui qui va produire la tension d'alimentation de 5V nécessaire au fonctionnement du montage.

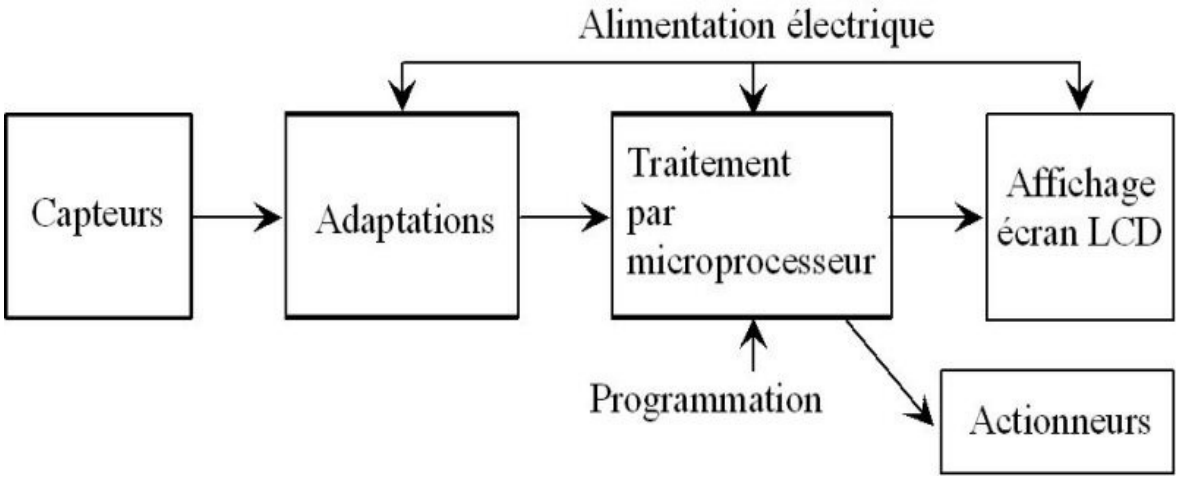


Illustration 4: Schéma fonctionnel

Le module RFID (décodeur/encodeur) va réaliser la liaison entre le badge et l'ATMEGA8535 en utilisant une antenne. Il va permettre de « traduire » les requêtes de l'ATMEGA8535 au badge et inversement.

L'afficheur dont nous disposons est un afficheur 16 caractères, 4 lignes et dispose d'un rétro-éclairage. Il va afficher les données transmises par l'ATMEGA8535 et informer le coureur. L'afficheur va permettre de visualiser : L'UID du badge, la présence d'une carte, ainsi que l'heure actuelle et le temps depuis le départ.

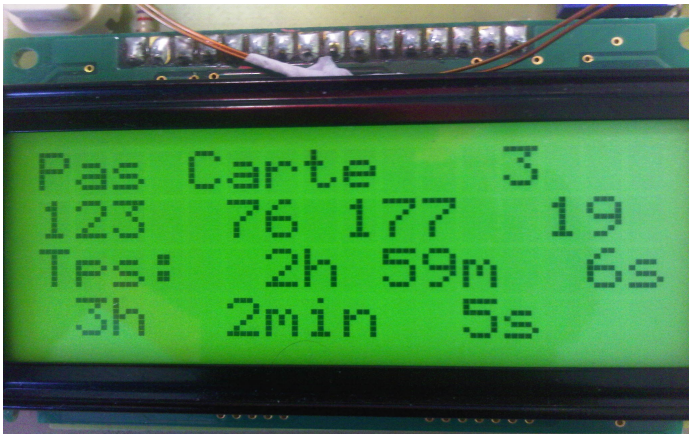


Illustration 5: Visualisation écran LCD

3. Réalisation

Dans cette partie, nous allons voir plus précisément la programmation des cartes de lecture et d'écriture, les modifications que nous avons apportées au programme, et les améliorations que nous avons réalisées tout au long de nos séances.

3.1 Programmation

La carte d'écriture

La carte d'écriture doit être capable de récupérer l'UID d'un badge (son identifiant) pour ensuite l'afficher sur l'écran LCD. Pour cela, nous utilisons la fonction "Recoit_UID", qui va envoyer la requête 0x55 au badge (requête qui correspond à une demande de l'identifiant de la carte), qui va ensuite nous répondre en renvoyant son UID. On utilise ensuite la fonction "USART_Transmit ()" qui est une fonction de la liaison série de l'ATmega8535 qui permet d'envoyer une trame sous un format déjà prédéfinis soit : 1 bit de start, 5 à 9 bits de données, un bit de parité et un ou deux bits de stop en fonction de la présence du bit de parité.

Fonction Recoit_UID :

```
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++;
   }
}
```

Après avoir récupéré l'UID pour l'afficher sur la borne, nous devions récupérer l'heure à laquelle le coureur a passé sa carte pour ensuite pouvoir calculer la différence entre les temps de départ et d'arrivée pour les afficher.

Pour modéliser une horloge, nous avons utilisé un des compteurs de l'ATmega8535 qui nous donne un top d'horloge pour le compteur toutes les 0,5 microsecondes. En lui demandant de compter jusqu'à 20 000, on arrive ainsi a avoir une interruption interne toutes les 10 millisecondes (0,5.10^-6 * 20 000 = 10.10^-3). On a donc décidé de compter jusqu'à 100 dans l'interruption interne pour en suite incrémenter une variable « seconde » qui incrémentera elle-même une variable « minute » puis « heure ».

Fonction d'interruption interne du compteur

```
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;
                                };
                        };
                };
        };
}
```

Une fois l'heure modélisée, on l'écrit sur le badge RFID grâce à la fonction "EcritureTemps". C'est grâce à cela que nous allons écrire l'heure de départ du coureur sur son badge RFID. Celle-ci sera ensuite lu par les bornes intermédiaires ainsi que par la borne d'arrivée.

Dans le programme principal, nous gérons la présence de la carte grâce à une fonction d'interruption. Lors de la présence de la carte, nous appelons chacune des fonctions ci-dessous.

Programme EcritureTemps

```
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();
}
```

La carte de lecture

La carte de lecture présente comme la carte d'écriture différentes fonctions, certaines sont identiques comme les fonctions "Recoit_UID" ainsi que la gestion de l'horloge avec sa fonction d'interruption de timer. La fonction VerifStatut est aussi présente dans la carte de lecture, mais pour des raisons d'équilibrage on vous la présentera ici.

La fonction VerifStatut va vérifier la présence d'un badge RFID devant l'antenne. Elle retourne 1 si la carte est présente et 0 dans le cas contraire. Cette fonction affichera aussi sur l'écran LCD "Pas de carte" quand il n'y en a pas et "Lecture" lors de la présence d'un badge RFID.

Fonction VerifStatut :

```
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;
}
```

Une fois que la présence du badge RIFD est détectée, on va lire l'heure de départ inscrit par la carte d'écriture grâce à la fonction "Recoit_Info". Dans cette fonction, nous allons récupérer un à un les heures, les minutes, les secondes et la variable Temps qui correspond aux centièmes. A ce même instant, nous allons récupérer l'heure actuelle pour pouvoir faire la comparaison entre l'heure de départ et l'heure réelle pour que le coureur connaisse le temps qu'il a mis jusqu'à maintenant si il est à une borne intermédiaire ou le temps total de sa course si il est arrivé à la borne d'arrivée. Pour faire cela, on met l'heure de départ en seconde ainsi que l'heure d'arrivée, on les soustrait et on place le résultat sous la forme : heures :minutes :secondes.

Fonction Recoit_Info :

```
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
  if(Information == 0x86)             nécessaire du Acknowledge
  {
      while(i<=15)                    //16 bits à récupérer
      {
          car = USART_Receive();
          Temps[i] = car;
          i++;
      }
      S_arr = Heure*3600 + Minute*60 + Seconde;
      S_mis = S_arr - (((int)Temps[7])*3600 + ((int)Temps[6])*60 + (int)Temps[5]);
      Heure_mis = S_mis /3600;
      Minute_mis = (S_mis%3600) /60;
      Seconde_mis = (S_mis%3600) %60;
  }
}
```

3.2 Améliorations

Nous avons amélioré différentes choses sur le programme et aussi rajouté un buzzer lors du passage à la borne de départ.

Utilisation du Buzzer

L'implantation du buzzer n'a pas été très difficile, car nous avions directement des pattes de l'ATmega8535 qui étaient libres. Nous avons ainsi choisi l'une d'entre elles pour le +5 V du buzzer et relié le deuxième fil de celui-ci au plan de masse. Du côté de la programmation, nous avons directement changé la valeur de la sortie sur laquelle le buzzer est branché. Avec un timer pour que celui-ci dure assez longtemps pour qu'il soit audible.

Modification de l'affichage

L'affichage de base ne montrait pas le temps mis par le coureur mais l'heure à laquelle il était parti. Nous avons préféré faire directement le calcul pour afficher le temps mis par le coureur. Il parait évident qu'une personne préférera voir le temps qu'il a mis pour réaliser le parcours plutôt que l'heure à laquelle il l'a commencé.

```
Programme d'affichage de l'heure actuelle

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

Dans ce code, nous mettons dans une même variable l'heure suivie de "h" puis les minutes suivies de "min" et enfin les secondes suivies de "s". Ensuite on se place au début de la 4e ligne pour afficher la variable.

Affichage borne d'écriture :

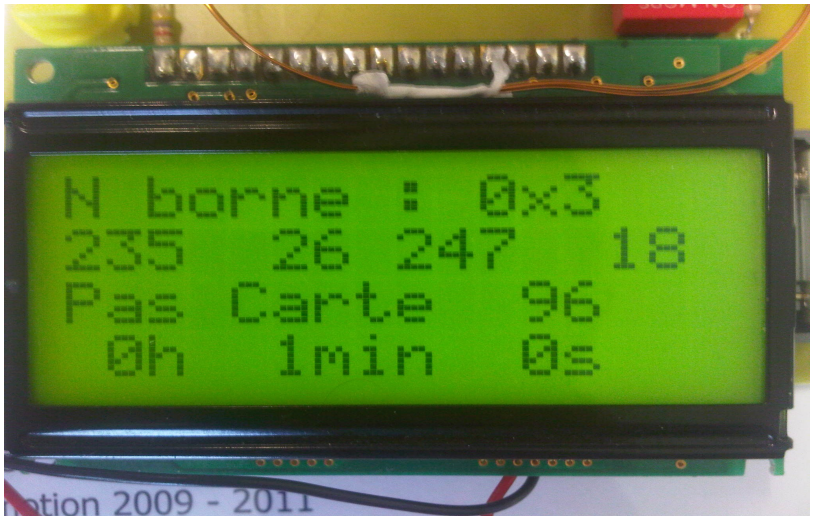


Illustration 6: Photo écran LCD de la borne d'écriture

La 1ère ligne correspond au numéro de borne. Vient ensuite l'identifiant du dernier badge détecté, puis la présence ou l'absence d'un badge sur la troisième ligne. On a ensuite le temps écoulé depuis le départ sur la dernière ligne.

Affichage borne de lecture :

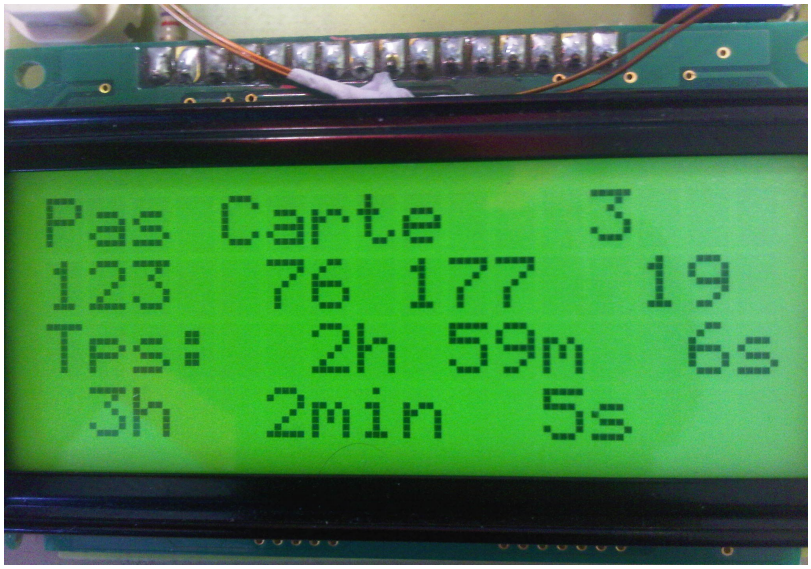


Illustration 7: Photo écran LCD borne de lecture

Comme pour la borne d'écriture, on peut observer l'identifiant et la présence de la carte sur les deux premières lignes de l'écran LCD. On observe sur la troisième le temps réalisé par le coureur : ici 2h 59min et 6 secondes.

Conclusion

La plus grande partie de ce sujet a consisté à la création d'un programme permettant d'utiliser des bornes RFID afin de mesurer des temps de parcours. En effet la carte étant déjà réalisée, nous avons pu nous concentrer sur l'élaboration du programme selon le cahier des charges et même apporter des améliorations.

Nous pouvons dire que ce projet aura été bénéfique pour la compréhension du mode de transmission en RFID. En effet, la RFID est de nos jours beaucoup utilisée, par exemple pour des badges faisant office de clé de porte de chambre d'hôtel, ou comme ici un badge appartenant à un coureur pour qu'il soit reconnu et que ce même badge puisse transporter des informations.

Nous avons repris un projet existant pour lui apporter des nouveautés, ce qui est le cas dans la plupart des entreprises. On a donc pu concilier nos connaissances sur ce sujet, les approfondir et les appliquer sur un projet concret. Une fois de plus nous avons été projeté dans l'ambiance de l'entreprise en totale autonomie, ce qui est très bénéfique et nous prépare notamment pour notre futur stage en entreprise de fin d'étude.

Résumé

Durant ce projet, nous avons pu découvrir la technologie RFID plus en détail, nous l'avions effectivement déjà rencontrée durant notre formation.

Nous avons continué le travail d'étudiants des années précédentes, le but de notre projet était de réaliser un système permettant de mesurer le temps de parcours d'un coureur lors de course grâce à la technologie RFID. Nous avons donc continué le travail déjà réalisé par d'anciens étudiants, la carte étant déjà créée nous nous sommes concentrés sur la réalisation du programme.

Nous avons donc réussi à mettre en œuvre le système, et avons même réussi à apporter quelques améliorations à celui-ci, en effet nous avons optimisé l'affichage de l'écran LCD afin de récupérer le temps effectuer par chaque concurrent, et avons ajouté un buzzer sur les bornes RFID permettant aux coureurs de savoir si le passage du badge a bien été pris en compte.

Ce projet nous a permis de découvrir plus précisément le fonctionnement de la RFID, et d'approfondir nos connaissances sur le sujet. Le système est maintenant fonctionnel et est prêt à être utilisé en conditions réelles.

Index des illustrations

Bibliographie

[1]A. BINTI ABDULLAH, C. SAPIENS, *Mesures du temps de parcours par RFID pour une course à pieds ou en vélo*, projet IUT GEII Tours, 2A-Q2, janvier 2011

[2]T. LEQUEU, *La documentation de Thierry LEQUEU,* http://www.thierry-lequeu.fr/

Annexes

Annexe 1 : Programme complet de l'ATMEGA8535

Module d'écriture :

```
#include <mega8535.h>

#asm
        .equ __lcd_port=0x15
#endasm

#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

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 )
{
while ( !( UCSRA & (0x20)) ); // Test de UDRE bit 5
UDR = data;
}

unsigned char USART_Receive( void )
{
while ( !(UCSRA & 0x80) ); // Test de RXC bit7
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)
{
// Input/Output Ports initialization
// Port A initialization
PORTA=0x00;
DDRA=0xF0;

// Port B initialization
PORTB=0x00;
DDRB=0x00;

// Port C initialization
PORTC=0x00;
DDRC=0xFF;

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

// Timer/Counter 0 initializatio_n
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
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);

           sprintf(tampon,"%x",Information);
              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;
    }
}
}
```

Module de lecture :

```
#include <mega8535.h>

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

#include <stdio.h>

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

unsigned char trame, Statut, Information;
unsigned char Temps[20];
unsigned char tampon[20], Identifiant[10], tampon2[20];
int Tps, Seconde, Minute, Heure, Heure_mis, Minute_mis, Seconde_mis, S_mis,S_arr;

interrupt [TIM1_COMPA] void timer1_compa_isr(void);
void Horloge(void);
interrupt [EXT_INT1] void ext_int1_isr(void);

void USART_Transmit( unsigned char data )
{
        while ( !( UCSRA & (0x20)) ); // Test de UDRE bit 5
        UDR = data;
}
```

```
unsigned char USART_Receive( void )
{
        while ( !(UCSRA & 0x80) ); // Test de RXC bit7
        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++;
        }
        S_arr = Heure*3600 + Minute*60 + Seconde;
```

```
        S_mis = S_arr - (((int)Temps[7])*3600 + ((int)Temps[6])*60 + (int)Temps[5]);

        Heure_mis = S_mis /3600;
        Minute_mis = (S_mis%3600) /60;
        Seconde_mis = (S_mis%3600) %60;

    }
}

void main(void)
{
// Port A initialization
PORTA=0x00;
DDRA=0x00;

// Port B initialization
PORTB=0x00;
DDRB=0x00;

// Port C initialization
DDRC=0x00;

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

// Timer/Counter 0 initialization
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=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
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
GICR|=0x80;
MCUCR=0x0A;
MCUCSR=0x00;
GIFR=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;

TIMSK=0x10;
```

```
	// 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);

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

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

		  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);

			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, "Tps: %2dh %2dm %2ds", Heure_mis, Minute_mis, Seconde_mis);
    lcd_gotoxy(0, 2);
    lcd_puts(tampon);
}
}




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

void Horloge(void)
{
    sprintf(tampon2,"%2dh %2dmin %2ds", Heure, Minute, Seconde);
    lcd_gotoxy(0,3);
```

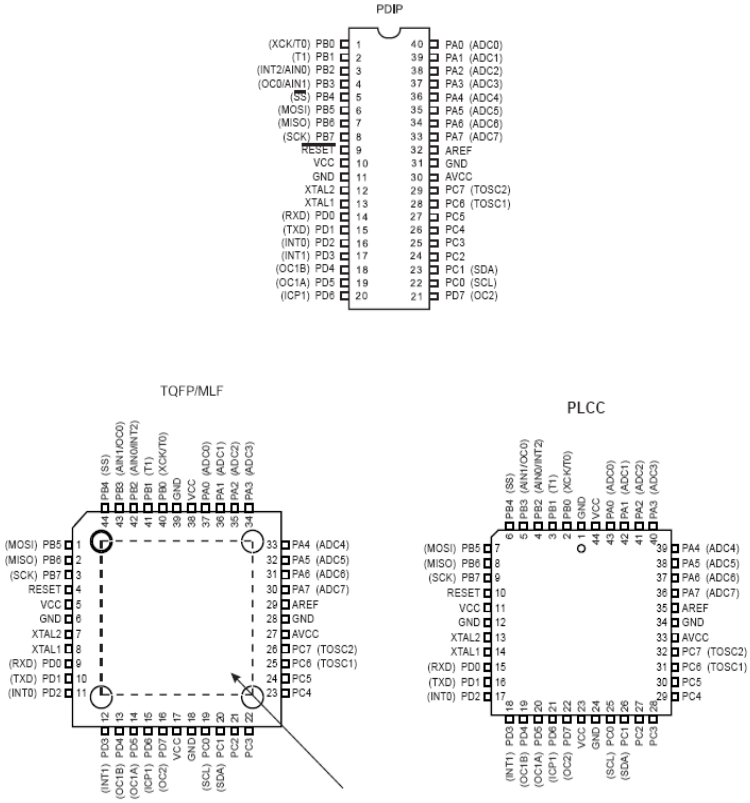
```
    lcd_puts(tampon2);
}

//Remise à zéro synchronisée
interrupt [EXT_INT1] void ext_int1_isr(void)
{
    Tps=0;
    Seconde=0;
    Minute=0;
    Heure=0;
}
```

Annexe 2 : Brochage de l'ATMEGA 8535

Pin Configurations

Figure 1. Pinout ATmega8535



NOTE: MLF Bottom pad should be soldered to ground.

Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Pin Descriptions

V_{CC}	Digital supply voltage.
GND	Ground.
Port A (PA7..PA0)	Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.
Port B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega8535 as listed on page 58.
Port C (PC7..PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.
Port D (PD7..PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega8535 as listed on page 62.
$\overline{\text{RESET}}$	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 35. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.