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

Rapport de projet tutoré
2e Année

Tableau d'affichage des scores
Escrime

Rémi BENOIT
Alban Lobozzo
2008/2010
Groupe : K4B

Enseignants
Thierry Lequeu
Christine MERY

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

Tableau d'affichage des scores Escrime

Rémi BENOIT
Alban Lobozzo
2008/2010
Groupe : K4B

Enseignants
Thierry Lequeu
Christine MERY

Sommaire

Introduction

Durant les séance d'Etude et réalisation nous avons mis en pratique nos connaissances afin de réaliser un appareil électrique de signalisation pour l'escrime. Cette appareil a donc pour but d'aider l'arbitre durant un assaut. L'appareil doit donc être capable d'afficher le score, le temps et les touches. De plus il doit être gérer par une télécommande qui sera en possession de l'arbitre. Celle ci devra incrémenter le tableau des score, et contrôler le temps.

1. Cahier des charges

1.1. Explication générale

Le but de ce projet est de réaliser un appareil électrique pour l'escrime. Le but est simple, pouvoir permettre à l'arbitre de comptabiliser le score, gérer le temps et les cartons, en fonction des trois différentes armes. Pour cela, l'appareil électrique, peut être divisé en trois parties différentes. La première qui gère les touches, et allume des leds, quand l'un des tireurs est touché. Il faut pourvoir différencier les 3 armes. (épée, fleuret, sabre). Ensuite, pourvoir comptabiliser et afficher le score de chaque tireur ou équipe, ainsi que les sanctions, pris envers lui. Le score sera présenté par l'intermédiaire de deux afficheurs LCD, et les cartons par des leds. Et enfin, l'arbitre pourra commander l'appareil à distance par l'intermédiaire d'une télécommande infrarouge. Il faudra donc créer une télécommande qui génèreras des trames et les enverra par infrarouge a un récepteur situer sur l'appareil.

1.2. Principales contraintes

- Affichage sur écran LCD
- Affichage score (Afficheurs 7 segments multiplexages)
- Liaison série (télécommande)
- Interface utilisateur simple
- Utilisation de l'ATméga 8535

2. Recherches bibliographiques

Notre système nous à mener à faire des recherches sur internet. Tout d'abord ils nous à fallut faire des recherches sur internet pour voir si le système étudié existé déjà. Nous avons également fait des recherches sur la détection des touches avec les différentes armes. Les dernières recherches ont été faite sur le micro-contrôleur Atméga 8535, pour connaître sa structure ainsi que sa programmation.

3. Analyse du projet

Tout notre projet est structuré autour d'un micro-contrôleur Atméga 8535. Ce composant est le cœur de la carte. En entré de celui-ci nous retrouverons la détection des touches de chaque joueur. Le micro-contrôleur aura pour rôle de traiter les informations reçus et de piloter les sorties en conséquence. Il nous permettra de lire le score sur des afficheurs 7 segments, de visualiser sur des LED si un des joueurs touche, si un joueur a un carton jaune ou rouge ou si une touche est non valable suivant le type d'arme utilisé.

Un écran LCD sera également piloté il permettra d' indiquer à l'arbitre le temps de jeu. Une télécommande sera également utilisé par l'arbitre se qui lui permettra de mettre les différents cartons et différentes pénalités. La télécommande sera piloté par une liaison série.

La signalisation des touches:

Par un manque de temps et du aux difficultés rencontrer sur d'autres partie, nous avons put faire que de la théorie sur cette partie.

L'appareil servira donc à arbitrer un assaut d'escrime qui oppose deux tireurs. Pour signaler une touche, une lampe s'allume sur l'appareil, qui indique le tireur qui a touché. Pour comprendre le phénomène il faut se représenter la piste (donc les deux tireur avec leur armes et le sol) comme un circuit fermer. L'arme du tireur fait office d'interrupteur. Lorsqu'il touche, l'interrupteur se ferme et le courant passe.

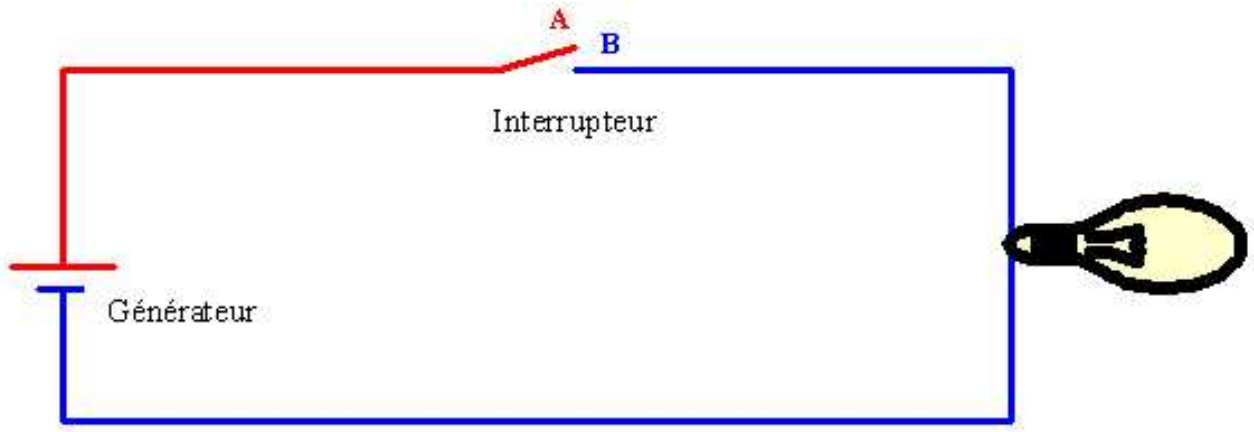


Illustration 1: Schéma de la piste

Texte 1: Image prise sur http://societe-escrime-lyon.over-blog.com

Cependant il y a différent type d'arme. Et chaque arme a un circuit spécifique. Le plus simple est celui de l'épée. La surface valable à l'épée est tout le corps.

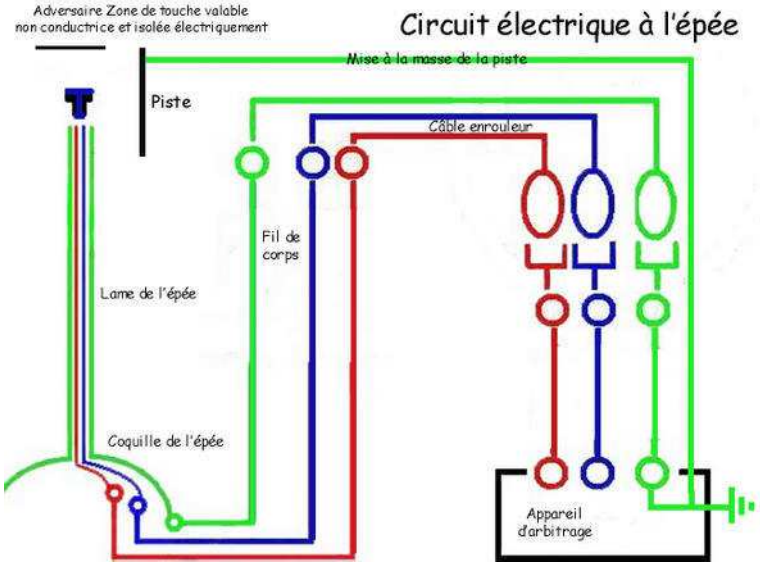


Illustration 2: shéma électrique de l'escrime

Texte 2: image prise sur: http://societe-escrime-lyon.over-blog.com/categorie-10477783.html

L'épée est relié a la masse par la terre. La piste est elle aussi relié à la masse. Chaque armes est relié à l'appareil électrique. Qui est le générateur. Ce dernier envoie un courant. Et tant que le tireur ne touche pas, le courant ne passe pas. Dès qu'il touche, la pointe ferme le circuit, et le courant passe, allumant ainsi la lampe.

Dans ce circuit, seul un tireur suffit, on peut dire que chaque tireur est autonome. Contrairement au fleuret ou au sabre, car la surface valable est limité. Cette dernière est établie par une veste électrique. Donc quand un tireur touche son adversaire, sur la surface valable. La boucle se faire

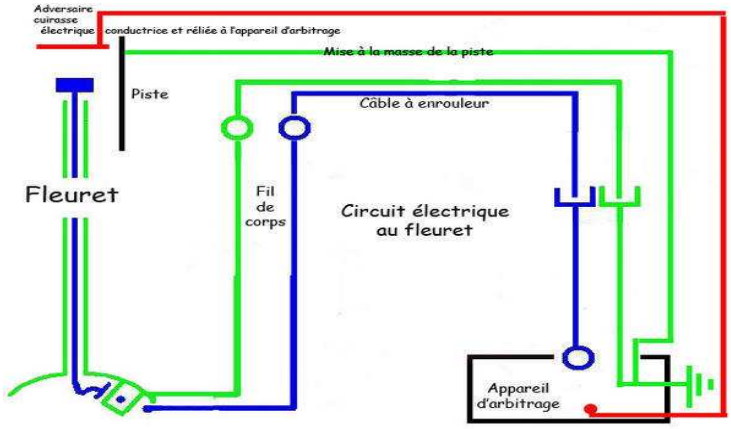


Illustration 3: Schéma électrique du fleuret

Pour notre projet, nous pensions détecter une différence de potentiel, qui nous aurais ensuite permis d'activer les lampes grâce a l'ATMEGA..

4. Schémas fonctionnels

4.1. Schéma de niveau 1

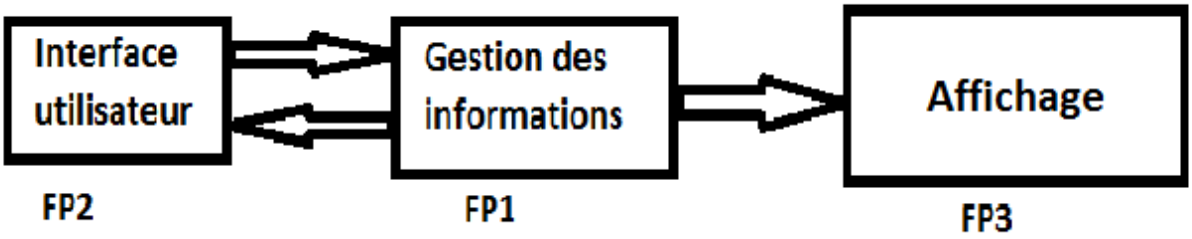


Illustration 4: Schéma fonctionnel de 1er niveau

Le système se divise en 3 fonctions principales. FP1 est au cœur de ce système c'est elle qui gère toute les informations.

4.2. Schéma de niveau 2

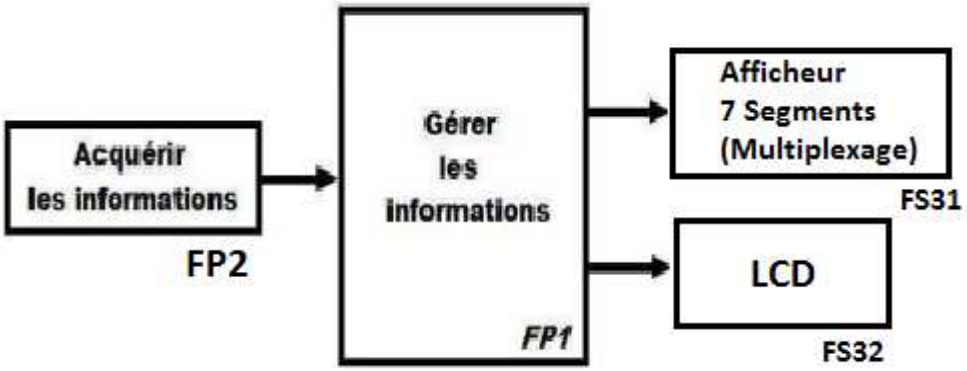


Illustration 5: Schéma fonctionnel de 2nd niveau

4.3. Analyse fonctionnelle

FP1 Gérer les informations : Un composant programmable récupère les informations, les interprètent et envoi en sortie des commandes, pour FP3 qui va afficher les différentes informations.

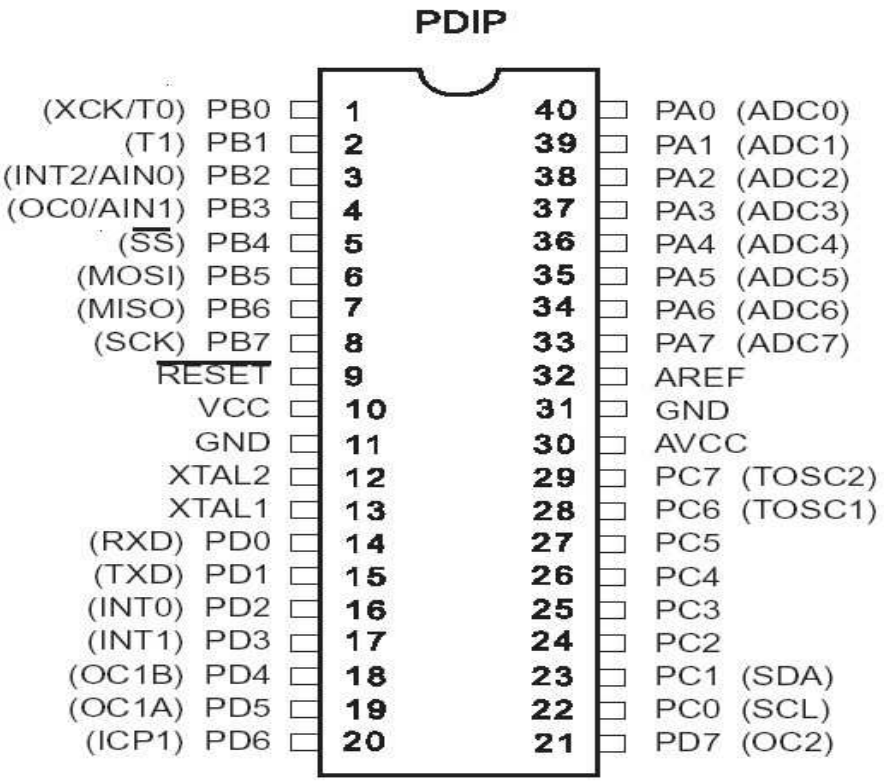


Illustration 6: Brochage Atmega 8535

Le micro-contrôleur offre différentes possibilités de programmation. Les broches peuvent être configuré soit en entrée ou en sortie avec ou sans résistance de tirage. Il est cadencé par un quartz. C'est sortie sont active sont niveau 0.

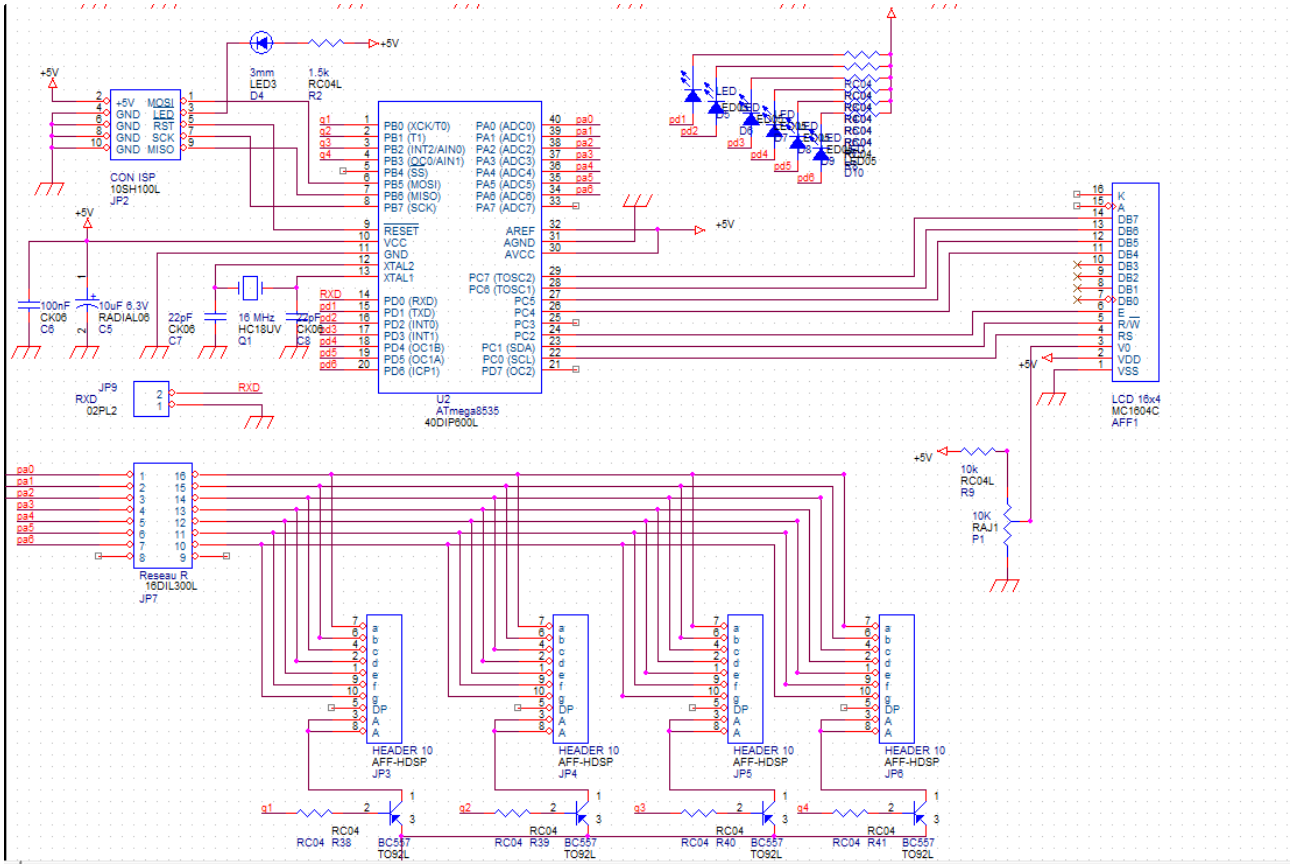
FP2 Acquérir les informations : Détecter les touches des deux joueurs, recevoir les informations de la télécommande.

FS31 Afficheurs 7 segments : Pour afficher les scores nous utilisons 4 afficheurs 7 segments. Pour gagner de la place en réduisant au maximum le nombre de fil, on utilise la technique du multiplexage. Cette technique consiste à envoyer les informations et allumer chaque afficheur successivement à une fréquence suffisamment élevé pour que l'utilisateur est l'impression que les afficheurs soient toujours allumé.

FS32 LCD : Affichage du temps de jeu.

5. L'Atémega

L'atéméga est le cœur de la carte. Il gère les afficheurs, les leds, et la connexion RS232.



Les 6 leds sont reliés au port D. L'afficheur LCD est sur le port C. Le port A gère les transistor qui permettrons de faire fonctionner les afficheurs 7 segment. Et le PortB permet d'indiquer la valeur afficher sur les 7 segments.

La partie qui nous a posé le plus de souci, fut le multiplexage. Car, malgré de nombreuse recherches, nous avons rein trouvé qui puisse nous permettre de le réaliser avec L'atémega.

Chaque afficheurs, içi a cathode commune, ne sera actif qu'un court instant. Ils seront tous actifs a tour de rôle. En affichant la valeur souhaité. Ils seront donc allumé seulement quand la base de leur transistor sera activer. Pour simplifier la chose, le transistor sert d'interrupteur. Dès qu'un courant arrive sur la base, l'afficheur est alimenté.

Cette manipulation doit s'effectuer très vite. Affin que l'œil humain puisse voir le score en continu.

Pour la programmation, dès que l'un des tireur touche, et que l'arbitre, envoie l'incrémentation d'un score, une variable sera augmenter de 1. EN fonction de la valeur de la valeur de cette variable, certains segment seront seront actifs et d'autre non. Pour cela on a effectuer par l'intermédiaire de switch, des bus. (voir annexe, le programme de réception.)

Le routage de la partie de réception, nous a demander beaucoup de temps du a sa complexité, notamment au niveau du multiplexage.

6. Programmation de L'Atéméga

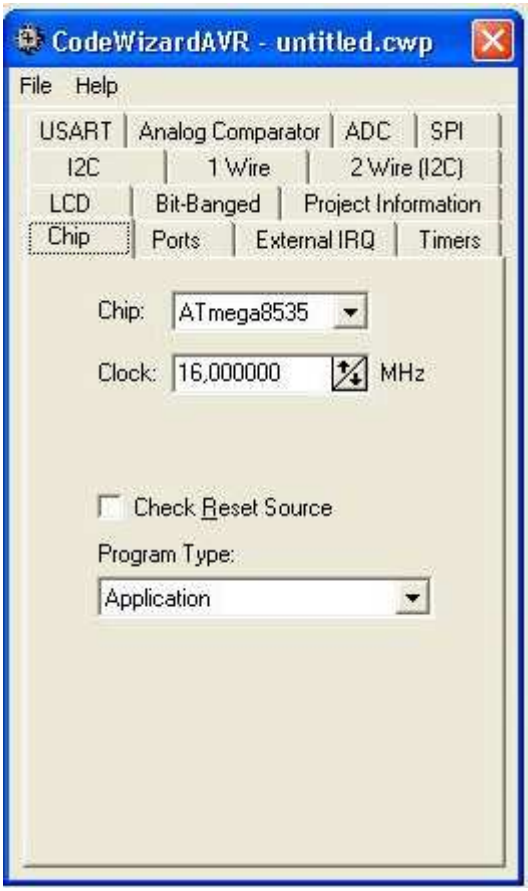


Illustration 7: Capture d'écran de la configuration de l'Atmega

L'appareil doit donc être commander par une télécommande utiliser par l'arbitre. Cette télécommande contient elle aussi un Atémega. Son rôle est simple, envoyer des trames chaque pression sur un bouton.

Pour émettre des trame avec un ATMEGA on utilise la fonction USART_Transmit

```
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;
}
```

Cette fonction nous fut donné par monsieur Lequeu. Elle attend paramètre de type caractère mais ne retourne rien. Nous utilisons cette fonction pour transmettre des caractère.

Pour tester la onction de transmission, nous avons mis en place un programme simple. Celui envoie en Permanence le caractère « a ». Ce caractère est ensuite envoyer à l'afficheur LCD par une liaison RS232 par l'intermédiaire de la patte 15 (PIND,0 TXD)

```
while (1)
    {
    if ( BJ1==0)
    {
     USART_Transmit('B');
    }

    if ( TJ1==0)
    {
     USART_Transmit('T');
    }

    if ( JJ1==0)
    {
     USART_Transmit('J');
    }

    if ( BJ2==0)
    {
     USART_Transmit('N');
    }

    if ( TJ2==0)
    {
     USART_Transmit('G');
    }

    if ( JJ2==0)
    {
     USART_Transmit('C');
```

```
    }

    if ( MJ1==0)
    {
     USART_Transmit('M');
    }

    if ( MJ2==0)
    {
     USART_Transmit('D');
    }

    }
}
```

On associe a chaque entrée entrée du micro-controleur un bouton poussoir en utilisant #define. Cette définition doit précéder le programme.
Cela donne ceci

```
#define BJ1 PINA.0
#define TJ1 PINA.1
#define JJ1 PINA.2
#define BJ2 PINA.3
#define TJ2 PINA.4
#define JJ2 PINA.5
#define MJ1 PINA.6
#define MJ2 PINA.7
```

Ainsi, cela facilite la compréhension du programme.
Ici B signifie bleu, qui correspond à led bleu qui se trouve sur la carte principal. T, signifie Touche , et J signifie Jaune, ce qui correspond à la led jaune. J1 ou J2 correspond respectivement un joueur 1 et joueur 2. Tandis que M signifie Moins.

Dès que l'on actionne un bouton (qui est détecté par un état logique 0), on va émettre une trame grâce à la fonction USART_Transmit. Chaque bouton Poussoir permet d'envoyer un caractère précis.
Dans notre projet on transmet le caractère B, par exemple, quand on actionne le bouton poussoir BJ1. (qui est relié à l'entrée A0 de l'ATMEGA).

Une fois que les trames sont émises, on doit être dans la capacité de les lire, et d'appliquer certains action en fonction de la trame reçu. La fonction nous permettant de recevoir les trame est USART_Receive

```
}
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;
}
```

Cette fonction retourne une chaine de caractère.

En fonction du caractère reçu, on effectue une action précise. On effectue de nombreuses structures de condition « if » pour spécifier l'action de chaque caractère. Par exemple, quand on reçoit le caractère B, on doit allumer une led de couleur jaune, qui représente le non-valable. Pour cela on active la sortie de l'une des patte dur port D.

7. Nomenclature

7.1. Carte Réception

N°	Quantité	Référence	Désignation	Empreinte
1	1	AFF1	LCD 16x4	MC1604C
2	1	C1	100uF 63V	RADIAL08
3	1	C2	330uF 6.3V	RADIAL06L
4	2	C3,C5	10uF 6.3V	RADIAL06
5	2	C6,C4	100nF	CK06
6	1	C7	22pF	CK06
7	1	C8	22pF	CK06
8	1	D1	1N4007	DO41
9	1	D2	11DQ04	DO41
10	1	D3	2 mA	LED3
11	1	D4	3mm	LED3
12	1	JP1	ALIM	WEID2
13	1	JP2	CON ISP	10SH100L
14	1	JP3	ANALOG	20SH100
15	1	JP4	n.c.	02PL1
16	1	JP5	HEADER 10X2	20SH100L
17	1	JP6	HEADER 10	10PL1
18	1	JP7	n.c.	05PL1
19	1	JP8	Port D	08PL1
20	1	JP9	BP1	02PL1
21	1	JP10	BP2	02PL1
22	1	JP11	BNC	02PL1
23	1	JP12	BNC	BNC1
24	1	L1	10uH	RADIAL06L
25	1	L2	47uH	RADIAL06L
26	1	P1	10K	RAJ1
27	1	Q1	16 MHz	HC18UV

28	2	R1,R5	1.5k	RC04L	
29	1	R2	1k	RC04L	
30	2	R3,R4	4.7k	RC04L	
31	2	SW2,SW1		TOUCHE REDROND	
32	1	U1	LM2575-5	TO220-5B	
33	1	U2	ATmega8535	40DIP600	
34	4	VIS1,VIS2,VIS3,VIS4		VISSERIE	M3L

7.2. Carte Transmission

N°	Quantité	Référence	Désignation	Empreinte
1	1	C1	100nF	CK06
2	1	D1	3mm 2mA	LED3
3	1	JP1	CON ANALOG	20SH100L
4	1	R1	4.7k	09PL1
5	1	R2	1.5k	RC04L
6	8	SW1,SW2,SW3,SW4,SW5,SW6, SW7,SW8		TOUCHE REDROND
7	4	VIS1,VIS2,VIS3,VIS4		VISSERIE M3L

Conclusion

Lors de ce projet nous avons mis en application les connaissance que nous avons acquise durant des années précédentes. Notre but était donc de créer un appareil de signalisation pour l'escrime. Il devait afficher le score, signaler les touches et compter le temps. Malheureusement, le sujet c'est révéler bien plus complexe qu'on ne l'imaginais, et le temps nous a fait défaut. Nous avons néanmoins réaliser des prototypes quasi fonctionnel. On a rencontré des problèmes dans la programmation, notamment sur la communication de la liaison série.

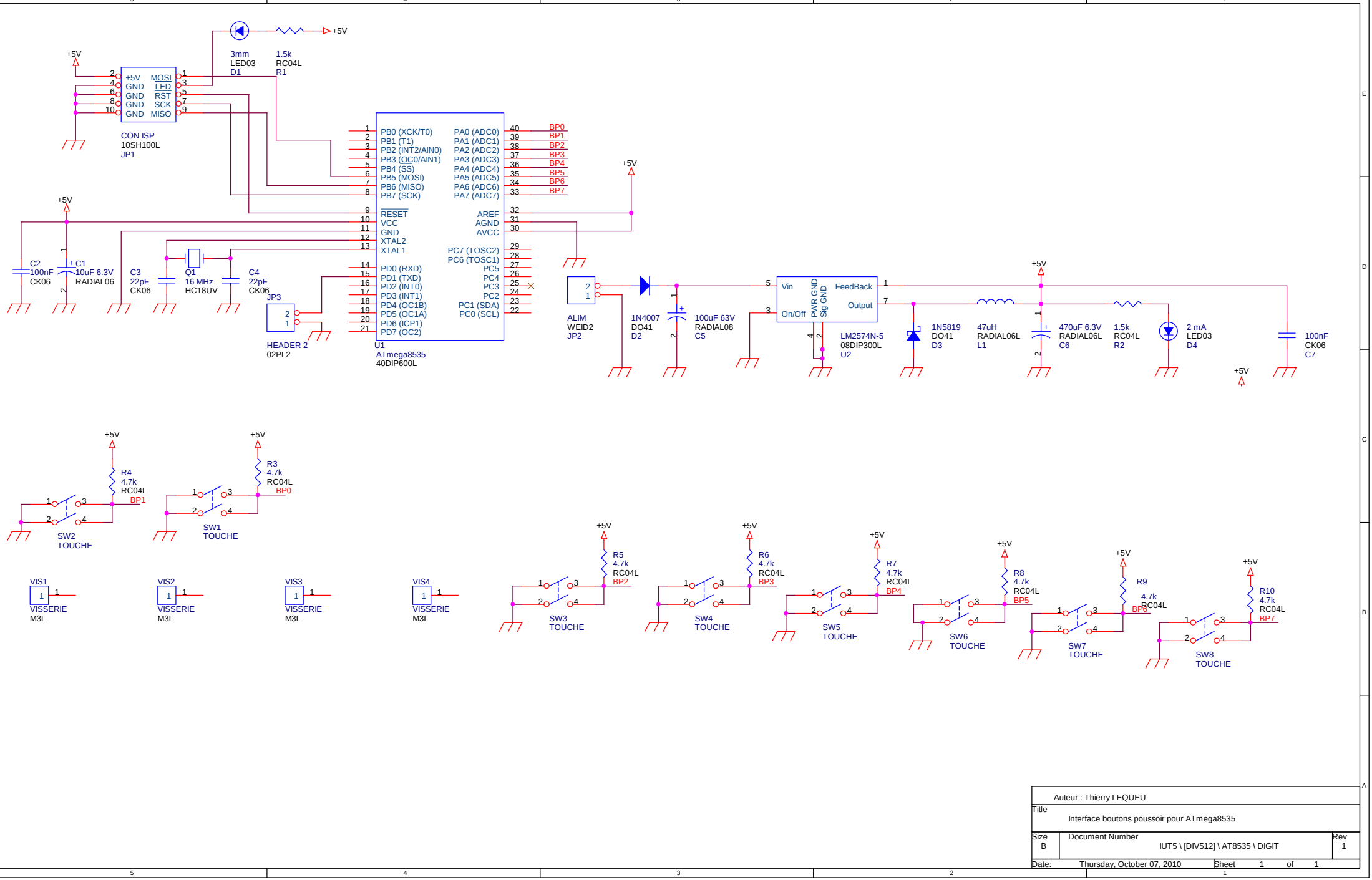
Index des illustrations

Bibliographie

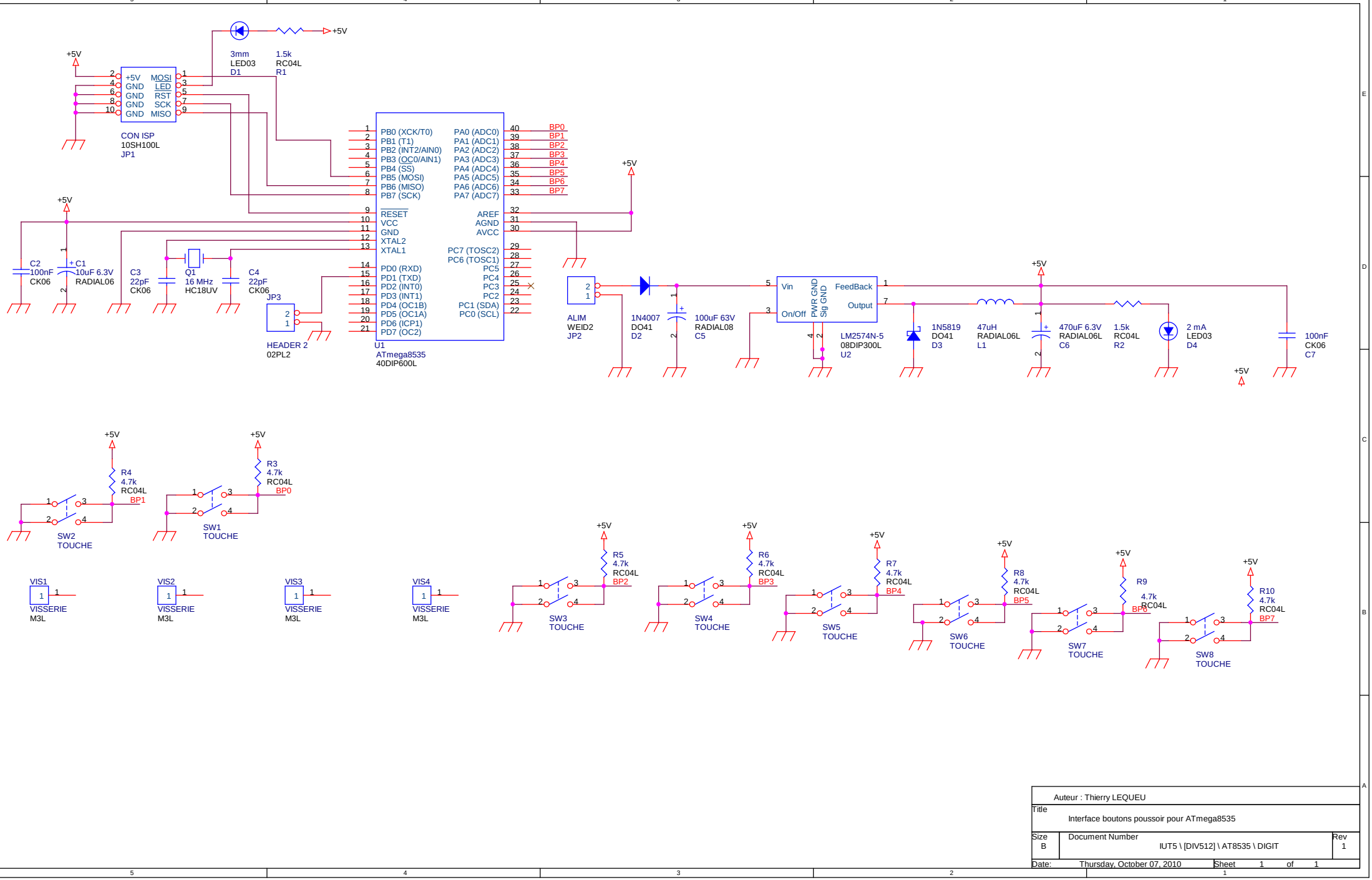
http://societe-escrime-lyon.over-blog.com

http://www.datasheetcatalog.net/

Annexes



Auteur : Thierry LEQUEU		
Title: Interface boutons poussoir pour ATmega8535		
Size: B	Document Number: IUT5 \ [DIV512] \ AT8535 \ DIGIT	Rev: 1
Date: Thursday, October 07, 2010	Sheet 1 of 1	



Auteur : Thierry LEQUEU		
Title: Interface boutons poussoir pour ATmega8535		
Size: B	Document Number: IUT5 \ [DIV512] \ AT8535 \ DIGIT	Rev: 1
Date: Thursday, October 07, 2010	Sheet 1 of 1	

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 20/10/2010
Author  : F4CG
Company : F4CG
Comments:


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

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

#define LED_rouge PIND.1
#define LED_bleuj1 PIND.2
#define LED_jaunej1 PIND.3
#define LED_verte PIND.4
#define LED_bleuj2 PIND.5
#define LED_jaunej2 PIND.6
#define G1 PINB.0
#define G2 PINB.1
#define G3 PINB.2
#define G4 PINB.3

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

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here

}
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;
}
// Declare your global variables here

void main(void)
{
// Declare your local variables here
 unsigned char Tampon;
 int score1=0;
 int score2=0;


// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
```

```
// State7=T State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
PORTA=0x7F;
DDRA=0x7F;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x0F;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=T State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xF7;

// Port D initialization
// Func7=In Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=T State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0x7F;

// 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: 15,625 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x3D;
OCR1AL=0x09;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
```

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

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x10;
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;

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

while (1)
      {
      G1=1;
      G2=1;
      G3=1;
      G4=1;
      // Place your code here
      PORTD=0xFF;
      Tampon=USART_Receive();

      if(Tampon=='B')
       {
       LED_jaunej1=0;
       delay_ms(3000);
       }

       if(Tampon=='N')
       {
       LED_jaunej2=0;
       delay_ms(3000);
       }

       if(Tampon=='J')
       {
       LED_bleuj1=0;
       delay_ms(3000);
       }

       if(Tampon=='C')
       {
       LED_bleuj2=0;
       delay_ms(3000);

       if(Tampon=='T')
       {
       LED_rouge=0;
       delay_ms(3000);
       score1=score1+1;
       G2=0;

                switch(score1)
                {
```

```
                case 1: PORTA=1000000;
                        break;
                case 2: PORTA=1111001;
                        break;
                case 3: PORTA=0100100;
                        break;
                case 4: PORTA=0110000;
                        break;
                case 5: PORTA=0011011;
                        break;
               case 6: PORTA=0011010;
                        break;
               case 7: PORTA=0000011;
                        break;
                case 8: PORTA=1111000;
                        break;
                case 9: PORTA=0000000;
                        break;
               case 10: PORTA=0010000;
                        break;
               }
          }

        if(Tampon=='G')
        {
        LED_verte=0;
        delay_ms(3000);
        score2=score2+1;
       G4=0;

               switch(score1)
               {
                case 1: PORTA=1000000;
                        break;
                case 2: PORTA=1111001;
                        break;
                case 3: PORTA=0100100;
                        break;
                case 4: PORTA=0110000;
                        break;
                case 5: PORTA=0011011;
                        break;
               case 6: PORTA=0011010;
                        break;
               case 7: PORTA=0000011;
                        break;
               case 8: PORTA=1111000;
                        break;
               case 9: PORTA=0000000;
                        break;
               case 10: PORTA=0010000;
                        break;
               }
          }
        }



       };
}
```

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 13/10/2010
Author  : F4CG
Company : F4CG
Comments:


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

#include <mega8535.h>

#define BJ1 PINA.0
#define TJ1 PINA.1
#define JJ1 PINA.2
#define BJ2 PINA.3
#define TJ2 PINA.4
#define JJ2 PINA.5
#define MJ1 PINA.6
#define MJ2 PINA.7



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

// Declare your global variables here
unsigned char var;
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;
//}

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=P State6=P State5=P State4=P State3=P State2=P State1=P State0=P
```

```
PORTA=0xFF;
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=Out Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x00;
DDRD=0x02;

// 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: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x20; // UDRE = 1
UCSRB=0x08;
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;
UDR = 0x55;
while (1)
      {
      if ( BJ1==0)
      {
      var='B';
       USART_Transmit(var);
      }

      if ( TJ1==0)
      {
      var='T';
       USART_Transmit(var);
      }

      if ( JJ1==0)
      {
      var='J';
       USART_Transmit(var);
      }

      if ( BJ2==0)
      {
      var='N';
       USART_Transmit(var);
      }

      if ( TJ2==0)
      {
      var='G';
       USART_Transmit(var);
      }

      if ( JJ2==0)
      {
      var='C';
       USART_Transmit(var);
      }

      if ( MJ1==0)
      {
      var='M';
       USART_Transmit(var);
      }

      if ( MJ2==0)
      {
```

```
        var='D';
         USART_Transmit(var);
        }

        }
}
```