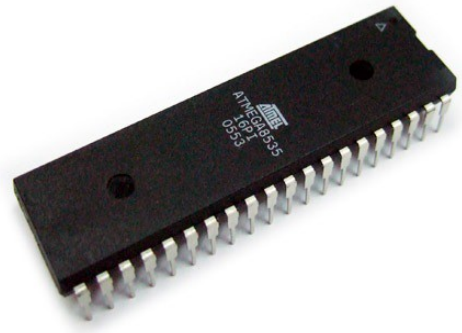
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle



Programmation sur ATmega

Études et réalisations

GERANTON Sylvain
Groupe P1
Promotion 2011-2013

LEQUEU Thierry
RODIER Sofi

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

Programmation sur ATmega

GERANTON Sylvain
Groupe P1
Promotion 2011-2013

LEQUEU Thierry
RODIER Sofi

Sommaire

Introduction

Le micro-contrôleur est un composant très utilisé dans le monde de l'informatique actuel, il permet de faire des programmes ou alors de faire des calculs à une très grande vitesse.

Ce composant peut se programmer en langage C, un langage de programmation très courant et très utilisé, dans ce rapport, nous verrons comment peut-on le programmer et surtout que peut-on faire de ce composant c'est pourquoi nous avons décidé de réfléchir à comment bien exploiter les capacités de L'ATmega.

Ce micro-contrôleur est utilisé dans des cartes de développement que nous avons à notre disposition à l'IUT, et nous avons été amenés à réfléchir sur sa programmation, nous verrons ce que nous avons décidé de programmer pour ensuite nous intéresser au programme et à ses modalités, finalement nous verrons quels en ont été les résultats et nous répondrons à la question de comment peut-on programmer avec un programmateur USB.

1. Cahier des charges

Nous avons donc décidé de programmer quelque chose sur l'ATmega, nous sommes parti sur l'idée d'un programme ludique et qui permettrait de voir les limites du processeur.

1.1. Analyse du projet

On a été amené à travailler sur les cartes de développement ATmega avec un micro-contrôleur 8535 intégré afin de les prendre en main pour améliorer nos compétences en programmation.

La carte étant fournie avec un écran LCD, nous avons décidé de voir comment on pouvait le programmer. L'idée d'un jeu utilisant le LCD a rapidement été envisagée, il fallait déterminer quel type de jeu nous allions coder.

On voulait quelque chose d'intéressant qui offre un peu de challenge et qui utilise les capacités de calcul de l'ATmega, le jeu devait être original pour ce micro-contrôleur et utiliser d'autres types de jeux hormis les morpions.

On est parti sur l'idée d'un jeu dans lequel on joue une voiture qui peut se déplacer sur un axe horizontal tandis que d'autres arrivent par le haut et qu'il faut les éviter, le défilement (ou scrolling) était envisagé en étant vu du dessus mais afin de faire faire plus de calculs à l'ATmega on à mis en place un scrolling parallaxe bilinéaire utilisé par SEGA dans des jeux tels que OUTRUN, SPACE HARRIER ou HANG-ON.

Illustration 1: OUTRUN

Le principe du scrolling parallaxe bilinéaire est de faire avancer des lignes parallèles à une certaine vitesse et de les disposer selon un certain angle afin de créer une sensation de profondeur. Cette technique fonctionne aussi avec des effets de distorsion d'objets placés sur les cotés afin de créer l'illusion qu'on avance, sans que l'image de la voiture ne bouge.

L'écran LCD de la carte de programmation étant petit (4 lignes par 16 colonnes) et ne disposant pas de plus grand écran à l'IUT, nous avons programmé avec l'écran par défaut dont il va falloir modifier les caractères à notre disposition.

Le tout était de créer un jeu :

- Jouable
- Intéressant
- Avec du challenge
- Qui exploite les capacités de calcul de l'ATmega

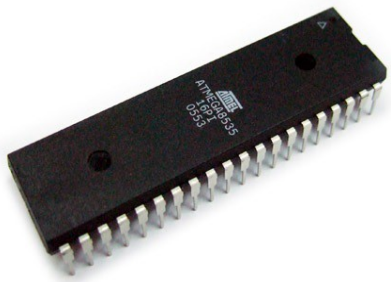


Illustration 2: ATmega 8535

L'ATmega 8535 est un micro-contrôleur 8 bits crée par la société Atmel, ce type de micro-contrôleur est très utilisé dans des applications nécessitant l'utilisation d'un programme informatique sans devoir investir dans de gros microprocesseurs pour l'exécution d'un petit programme.[1]

1.2. Planning

Afin de voir comment nous allions pouvoir coder cette application, nous avons établi un planning des tâches à réaliser.

Séance	1	2	Vacances		3	4	5	6	7
Réflexions sur le projet									
Réflexions sur les contraintes									
Prise en main CodeVision									
Prise en main ATmega8535									
Codage									
Débogage du programme et phase de tests									
Finitions									

Tableau 1: Planning

Avec en rouge le prévu, et en vert le réalisé, on remarque que nous avons modifié le prévisionnel en décidant d'effectuer la programmation en même temps que la prise en main de CodeVision et du micro-contrôleur afin de voir si de nouvelles contraintes apparaissaient.

2. Algorithmes et programmation

Un programme informatique afin d'en simplifier le codage est composé d'algorithmes qui permettent de voir les différentes opérations ou comparaisons qui peuvent être appliqué à celui-ci

2.1. Algorithmes

2.1.1. Programme de démarrage

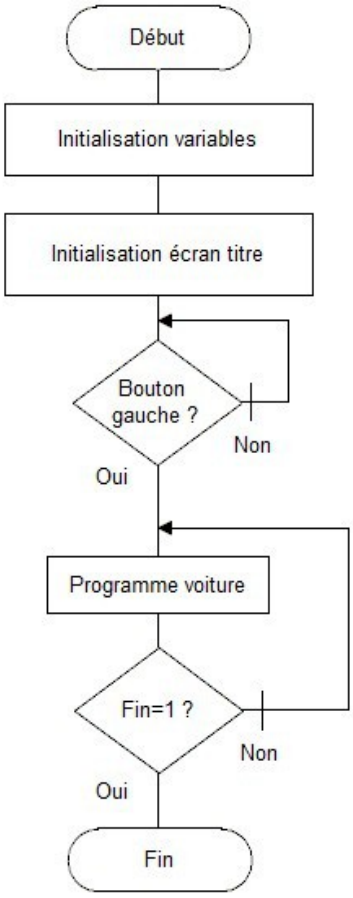


Illustration 3: Algorithme de démarrage

Cette partie du programme consiste à le démarrer en faisant l'initialisation des variables et de l'écran, puis faire les tests de démarrage en contrôlant l'appui sur un bouton, si le bouton à été enfoncé, on appelle le programme principal.

2.1.2. Programme principal

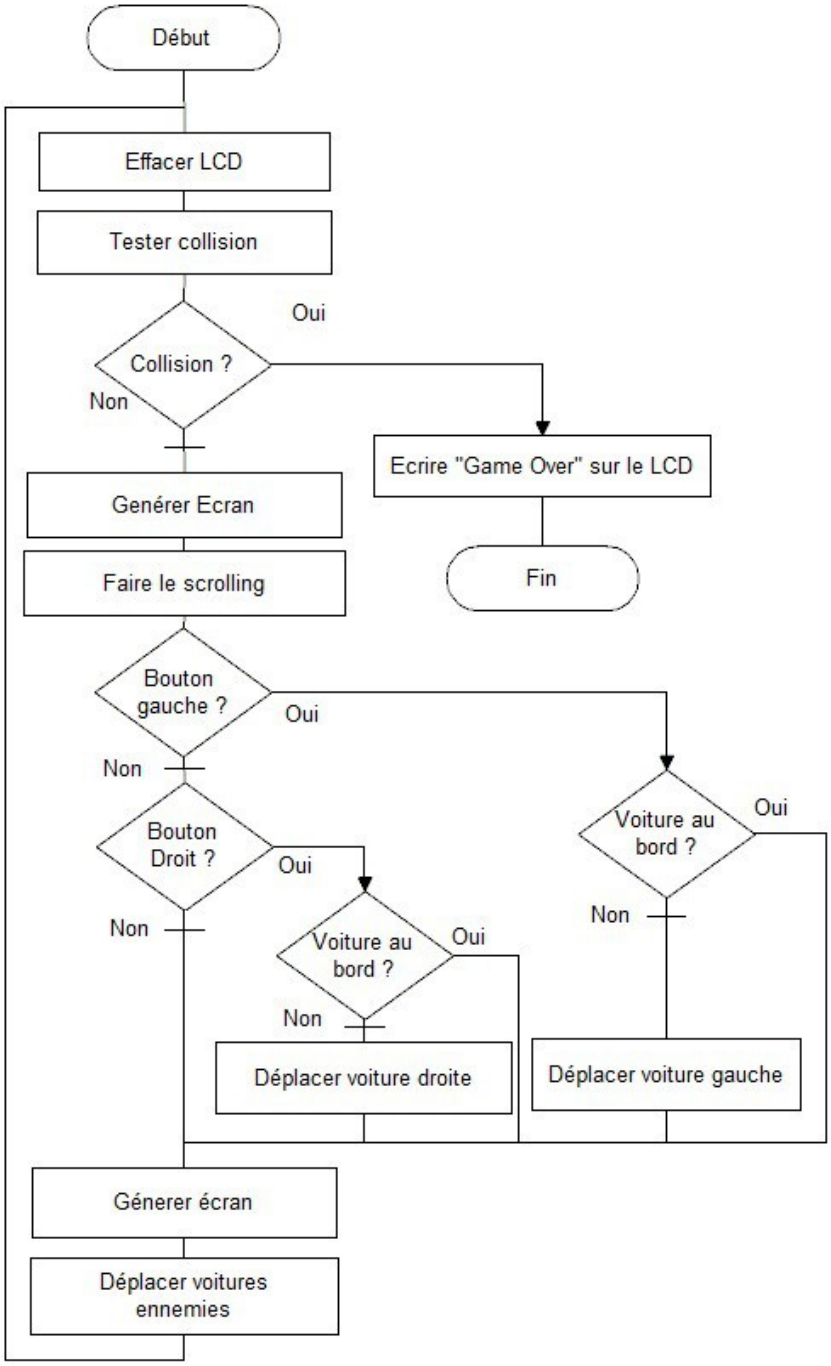


Illustration 4: Algorithme du programme principal

Cet algorithme est le programme principal, il teste en premier lieu, l'existence ou non d'une collision et ne continue à s'exécuter que s'il n'y en a pas eu. Ensuite après avoir généré l'écran et fait le déplacement des caractères, il teste l'appui sur les boutons et fait le déplacement en conséquence après avoir testé si la voiture ne sortait pas de l'écran. Enfin le programme régénère l'écran et déplace les voitures ennemies.

2.1.3. Programme scrolling

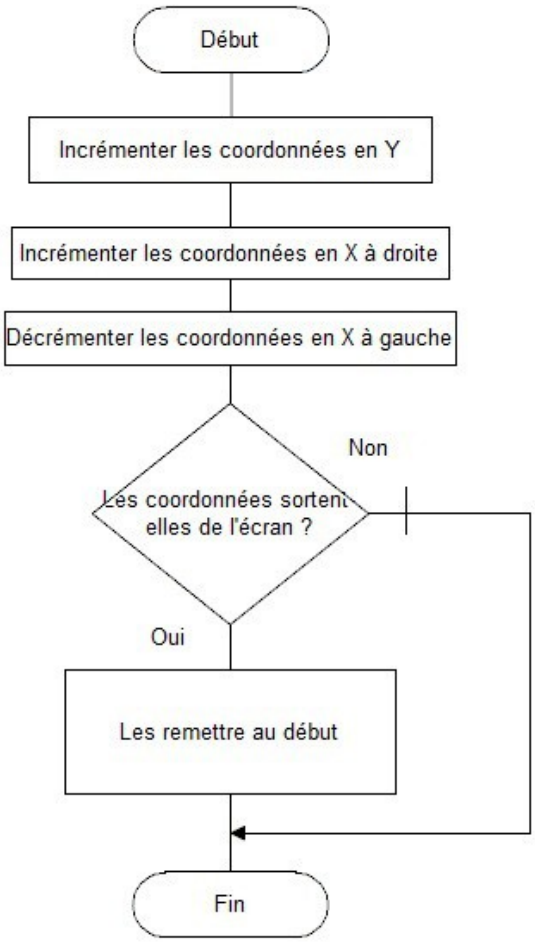


Illustration 5: Algorithme de gestion du scrolling

Cet algorithme est responsable du déplacement des caractères sur l'écran LCD, lorsqu'il est appelé, il change les coordonnées en Y et en X en leurs appliquant des additions de coordonnées. Enfin le programme teste si certaines coordonnées sortent de l'écran afin de les régénérer au début de celui-ci.

2.1.4. Programme génération de l'écran

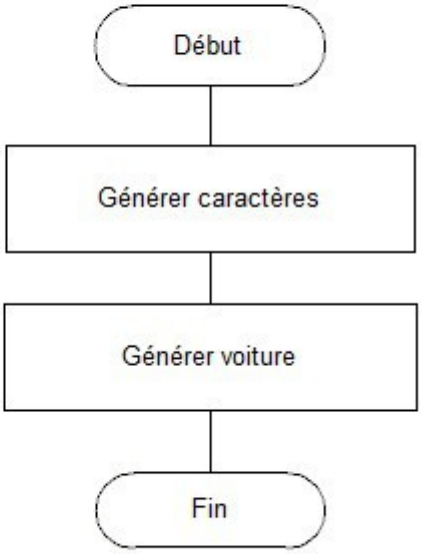


Illustration 6: Algorithme de génération de l'écran

Cet algorithme est celui qui permet de placer les caractères et la voiture sur l'écran en générant les caractères aux coordonnées souhaitées.

2.1.5. Programme gestion des collisions

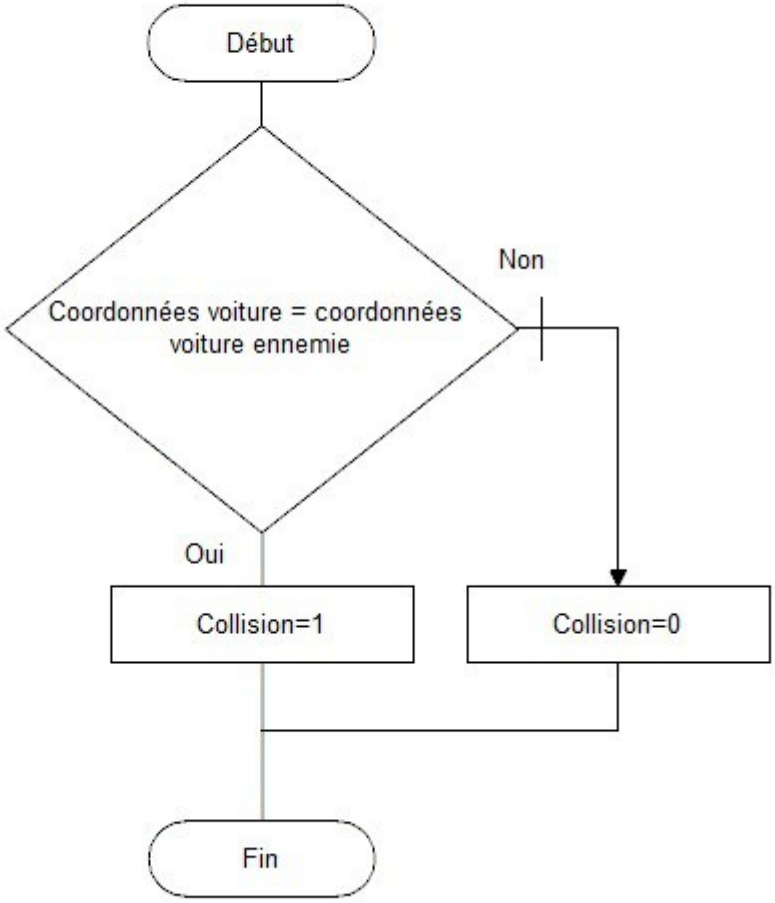


Illustration 7: Algorithme de gestion des collisions

Cet algorithme effectue le calcul des coordonnées et regarde si les coordonnées d'une voiture ennemie sont égales à celles de la voiture du joueur. Si c'est le cas il change l'état de la variable de collision.

2.1.6. Programme direction des voitures

Cet algorithme est le plus important car il gère le déplacement des voitures ennemies ainsi que leur positionnement qui obéira à un nombre aléatoire. Ce nombre aléatoire permet de savoir si on déplace la voiture ennemie vers la gauche ou vers la droite avant de la descendre.

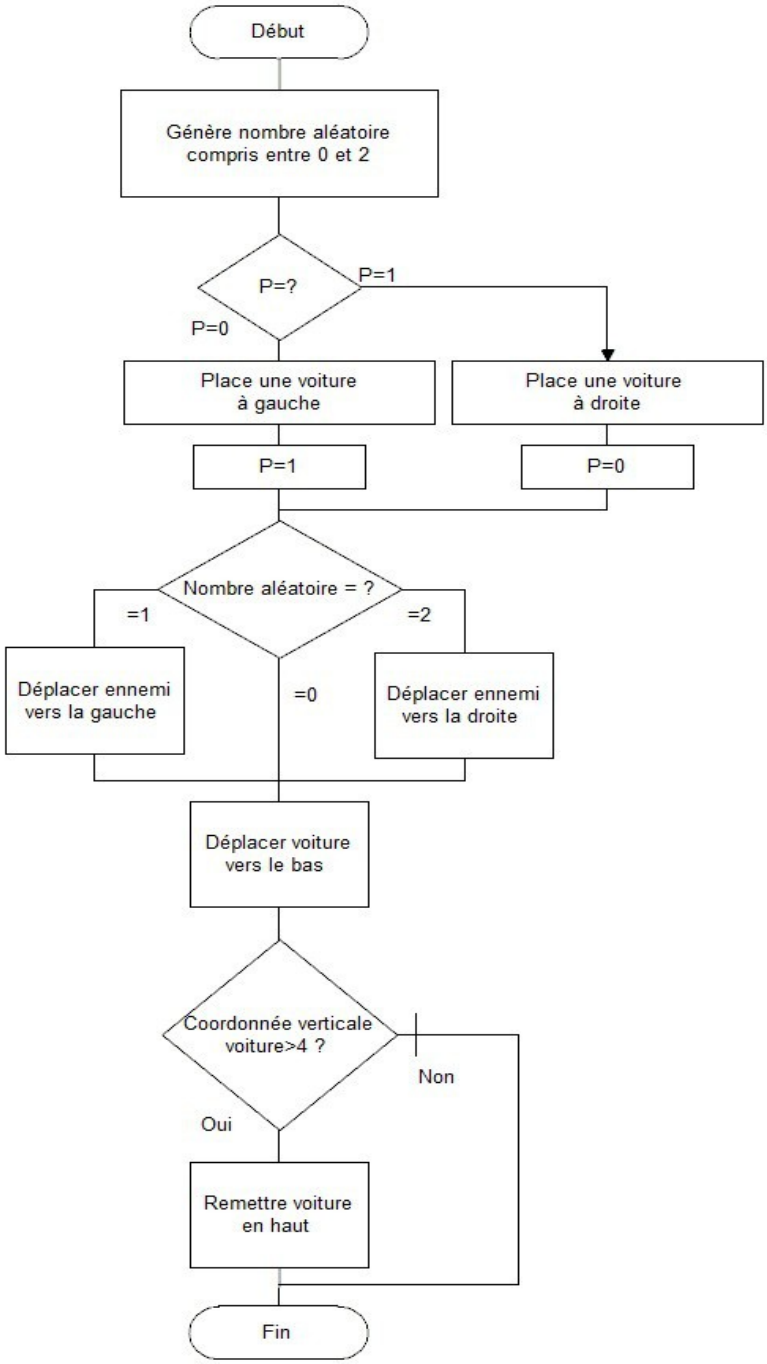


Illustration 8: Algorithme de gestion des ennemis

2.2. Carte de développement

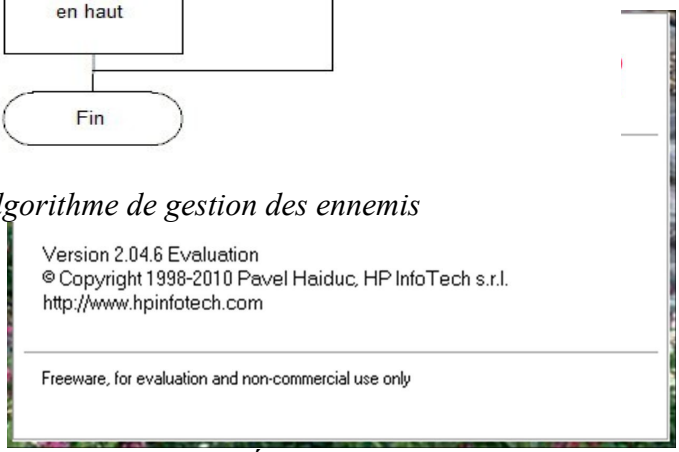


Illustration 10: Écran d'accueil de CodeVision

La carte de développement ATmega est une carte fabriquée par l'IUT qui dispose d'un ATmega8535, d'un écran LCD, de deux boutons et de ports de programmation. La carte est alimentée par une prise secteur 230V.

2.3. Programmation

Nous avons programmé à l'aide de CodeVision qui est un environnement de développement intégré très puissant et qui possède de bonnes capacités de débogage.

Ce logiciel n'étant pas gratuit, nous nous sommes procurés une version d'évaluation qui est limitée en taille de programme. Comme nous utilisons uniquement des variables entières, nous ne devrions pas dépasser les capacités maximales de la version d'essai.

Pour programmer sur l'ATmega, il faut utiliser un connecteur de type J-tag / HE10 que l'on branche sur un des ports parallèles de l'ordinateur.

Lorsqu'on compile sur CodeVision et qu'on veut transférer le programme sur l'ATmega, CodeVision appelle un autre programme appelé AVR Studio conçu par Atmel afin de traduire le programme de CodeVision en un programme compréhensible et utilisable pour l'ATmega.

Sur le programme, nous avons utilisé plusieurs bibliothèques à savoir :

- <mega8535.h> qui est la bibliothèque de base pour faire fonctionner l'ATmega ;
- <DELAY.H> bibliothèque nécessaire pour les temporisations ;
- <stdlib.h> bibliothèque nécessaire pour les variables aléatoires ;
- <lcd.h> bibliothèque pour l'écran LCD ;

Les variables aléatoires sont nécessaires pour le déplacement des véhicules car l'état de la variable sert à définir la direction du véhicule lorsqu'il va changer de ligne sur le LCD.

Nous voulions utiliser des caractères de type slash et antislash pour créer un genre d'entonnoir à l'écran afin de donner une sensation de profondeur. Mais le caractère antislash n'était pas intégré aux caractères de base du LCD, et le créer reviendrait à manipuler les registres de la RAM du driver de LCD, ce qui n'était pas aisé à comprendre et à mettre en œuvre.

3. Tests

Maintenant que nous savons comment nous allons programmer, nous allons tester le programme et voir s'il y a eu des problèmes et les solutions apportées

3.1. Les tests

Les tests se sont faits au fur et à mesure de la programmation, car ils permettaient de voir les limites de l'ATmega en matière de calcul. De plus ils nous montraient les limites de la programmation, notamment lors d'un test où la très mauvaise gestion des boucles et des conditions à fait planter le programme.

L'ATmega est plutôt limité dans la puissance de calcul puisque placer plusieurs boucles et tests dans un programme le ralentit fortement, voire l'arrête complètement.

Les calculs de scrolling le ralentit aussi fortement, il peut difficilement gérer des déplacements de caractères lorsque ceux-ci sont trop nombreux sur l'écran.

Les tests de l'affichage nous ont pris plusieurs semaines car les calculs de coordonnées causaient quelque problèmes notamment lors des calculs de variables pour le déplacement des caractères.

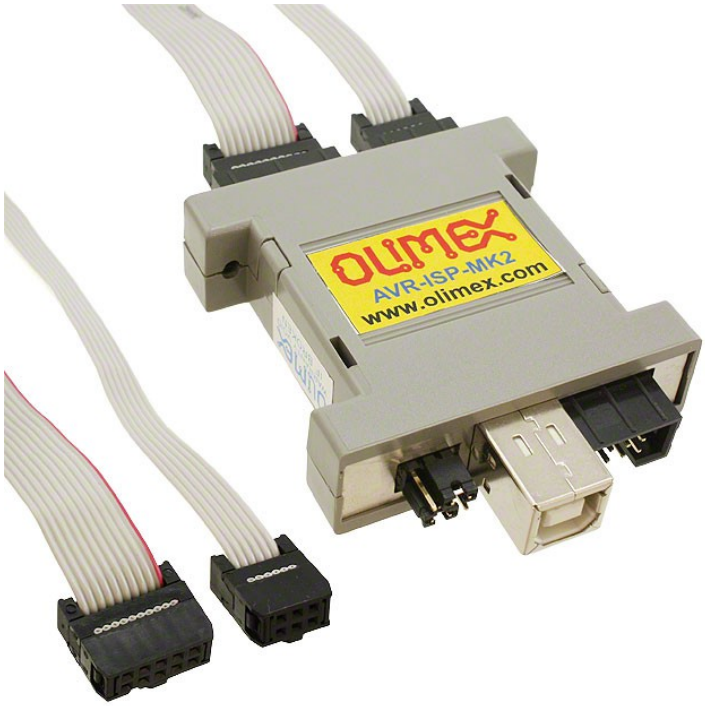


Illustration 11: OLIMEX AVR-ISP-MKII

3.2. Programmation par USB

Durant les séances, nous avons cherché à pouvoir programmer chez nous afin de pouvoir nous avancer dans notre travail. Mr Thierry LEQUEU[2] qui avait à sa disposition un programmateur USB, nous a permis de pouvoir essayer son fonctionement sur notre ordinateur personnel tournant sous WINDOWS 7 64 bits, le mettre en place était très compliqué puisque la notice ne nous éclairait pas vraiment sur le mode opératoire. C'est pourquoi nous avons décidé de l'écrire afin de le mettre à disposition pour les futurs étudiants.

Le programmateur est un programmateur de la marque OLIMEX de référence AVR-ISP-MkII, ce programmateur est connu de CodeVision. Il suffisait juste d'installer les pilotes de logiciels après les avoir téléchargés sur le site du fabriquant, pour la procédure complète, référerez-vous à l'annexe 1 où est détaillée la mise en œuvre sur un WINDOWS 7 et un WINDOWS XP.

Conclusion

Programmer le micro-contrôleur n’a pas été une chose simple car il fallait maîtriser le composant relativement capricieux ainsi que le logiciel. De plus faire fonctionner le programmateur USB à pris plus de temps que prévu et n'était pas prévu à la base dans le projet.

Le programme est quasiment fini il y a juste quelques problèmes avec les nombres aléatoires puisque le programme ne veut pas les regarder, sinon les collisions, le scrolling ainsi que le déplacement réagissent très bien.

Si nous avions eu un peu plus de temps, peut-être que nous aurions pu mettre un score relié à la vitesse des ennemis, plus le score est élevé plus les ennemis accélèrent ou alors modifier le code de l'afficheur LCD pour pouvoir créer nos propres caractères.

Le codage des algorithmes s'est bien passé et CodeVision était vraiment facile à utiliser et très clair sur l'origine des bugs. Le projet pourrait continuer si on avait à disposition une carte et un programmateur personnels, mais ce n'est pas le cas. Si un futur étudiant voulait l'améliorer, nous lui céderions notre programme avec plaisir.

Résumé

Ce projet à pour but de programmer un jeu sur une platine de développement ATmega8535 de l'IUT, le jeu se composait de voitures gérées par le programme et on devait les éviter avec la notre en utilisant les boutons de la platine.

La programmation s'est découpée en deux parties. Une partie d'analyse des algorithmes et une autre partie de programmation et de débogage avec un logiciel que nous devions apprendre à maîtriser.

La partie analyse a permi d'établir les routines nécessaires au bon fonctionnement du programme, et la partie programmation et débogage à permi de voir ce qui marchait et ce qui avait besoin d'améliorations.

Le projet est quasiment arrivé à terme hormis quelques problèmes de nombres aléatoires qui empêchaient les autre voitures de se déplacer sur l'écran pour venir nous percuter.

Index des illustrations

Bibliographie

[1] **Atmel Corporation**, "*8-bit Microcontroller with 8K Bytes In-System Programmable Flash*", , 2010.

[2] **Thierry Lequeu**. *La documentation de Thierry LEQUEU sur OVH*, 2013, [En ligne]. (Page consultée le 7 Avril 2013) <http://www.thierry-lequeu.fr/>

Table des annexes

Annexe 1 : Notice programmateur

Programmateur USB Olimex AVR-ISP-MK2

Mode Opératoire

Remarques générales :

ATTENTION : Il peut y avoir un conflit d'alimentations car si vous branchez l'adaptateur secteur en plus du programmateur, l'écran LCD peut réagir bizarrement. Vous pouvez laisser le programmateur USB seul, mais placez le domino sur TARGET-ON, par contre vous pouvez dire adieu au rétroéclairage et à certains caractères. Mais si vous faites le choix de vous alimenter sur le secteur, enlevez le domino de tension placé en BAS du programmateur.

1-Si vous travaillez avec les ordinateurs de l'IUT, n'oubliez pas que tout peut être effacé, notre conseil est de conserver le dossier des firmwares téléchargés sur un support amovible tel qu'une clé USB.

2-Les firmwares du fabriquant existent aussi pour le programme AVR-DUDE sous LINUX, n'ayant pas de LINUX, nous n'avons pas pu tester cette solution même si on pense que la procédure doit globalement être la même.

3-Lors de l'installation du Firmware vous aurez deux dossiers dans le répertoire : C:/Program Files/Atmel/Atmel USB/SeggerUSB Driver, un dossier x64 et x86, le dossier x64 est destiné aux systèmes d'exploitations 32 bits, tandis que le dossier x86 est destiné aux systèmes d'exploitations 64 bits, si vous savez pas quel est votre système vous trouverez cette information dans les propriétés système du panneau de configuration.

4-Les cartes de programmation sont alimentées en +5V.

(Dernière révision le 27 Mars 2013)

Annexe 1 : Notice programmateur

Comment installer le programmateur USB Olimex AVR-ISP-MK2 sur son ordinateur personnel :

(test effectué avec un WINDOWS 7 64 bits (si vous possedez un 32 bits référez-vous à la remarque générale n°3 page 1) ainsi qu'avec un windows XP)

Télécharger et installer la version démo de CodeVision http://hpinfotech.ro/html/download.htm

Télécharger et installer AVR Studio 6 (obligatoire pour CodeVision) http://www.atmel.com/microsite/atmel_studio6/

Télécharger les Firmwares du programmateur USB sur le site du fabriquant, à l'adresse https://www.olimex.com/Products/AVR/Programmers/AVR-ISP-MK2/ décompresser le fichier ZIP, puis éxécuter le fichier AtmelUSB.exe dans le répertoire AVR-ATMEL-STUDIO de ce même fichier

Insérer le programmateur sur votre machine sans le connecter sur la carte à programmer et en plaçant le domino sur TARGET OFF

Pour WINDOWS 7

- Quand Windows va chercher le driver, dites-lui de ne pas chercher sur Windows Update, il devrait réussir à les trouver tout seul

Pour WINDOWS XP

- Allez chercher le périphérique dans le gestionnaire et demander d'installer son driver, il faut aller chercher le driver dans C:/Program Files/Atmel/Atmel USB/SeggerUSB Driver/x64

- Sur CodeVision, allez dans Settings/Programmer et sélectionnez le Atmel AVRISP MkII (USB)

- Passez sur Project/Configue et allez sur les onglets C Compiler/Libraries/AlphaNumeric LCD (alcd.h) et mettez tout sur le PORTC

- Une fois tout cela fini, revenez sur CodeVision, Compilez, connectez votre carte ATMEGA au programmateur, placez le domino sur TARGET ON si vous n'avez pas de bloc secteur, et envoyez le programme.

- Si vous voulez éviter que votre programme reboote quand vous enlevez le connecteur, placez le domino sur TARGET OFF une fois le programme lancé.

Comment installer le programmateur USB Olimex AVR-ISP-MK2 sur les ordinateurs de l'IUT :

(test effectué avec un WINDOWS XP Professionnel Service Pack 3)

- Contrairement à la procédure pour ordinateurs personnels, il faut juste vérifier si l'ordinateur à CodeVisionV2.03.4 et AVRStudio.exe, si l'ordinateur ne les possède pas, inutile de chercher à les télécharger, CodeVisionV2.03.4 ne fonctionnera pas et AVRStudio 4 peut encore exister sur la toile mais il est assez lourd.

- Si les logiciels sont installés, téléchargez les Firmwares du programmateur USB sur le site du fabriquant, à l'adresse https://www.olimex.com/Products/AVR/Programmers/AVR-ISP-MK2/ décompressez le fichier ZIP, puis éxécutez le fichier AtmelUSB.exe dans le répertoire AVR-ATMEL-STUDIO de ce même fichier.

- Une fois fini, branchez le programmateur avec le domino sur TARGET-OFF

- Exécutez CodeVision, allez dans Settings/Programmer et sélectionnez le Atmel AVRISP MkII (USB)

- Allez chercher le périphérique dans le gestionnaire et demander d'installer son driver, il faut aller chercher le driver dans C:/Program Files/Atmel/Atmel USB/SeggerUSB Driver/x64

- Compilez, placez le domino sur TARGET-ON, et envoyez, suivant la façon dont vous allez programmer, CodeVision peut afficher un message d'erreur "AVRISP MkII error reading Fuse Bits" cette erreur traduit une incapacité de CodeVision à lire les bits de protection de l'Atmega, si vous programmez "normalement", vous ne devriez pas avoir de problèmes avec ces bits.

Annexe 2 : Journal de bord

Semaine 1 : la prise en main de codevision est plutôt aisée, le logiciel est facile à prendre en main, faut juste faire attention à bien créer un projet et à le sauvegarder autre part que sur la clé USB (répertoire non valable)

Semaine 2 : j'ai décidé de me faire la main sur l'atmaga 8535 en le programmant, cela m'avancera pour la suite.

Je décide de faire un jeu de voiture où il faudra éviter les autres qui arrivent sur l'écran, afin de créer une sensation de profondeur sur cet écran, je vais me baser sur la technologie du scrolling parallaxe bilinéaire de SEGA apercu dans des jeux tels que Hang-On, Outrun ou Space Harrier.

Les lignes de codes pour le scrolling parallaxe me posent des problèmes : déjà le driver de l'afficheur LCD ne possède pas le caractère "\" dans sa bibliothèque, peut-être que je le programmerai plus tard en attendant on va mettre un "/" , de plus l'écran se brouille si on à raté quelque chose dans la programmation.

Fin de la séance j'ai fini le coté gauche du scrolling, j'ai commencé à attaquer le coté droit sans succès pour l'instant.

Semaine 3 : je reprends le coté droit, et réussit à le finir non sans heurts, je tente une structuration du programme et lui trouve un nom "Clio 70ch", qui sera affiché sur un pseudo écran-titre, c'était juste pour voir si la structuration ne posait pas de problèmes sur codevision.

Je pose un caractère "A" en bas de l'écran afin de symboliser la voiture, comme les jeux sur Atari 2600 faudra utiliser son imagination.

Je commence à implémenter les voitures ennemies, déjà ce sera un caractère de coordonnées X constant pour voir s'il s'adapte au scrolling, pas de problème pour la programmation mais j'ai fait une très mauvaise gestion des while, ce qui fait que ça fait ralentir le scrolling, faudra solutionner ça.

Pour changer un peu, j'ai commencé à coder "Jumping B", un prototype de jeu de plates-formes à scrolling horizontal comme Super Mario Bros. , le codage du scrolling horizontal est pénible, puisque je veux créer des trous, j'abandonne le proto pour l'instant et me concentre sur "Clio 70ch"

dont il reste encore à programmer le déplacement de la voiture, la trajectoire des ennemis ainsi que la hitbox, mais j'espère que le progammateur USB passera sur ma machine.

Annexe 2 : Journal de bord

Semaine 4 :

Comment installer le programmateur USB Olimex AVR-ISP-MK2 :

(test effectué avec un WINDOWS 7 64 bits et un WINDOWS XP)

-Télécharger et installer la version démo de CodeVision

-Télécharger et installer AVR Studio 6 (obligatoire pour CodeVision)

-Télécharger les Firmwares du programmateur USB sur le site du fabriquant, puis éxécuter le fichier AtmelUSB.exe dans le répertoire AVR-ATMEL-STUDIO

-Insérer le programmateur sur votre machine sans le connecter sur la carte à programmer et en plaçant le domino sur TARGET OFF

Pour WINDOWS 7

-Quand Windows va chercher le driver, dites-lui de ne pas chercher sur Windows Update, normalement ça devrait marcher

Pour WINDOWS XP

-Allez chercher le périphérique dans le gestionnaire et demander d'installer son driver, il faut aller chercher le driver dans C:/Program Files/Atmel/Atmel USB/SeggerUSB Driver

-Sur CodeVision, allez dans Settings/Programmer et sélectionnez le Atmel AVRISP MkII (USB)

-Passez sur Project/Configue et allez sur les onglets C Compiler/Libraries/AlphaNumeric LCD (alcd.h) et mettez tout sur le PORTC

-Une fois tout cela fini, revenez sur CodeVision, Compilez, connectez votre carte ATMEGA au programmateur, placez le domino sur TARGET ON, et envoyez le programme.

-Si vous voulez éviter que votre programme reboote quand vous enlevez le connecteur, placez le domino sur TARGET OFF une fois le programme lancé.

Annexe 2 : Journal de bord

Semaine 5 : Je finis la note pour le programmateur OLIMEX, peut-être que mon projet se transformera en mise en application de programmateurs USB, au moins il sera utile aux futurs étudiants, mais comment je vais expliquer ça dans le dossier, je pense que je vais faire mon programme en parallèle de ceci, je peux dire adieu à Jumping B, mais j'ai programmé la HitBox qui marche relativement bien, je m'attelle au déplacement de la voiture.

Semaine 6 : La programmation du déplacement de la voiture est terminée, elle ne sort pas de l'écran et la hitbox réagit globalement bien hormis quelques bugs d'affichage, la trajectoire des voitures est difficile à programmer car les nombres aléatoires n'existent pas pour le programme, faudra finir ça rapidement, je m'attaque à la rédaction du dossier des ce midi ou ce soir.

Semaine 7 : La majeure partie de la séance à été consacrée à la programmation des nombres aléatoires régissant le déplacement des voitures sur l'écran, ceci n'a pas été concluant, les voitures ne réagissent toujours pas à ces changements, l'autre partie ayant été consacrée à la rédaction du rapport d'E&R.

Annexe 3 : Programme complet

```
/*****************************************************
Project : Jeu
Version : 1
Date    : 28/03/2013
Author  : Sylvain GERANTON
Company :
Comments:
Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************************/

// Bibliothèques
#include <mega8535.h>
#include <DELAY.H>
#include  <stdlib.h>

// Branchement du LCD
#asm
   .equ __lcd_port=0x15
#endasm

// Bibliothèque LCD
#include <lcd.h>
```

Annexe 3 : Programme complet

```
// Declaration des prototypes
void clio(void);
void genecranC(void);
void paralaxe(void);
void IAC (void);
void hitbox (void);

  // Declaration des variables

    int XV=7;  // Position voiture
    int Hit=0; // Detection collision
    int flag=0; // Detection front montant
    int U=0;    // Valeur aleatoire
    int YIAC=0; // Coordonnee verticale des ennemis
    int XIAC=0; // Coordonnee horizontale des ennemis
    int P=0;    // Changement coordonnées voiture
    int end=0;  // Detection fin de programme

    // Ecran
      //Coté gauche

      //Etoile gauche haut
      int XEGH=1;
      int YEGH=3;

      //Slash gauche haut
      int XSGH=3;
      int YSGH=1;
```

Annexe 3 : Programme complet

```
//Etoile gauche bas
int XEGB=2;
int YEGB=2;

//slash gauche bas
int XSGB=4;
int YSGB=0;

//Coté droit

//Etoile droite haut
int XEDH=11;
int YEDH=0;

//Slash droite haut
int XSDH=12;
int YSDH=1;

//Etoile droite bas
int XEDB=13;
int YEDB=2;

//Slash droite bas
int XSDB=14;
int YSDB=3;
```

Annexe 3 : Programme complet

```
void main(void)
{

  //Port A en entrée
  PORTA=0x00;
  DDRA=0x00;

  //Port B en entrée
  PORTB=0x00;
  DDRB=0x00;

  //Port C en sortie
  PORTC=0x00;
  DDRC=0xFF;

  //Port D en entrée
  PORTD=0x00;
  DDRD=0x00;

  //Initialisation timers
  TCCR0=0x00;
  TCNT0=0x00;
  OCR0=0x00;

  TCCR1A=0x00;
  TCCR1B=0x00;
  TCNT1H=0x00;
```

```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

//initialisation interruptions externes
MCUCR=0x00;
MCUCSR=0x00;

//initialisation interruptions timers
TIMSK=0x00;

//initialisation comparateurs analogiques
ACSR=0x80;
SFIOR=0x00;

//initialisation LCD
lcd_init(16);
```

Annexe 3 : Programme complet

```
//Programme CLIO
srand(2);
lcd_clear();
lcd_gotoxy(3,0);
lcd_putsf("CLIO  70Ch");
lcd_gotoxy(1,1);
lcd_putsf("-");
lcd_gotoxy(2,1);
lcd_putsf("-");
lcd_gotoxy(3,1);
lcd_putsf("-");
lcd_gotoxy(4,1);
lcd_putsf("-");
lcd_gotoxy(5,1);
lcd_putsf("-");
lcd_gotoxy(6,1);
lcd_putsf("-");
lcd_gotoxy(7,1);
lcd_putsf("-");
lcd_gotoxy(8,1);
lcd_putsf("-");
lcd_gotoxy(9,1);
lcd_putsf("-");
lcd_gotoxy(10,1);
lcd_putsf("-");
lcd_gotoxy(11,1);
lcd_putsf("-");
lcd_gotoxy(12,1);
```

```
        lcd_putsf("-");
        lcd_gotoxy(13,1);
        lcd_putsf("-");
        lcd_gotoxy(14,1);
        lcd_putsf("-");
        lcd_gotoxy(2,2);
        lcd_putsf("Appuyez  sur  les deux boutons");
        srand(1);

    while(PIND.1!=0);  //test bouton de démarrage
        if(PIND.1==0)
          {
            do
              {

              clio();

              }
            while(end!=1);
          }
}


/*****************************************/
/*Programme principal*/
/*****************************************/

void clio (void)
{
```

Annexe 3 : Programme complet

```
lcd_clear();
      hitbox();



  if (Hit==0)
  {
    genecranC();
     U=rand();
     delay_ms(100);
     paralaxe();
        if(PIND.1 ==0)
          {
            if(XV==2) //test limite écran gauche
              {
              XV=2;
              }
            else
              {
              XV=XV-1;
              }
          }
        else if(PIND.0==0)
          {
            if(XV==13) //test limite écran droit
              {
              XV=13;
              }
            else
```

```
                {
                XV=XV+1;
                }
            }
    genecranC();
    IAC();
}


if(Hit==1) //test
{
    if(flag==0)
        {
        lcd_clear();
        genecranC();
        lcd_gotoxy(XV,YIAC);
        lcd_putchar (0xFF);
        delay_ms (400);
        flag=1;
        }
    else if (flag==1)
        {
        lcd_gotoxy(4,1);
        lcd_putsf ("GAME");
        lcd_gotoxy(7,2);
        lcd_putsf ("OVER");
        delay_ms(4000);
        flag=0;
```

```
            Hit=0;
            end=1;
            }
      }
}

/*********************************************************/
/*Programme gestion des collisions*/
/*********************************************************/
void hitbox (void)
{
   if ((XV==XIAC) && (YIAC == 3))
      {
      Hit=1;
      }
   else
      {
      Hit=0;
      }

}

/*********************************************************/
/*Programme génération écran*/
/*********************************************************/
void genecranC (void)
{
```

```
        //cote gauche



        lcd_gotoxy(XSGH,YSGH);
        lcd_putsf("/");
        lcd_gotoxy(XSGB,YSGB);

        lcd_gotoxy(XEGH,YEGH);
        lcd_putsf("/");
        lcd_gotoxy(XEGB,YEGB);
        lcd_putsf("*");

        //cote droit

        lcd_gotoxy(XSDH,YSDH);
        lcd_putsf("/");
        lcd_gotoxy(XSDB,YSDB);
        lcd_putsf("/");
        lcd_gotoxy(XEDB,YEDB);
        lcd_putsf("*");

        //voiture

        lcd_gotoxy(XV,3);
        lcd_putsf("A");
}
```

Annexe 3 : Programme complet

```
/*******************************************/
/*Programme gestion scrolling*/
/*******************************************/
void paralaxe(void)
{
    lcd_clear();
    YEGH=YEGH+1;
    YSGH=YSGH+1;
    YEGB=YEGB+1;
    YSGB=YSGB+1;

    XEGH=XEGH-1;
    XSGH=XSGH-1;
    XEGB=XEGB-1;
    XSGB=XSGB-1;

    YEDH=YEDH+1;
    YSDH=YSDH+1;
    YEDB=YEDB+1;
    YSDB=YSDB+1;

    XEDH=XEDH+1;
    XSDH=XSDH+1;
    XEDB=XEDB+1;
    XSDB=XSDB+1;

    //Etoile gauche haut
    if(XEGH==0)
```

```
    {
   XEGH=3;
   }
   if(YEGH==4)
   {
   YEGH=1;
   }

//Slash gauche haut
if(XSGH==0)
   {
   XSGH=3;
   }
if(YSGH==4)
   {
   YSGH=1;
   }

//Etoile gauche bas
if(XEGB==0)
   {
   XEGB=3;
   }
if(YEGB==4)
   {
   YEGB=1;
   }
```

Annexe 3 : Programme complet

```
//Slash gauche bas
    if(XSGB==0)
      {
      XSGB=3;
      }
    if(YSGB==4)
      {
      YSGB=1;
      }

    //Etoile droite haut
    if(XEDH==15)
      {
      XEDH=12;
      }
    if(YEDH==4)
      {
      YEDH=1;
      }

    //Slash droite haut
    if(XSDH==15)
      {
      XSDH=12;
      }
    if(YSDH==4)
      {
      YSDH=1;
```

```
        }

    //Etoile droite bas
    if(XEDB==15)
      {
      XEDB=12;
      }
    if(YEDB==4)
      {
      YEDB=1;
      }

    //Slash droite bas
    if(XSDB==15)
      {
      XSDB=12;
      }
    if(YSDB==4)
      {
      YSDB=1;
      }
}

/*****************************************************/
/*Programme direction des voitures*/
/*****************************************************/
void IAC (void)
{
```

Annexe 3 : Programme complet

```
U=rand();
if(P==0)
  {
     lcd_gotoxy(XIAC,YIAC);
     lcd_putchar(0xFF);
     delay_ms(200);
     YIAC++;
     if(U==0)
        {
        XIAC--;
        }
     else if(U==2)
        {
        XIAC++;
        }
     else;

     if(YIAC==4)
        {
        P=1;
        YIAC=0;
        XIAC=6;
        }
  }
 if(P==1)
  {

     lcd_gotoxy(XIAC,YIAC);
```

```
        lcd_putchar(0xFF);
        delay_ms(100);
        YIAC++;
          if(U==0)
            {
            XIAC--;
            }
          else if(U==2)
            {
            XIAC++;
            }
          else;
          if(YIAC==4)
            {
            P=0;
            YIAC=0;
            XIAC=9;
            }
    }
}
```