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

Projet PONG

Alexandre LUU
Nicolas MARTIN
Année 2013/2014
Groupe P1

Thierry LEQUEU
Véronique AUGER
Études et Réalisation

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

Alexandre LUU
Nicolas MARTIN
Année 2013/2014
Groupe P1

Thierry LEQUEU
Véronique AUGER
Études et Réalisation

Introduction

Durant ce 4ème semestre nous devons réaliser notre dernier projet d'étude et réalisation de notre DUT. Notre groupe, composé de Nicolas MARTIN et Alexandre LUU, a commencé par confronter les différentes idées possibles de projet. Cependant quelle que soit l'idée choisie, elle se doit d'être réalisable rapidement puisqu'en raison du stage de fin de semestre, nous avons moins de temps pour la mener à bien, à savoir 8 semaines. Après avoir consulté l'enseignant sur la difficulté de nos idées, nous avons décidé de créer un jeu PONG. Comme matériel, nous avons utilisé une carte arduino uno et un shield arduino composé d'un petit écran à LED. Ces cartes sont nouvellement utilisées en 1ère année, donc nous ne connaissons pas leurs caractéristiques. L'objectif que nous nous sommes fixé est d'utiliser du matériel méconnu afin de pouvoir joué au jeu PONG en mode 1 joueur. Ce projet va nous permettre d'utiliser nos compétences en programmation et en lecture de schéma électrique.

Comment créer un jeu PONG avec les ressources arduino de notre IUT et quelles sont les difficultés rencontrées ?

Tout d'abord, nous étudierons la carte et le shield arduino dans le but de comprendre leur fonctionnement. Puis nous verrons le résultat de notre travail et les possibilités de développement. Enfin nous expliquerons les différentes parties du programme pour savoir comment marche le jeu PONG.

Sommaire

1. Le matériel arduino

Nous avons décidé d'utiliser les cartes arduino pour mener à bien notre projet. Les raisons qui expliquent ce choix sont que la programmation sur arduino est simple et intuitive ; que cette carte était assez puissante pour réaliser le projet ; et que ces cartes étaient disponibles à l'IUT.

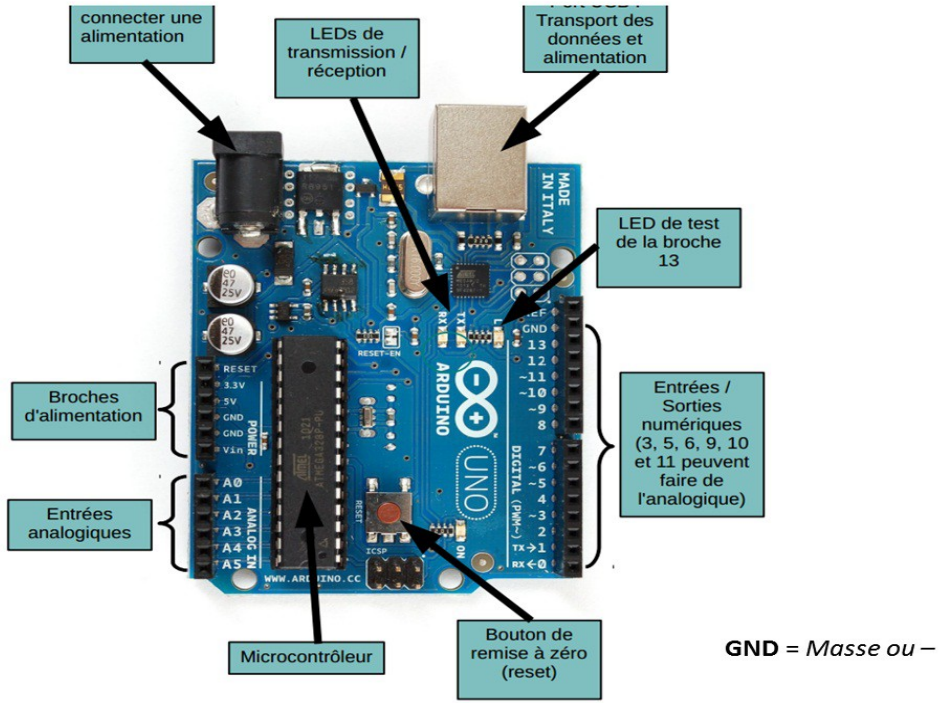


Illustration 1: Composition de la carte Arduino

1.1. La carte arduino Uno

Arduino est né en 2005 en Italie, par Massimo Banzi. C'est d'abord un outil qu'il a créé pour ses étudiants de l'Interaction Design Institute Ivrea. C'est une carte électronique qui permet des prototypages de façon intuitive, basée sur un microcontrôleur et des entrées/sorties. Aujourd'hui, arduino est devenu un vrai couteau suisse électronique autant pour les développeurs chevronnés que pour les néophytes de part sa conception simple. Cette carte électronique ainsi que son logiciel de programmation nommé (IDE : Integrated Development Environment), sont Open Source. Cela signifie qu'ils ne sont pas soumis aux droits d'auteur, et que quiconque peut les transformer, les améliorer et les redistribuer sous un autre nom. Les cartes Arduino, ou plus généralement de prototypage, ont pour but de simplifier grandement les circuits électroniques. En fait, une même carte peut être utilisée pour des applications différentes sur l'arduino comme un analyseur les données d'un capteur, allumer des LED ou activer un servomoteur.

Microcontrôleur	Atmega328
Tension d'alimentation (limites)	4V – 20V
Broches d'entrée/sortie numériques	14 (dont 6 PWM)
Broches d'entrée/sortie analogique	6
Mémoire flash	32KB

Tableau 1: Caractéristiques de l'arduino uno

La carte arduino « Uno » est la plus répandue et aussi la moins cher (une vingtaine d'euros). Elle possède donc un microcontrôleur et des capacités inférieures aux autres cartes. Cependant, une meilleure carte n'était pas indispensable pour réaliser notre projet. En effet la puissance d'une carte passe par son microcontrôleur qui permet de simplifier les circuits électroniques, de diminuer leur taille, de centraliser les processus car il est capable de réaliser une multitude de fonctions grâce à sa partie programmable. Ainsi pour un petit projet tel que le nôtre, le microcontrôleur de la uno (le ATmega328) est suffisant. Meilleur le microcontrôleur sera, plus la mémoire flash est grande et plus le programme que l'on peut implémenter dans la carte arduino peut être volumineux. Or pour notre programme la capacité de stockage de la uno est plus que suffisante (6 KB nécessaire pour 32 KB disponible).

En plus de sa simplicité d'utilisation et de son libre accès, de nombreuses entreprises et une large communauté ont contribué au développement de cartes pouvant être combinées avec les cartes arduino. Ces cartes sont nommées « Shields ».

1.2. Le shield arduino

Depuis 2005, une large famille de cartes a été développée, de l'arduino Nano, à l'arduino Tre. Ce sont des cartes électroniques complémentaires à l'arduino, que l'on peut rajouter directement sur l'arduino, pour lui ajouter des compétences. Par exemple, l'arduino motor shield permet de piloter 2 moteurs à courant continu, 2 servomoteurs, etc... à l'aide d'une alimentation extérieure à la carte arduino. Par ailleurs, les shields sont adaptés à la forme de la carte pour lesquelles ils sont fabriqués. Par exemple le shield que nous utilisons est un shield pour l'arduino uno et ne pourra donc pas fonctionner sur l'arduino mega 2560.

Notre Shield est l'écran OLED nommé 4D Systems 4Display Shield 128. L'écran OLED est un afficheur graphique couleur de 128 pixels par 128 pixels. Cependant nous n'utilisons pas de couleurs vives dans ce projet car le jeu Pong est un jeu en noir et blanc. La carte intégrant l'afficheur possède un joystick ainsi qu'un lecteur de cartes micro SD qui permet de stocker des images et de la

vidéo. Le dialogue entre l'écran OLED et la carte arduino UNO se fait par une liaison série. Nous utilisons cette carte afin de pouvoir afficher le jeu sur son écran et de pouvoir y jouer grâce à son joystick.

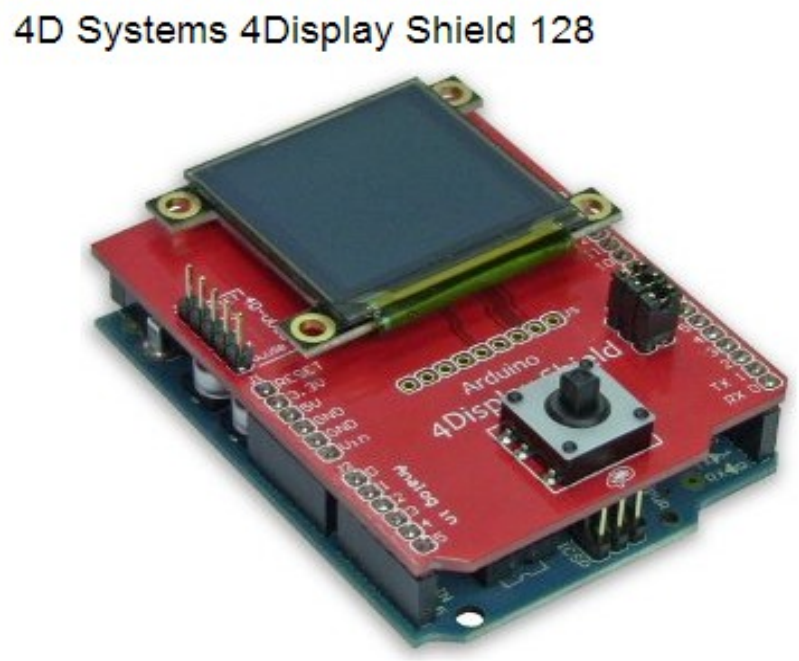


Illustration 2: Shield arduino OLED

Position du joystick	Broche du shield utilisée
Gauche	D3
Droite	D4
Haut	D6
Bas	D2
Appui	D5

Tableau 2: Relation : joystick/broche

Le joystick à 5 positions : appui, haut, Bas, Droite et gauche. Ces 5 actions du joystick sont connectées au microcontrôleur de la carte UNO via les entrées DIGITAL D2 à D6 (voir annexe « shield OLED »), comme indiqué ci-dessous :

Ainsi, comme les broches du shield s'emboitent dans les pins de la carte arduino, la commande du joystick se fera sur ces dernières. Soit respectivement comme le tableau ci-dessous :

Broche du shield	Pin de la carte uno utilisée
D3	3
D4	4
D6	6
D2	2
D5	5

Tableau 3: Relation broche/pin

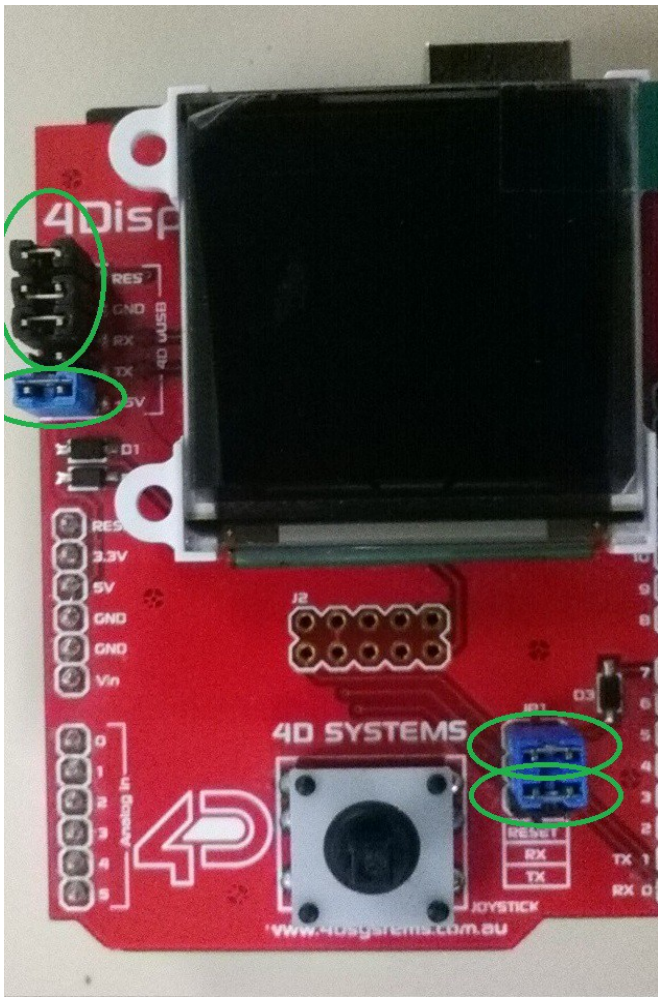


Illustration 3: Position des cavaliers

Enfin pour pouvoir utiliser le shield il faut brancher correctement les cavaliers[1] (voir ci-contre entourés en vert). Après plusieurs tests nous avons trouvé le bon emplacement de chaque cavalier. Le cavalier du TX devait être branché comme indiqué sur le cours de 1ère année, contrairement au 5V et au GND qui permettent d'alimenter l'écran, et au RX car il permet de faire communiquer la carte arduino uno avec l'écran.

C'est avec l'aide de ces cartes que nous avons décidé de mener notre projet de recréer le jeu Pong, tout en souhaitant respecter l'aspect rétro du jeu.

1 Cavalier : composant permettant de relier 2 broches

2. Le résultat

Le résultat attendu se devait d'être fidèle à l'œuvre origine du jeu PONG. Avec les cartes arduino et notre programme créé de toutes pièces (voir les parties 1 et 3), nous voulions pouvoir jouer à ce jeu de la même manière et avec la même interface qu'il y a plus de 40 ans, en utilisant nos propres compétences.

2.1. Le jeu PONG original

2.1.1. L'histoire

Illustration 4: Borne pong originelle

PONG est l'un des jeux les plus simples et les plus connus du monde car il est selon la croyance le tout premier jeu vidéo commercialisé. Sa première apparition se fait en 1958 dans le laboratoire de Brookhaven aux Etats Unis sur un oscilloscope. Cependant les chercheurs ne remarquent pas le potentiel commercial de cette création. Ce n'est que beaucoup plus tard qu'il fut développé par Nolan Bushnell créateur de l'entreprise Atari en 1972. Il devait être (pour les moyens de l'époque) une simulation de ping-pong. D'ailleurs le nom de Pong vient du mot ping-pong mais comme le nom Ping-Pong appartenait déjà à une marque, le jeu a été nommé simplement Pong. Il marqua le début de l'air des jeux vidéo par le biais de bornes d'arcade qui connurent un grand succès.

2.1.2. Le principe du jeu

Il s'agissait d'un principe très simple. Le joueur avait la possibilité de jouer contre un autre joueur ou contre une intelligence artificielle dans un jeu avec une balle. Les deux joueurs contrôlaient chacun une barre verticale située sur les bords gauche et droit d'un écran. Le but était d'empêcher la balle de sortir de son bord d'écran en faisait rebondir celle-ci à l'aide de la barre verticale. La balle rebondissait aussi sur les bords haut et bas de l'écran. De plus sa vitesse et son orientation dépendaient de la manière dont elle était frappée

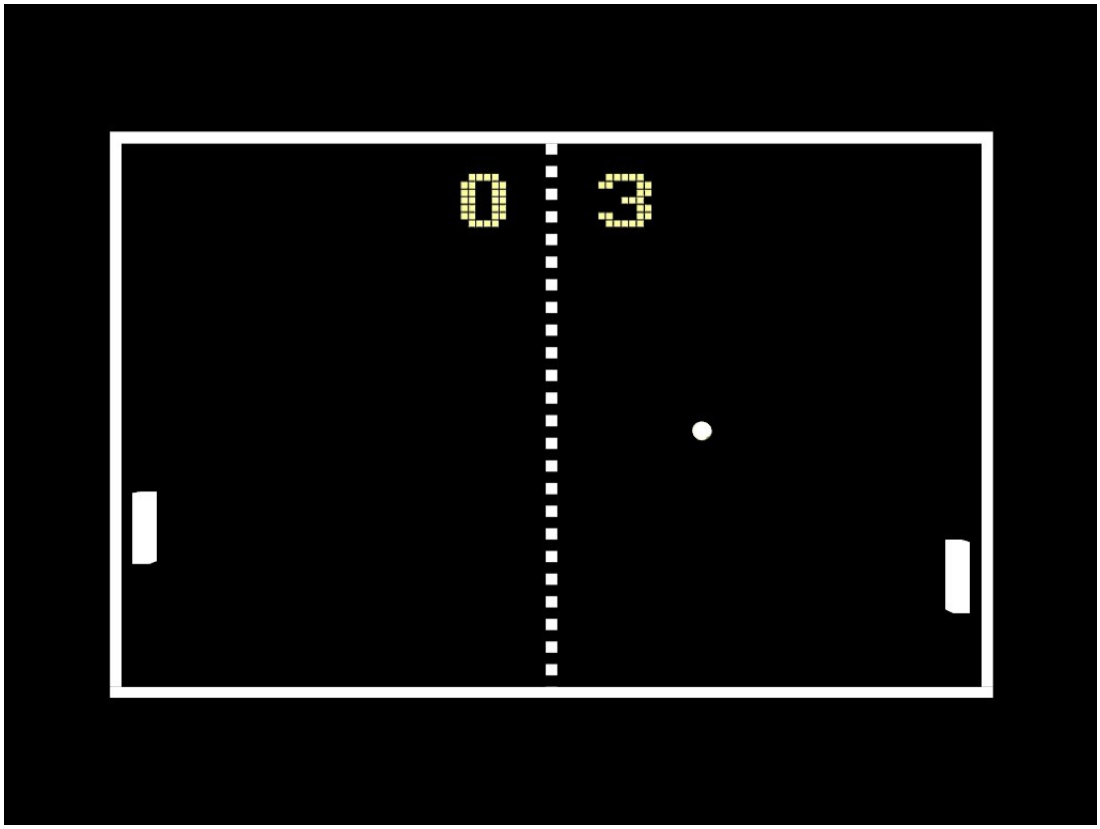


Illustration 5: Interface originelle de Pong

S'il perdait, le joueur adverse gagnait un point et la balle était remise en jeu sur une ligne délimitant le milieu du terrain de jeu. Il n'y avait pas de musique, mais il y avait quelques effets sonores lorsque la balle percutait une barre ou un bord ou qu'un point était marqué.

Voici donc le jeu PONG tel qu'il est réellement et que nous souhaitions imiter du fait de sa relative simplicité, du temps impartis et de l'apport pédagogique que cela nous apporterait. En revanche la création de notre jeu à fait face à quelques difficultés.

2.2. Les problèmes

Au cours de notre projet, nous avons rencontré plusieurs problèmes imprévus qui nous ont fait prendre du retard sur notre planning initial.

Tout d'abord nous n'avions pas pu obtenir rapidement la carte arduino nécessaire à notre projet et avons dû alors commencer à travailler sur une carte AT méga créée par M.Lequeu. Bien que cette carte soit très pratique d'utilisation, son souci était son écran. En effet il s'agissait d'un écran sur 4 lignes seulement. Ainsi la balle et les barres n'auraient eu que 4 positions possibles suivant l'axe des y. Cela ne convenait pas à ce que nous avions prévus car le mouvement de déplacement de la balle en serait faussé.

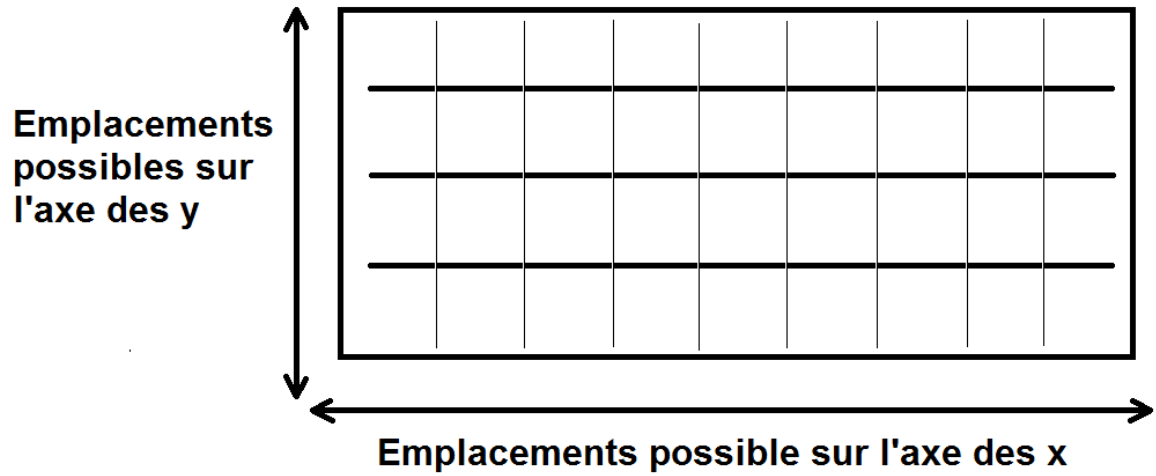


Illustration 6: Schéma de l'écran de l'AT méga

Après avoir obtenu le matériel souhaité nous avons néanmoins rencontré un autre problème ; à savoir que nous manquions de documentation sur l'équipement choisi. Ainsi beaucoup de temps a été consacré à la recherche de documentation afin de comprendre comment faire fonctionner l'écran du shield arduino. De plus nous souhaitions créer nous-même créer le programme sans s'inspirer des programmes facilement trouvables sur internet. Il a, alors, été difficile de crée la balle de manière à ce qu'elle rebondisse correctement sans créer de bug.

Par ailleurs d'autres bugs mineurs sur la programmation ont dû être, par la suite, résolus. Cependant, même si nous avons choisi un parcours plus difficile nous avons au final crée un jeu Pong en utilisant notre propre logique et nos propres connaissances.

2.3. Notre PONG

La première différence de notre version du jeu par rapport à l'original est qu'il n'y a pas de mode 2 joueurs. En effet, la carte ne possède qu'un seul joystick qui permet de commander une seule barre. Il n'est alors possible de jouer qu'en mode 1 joueur contre l'ordinateur programmé. Un mode 2 joueurs aurait pu être crée si nous avions eu plus de temps. Pour cela nous avions envisagé de fabriquer nous-même un autre shield afin d'ajouter des boutons ou un potentiomètre afin de pouvoir jouer la deuxième barre.

En ce qui concerne l'interface de jeu, elle est composée juste de la balle et des deux barres. Contrairement à la première version il n'y a pas de délimitation verticale de milieu de terrain ni de score affichés. Nous avons préféré nous concentrer sur la fluidité et la fiabilité du jeu plutôt que sur son apparence visuelle. De plus, l'écran du shield arduino étant très petit (128 par 128 pixel : 9 cm^2) il aurait été difficile de jouer correctement si celui-ci était surchargé graphiquement. Enfin il n'y a pas d'effet sonore pour la simple raison que nous n'avons mis aucun appareil sonore en liaison avec notre carte.

Notre jeu répond néanmoins aux principales caractéristiques du jeu Pong :

- Les 2 barres font rebondir la balle et celle-ci rebondit différemment en fonction de l'endroit de la barre où la balle tape.
- Elle rebondit contre les bords haut et bas de l'écran.
- Une des barres est contrôlée par un joueur via le joystick : pousser le joystick vers le haut ou vers le bas fait bouger la barre dans la même direction.

Nous avons aussi ajouté des éléments qui n'existaient pas avant.

- Lorsque la balle sort du bord d'un joueur, l'écran affiche alors message « vous avez perdu » afin de pallier l'absence de score sur l'afficheur.
- Au lancement du jeu, nous avons affiché deux informations : la version du jeu et la durée du timer à chaque appel de fonctions afin de pouvoir s'en rappeler en cas de modifications.
- En raison de son programme, l'IA[2] ne peut pas perdre

2 Intelligence Artificielle

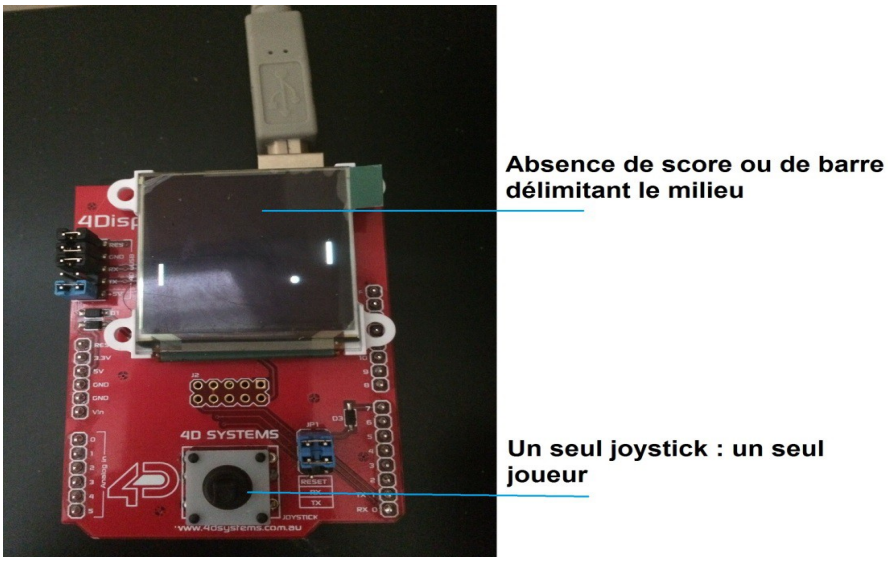


Illustration 7: Interface de Pong réalisé

C'est donc en voulant respecter l'œuvre originel tout en prenant quelques libertés sur celle-ci que nous nous sommes lancés dans la programmation du jeu.

3. La partie programmation

3.1. bibliothèques

Afin d'utiliser notre shield correctement, nous devons utiliser une bibliothèque. La bibliothèque aura pour but de programmer plusieurs fonctions qui seront utilisées par le programme dans un fichier extérieur au fichier de programmation. L'intérêt de faire une bibliothèque est la possibilité de le réimplanter dans un autre fichier de programmation via la commande #include « monfichier.h » où « monfichier.h » est la bibliothèque et donc ne pas devoir tout réécrire dans le nouveau projet.

On voit que l'extension de la bibliothèque est .h qui signifie entête (header).

Dans notre programme, nous utiliserons 4 bibliothèques :

1. « Goldelox_Serial_4DLib.h »
2. « Goldelox_const4d.h »
3. « SoftwareSerial.h »
4. « oled128.h »

Les fichiers Goldelox sont des fichiers liés au processeur GOLDELOX utilisé par notre shield, la bibliothèque SofwareSerial est la bibliothèque servant au port de communication série et la dernière oled128 est la bibliothèque que nous utiliserons pour utiliser notre écran.

La bibliothèque peut être construite sous ces formes :

C'est-à-dire qui ne contient aucune fonction mais déclaration de variable avec des valeurs

```
#define F_charheight 1
#define F_charwidth 2
#define F_gfx_BGcolour -146
#define F_gfx_ChangeColour -66
#define F_gfx_Circle -51
#define F_gfx_CircleFilled -52
#define F_gfx_Clipping -148
#define F_gfx_ClipWindow -65
#define F_gfx_Cls -41
#define F_gfx_Contrast -154
#define F_gfx_FrameDelay -151
#define F_gfx_GetPixel -54
#define F_gfx_Line -46
#define F_gfx_LinePattern -155
#define F_gfx_LineTo -44
#define F_gfx_MoveTo -42
#define F_gfx_Orbit 3
#define F_gfx_OutlineColour -153
#define F_gfx_Polygon 4
#define F_gfx_Polyline 5
#define F_gfx_PutPixel -53
#define F_gfx_Rectangle -49
#define F_gfx_RectangleFilled -50
#define F_gfx_ScreenMode -152
#define F_gfx_Set -40
#define F_gfx_SetClipRegion -68
#define F_gfx_Transparency -150
#define F_gfx_TransparentColour -149
```

associées.

```
void blitComtoDisplay(word  X, word  Y, word  Width, word  Height, t4DByteArray  Pixels) ;
void gfx_BGcolour(word  Color) ;
void gfx_ChangeColour(word  OldColor, word  NewColor) ;
void gfx_Circle(word  X, word  Y, word  Radius, word  Color) ;
void gfx_CircleFilled(word  X, word  Y, word  Radius, word  Color) ;
void gfx_Clipping(word  OnOff) ;
void gfx_ClipWindow(word  X1, word  Y1, word  X2, word  Y2) ;
void gfx_Cls(void) ;
void gfx_Contrast(word  Contrast) ;
void gfx_FrameDelay(word  Msec) ;
void gfx_Line(word  X1, word  Y1, word  X2, word  Y2, word  Color) ;
void gfx_LinePattern(word  Pattern) ;
void gfx_LineTo(word  X, word  Y) ;
void gfx_MoveTo(word  X, word  Y) ;
void gfx_OutlineColour(word  Color) ;
void gfx_Polygon(word  n, t4DWordArray  Xvalues, t4DWordArray  Yvalues, word  Color) ;
void gfx_Polyline(word  n, t4DWordArray  Xvalues, t4DWordArray  Yvalues, word  Color) ;
void gfx_PutPixel(word  X, word  Y, word  Color) ;
void gfx_Rectangle(word  X1, word  Y1, word  X2, word  Y2, word  Color) ;
void gfx_RectangleFilled(word  X1, word  Y1, word  X2, word  Y2, word  Color) ;
void gfx_ScreenMode(word  ScreenMode) ;
void gfx_Set(word  Func, word  Value) ;
void gfx_Transparency(word  OnOff) ;
void gfx_TransparentColour(word  Color) ;
void gfx_Triangle(word  X1, word  Y1, word  X2, word  Y2, word  X3, word  Y3, word  Color) ;
void media_Image(word  X, word  Y) ;
void media_SetAdd(word  Hiword, word  Loword) ;
void media_SetSector(word  Hiword, word  Loword) ;
void media_Video(word  X, word  Y) ;
void media_VideoFrame(word  X, word  Y, word  Framenumber) ;
```

Qui ne contient que des fonctions avec des paramètres.

Les 3 premières bibliothèques que nous utilisons (Goldelox_Serial_4DLib.h, Goldelox_const4d.h, SoftwareSerial.h) sont des bibliothèques données par les constructeurs pour pouvoir utiliser leurs équipements. GOLDELOX lié au processeur GOLDELOX utilisé par le shield provenant la société 4D Systems et SoftwareSerial provenant d'Arduino.

La dernière bibliothèque a été créée par Oscar GONZALEZ que nous avons repris et modifié certains points pour notre projet. Vous trouverez notre bibliothèque en annexe.

3.2. Le programme

3.2.1. Bibliothèques

Nous allons passer à la partie programme de notre projet pour commencer, nous avons déclaré les bibliothèques comme expliqué précédemment :

```
#include "Goldelox_Serial_4DLib.h"
#include "Goldelox_const4d.h"
#include "SoftwareSerial.h"
#include "oled128.h"
```

3.2.2. Variables

3.2.2.1. Définitions

Les variables sont utilisées pour stocker des données, qui seront utilisées par le programme. Il existe plusieurs types de variables que nous utilisons dans notre programme. Par souci de lisibilité, nous allons définir les types de variables susceptibles d'être utilisé.

La variable type :

Type de donnée	**Signification**	**Taille (en octets)**	**Plage de donné**
Int	Entier	2 sur processeur 32bit 4 sur processeur 64bit	-32 768 à 32767 -2 147 483 64 8 à 2 147 483 647
Unsigned int	Entier non signé	2 sur processeur 32bit 4 sur processeur 64bit	0 à 65535 0 à 4 294 967 295
bool	Booléen	1 ou 2 dépend du compilateur	0 ou 1 True or false

On voit qu'il y a plusieurs types de variable que l'on peut utiliser en fonction de nos besoins et/ou de la capacité de la machine. En effet si on est limité par notre machine pour l'utilisation d'une variable qui variera entre 0 et 1 on préférera l'utilisation de la variable bool.

3.2.2.2. Variables fixes

Ensuite nous avons déclaré toutes les variables que nous utiliserons certains fixe comme les paramètres comme les couleurs ou bien la hauteur des barres.

```
// Color
#define WHITE 65535
#define BLACK 0
//POSITION

#define H 18 // Hauteur des barres
```

Les couleurs étant codées sur 16 bits on obtient donc un total de $2^{16}=65536$ couleurs, le blanc sera la dernière couleur à la valeur 65535 et le noir la première couleur à la valeur 0.

Pour déclarer les variables en variables fixe, on utilisera une directive #define qui associera la macro à la définition.

Ici on associe WHITE (le nom de la macro) à la valeur 65535 qui est la définition.

3.2.2.3. Variables globales

Nous utiliserons aussi des variables globales, il faut savoir qu'une variable non globale sera détruite lorsqu'elle sort de sa fonction (sauf exception). L'utilisation des variables globales permet que la variable ne soit pas détruite après l'exécution des fonctions.

Les variables globales que nous utilisons sont :

```
bool i = 0;
unsigned int X = 0;
unsigned int Y = 0;
unsigned int IBGY = 0;
unsigned int IBDY = 0 ;
unsigned int NBY;
int BG_Y;
int BG_Black;
int BD_Black;
int BD_Y;
bool flag = 1;
bool flag2 = 1;
bool start = 1;
bool PERDU = false;

int VitX = 1;
int VitY = 1;

unsigned int BALL_X_INIT = 80;
unsigned int BALL_Y_INIT = 71;
unsigned int BG_Y_INIT = 59;
unsigned int BD_Y_INIT = 59;
```

Nous allons expliquer plus en détails, les variables que nous avons déclarées en variables globales.

3.2.2.4. Variables statiques

Une variable statique est une variable locale, c'est-à-dire qu'elle est ne peut être utilisée que dans la fonction où elle est déclarée et devrait être détruite à la fin de l'exécution de la fonction, mais comme on lui met le type STATIC qui aura pour effet de garder la valeur de la variable même lorsque l'on quitte la fonction.

3.2.3. Les fonctions

Pour utiliser des fonctions, on doit avant tout les déclarer comme ceci :
type nomdelafonction(type paramètre) ;

```
int DB();
void BG(int NB);  //fonction barre gauche
void BD(int NB);  //fonction barre droite
```

Les fonctions déclarées sont :

- BD(), utilisé pour le déplacement de la balle.
- BG(int NB), utilisé pour la barre gauche.
- BD(int NB), utilisé pour la barre droite.

Ici les types utilisés sont void qui signifie que la fonction ne retourne aucune valeur, la fonction s'exécutera sans que la fonction retourne une valeur ou on peut leur mettre le type int comme la fonction DB() qui retournera une valeur dans la plage de donnée du int.

Les fonctions BG et BD respectivement les fonctions pour la barre gauche et pour la barre droite, on voit que ces fonctions ont ce qu'on appelle un paramètre, c'est une variable externe à la fonction qui sera utilisée dans la fonction.

Exemple :

On déclare la variable suivante Int x et la fonction Void Mafonction(int i). La fonction Mafonction(int i) retourne rien car il a comme type void.

On peut utiliser l'exemple suivant :

```
X = 5 ;  // La valeur X vaut 5
Mafonction(X) ; // La fonction Mafonction utilisera la variable X
```

On peut donc utiliser ce que contient la variable X (ici 5) via la variable i, donc lorsqu'on utilise la variable i dans la fonction Mafonction.

```
Void Mafonction(int i)
{
        Int y ;
        y = i ; // i vaut la valeur X (5) et on met la valeur i dans y donc la valeur de y vaut 5
}
```

3.2.3.1. La fonction Setup

Dans Arduino on doit utiliser une fonction setup qui est automatiquement déclarée et qui ne s'exécutera qu'une seule fois à l'exécution de l'Arduino.

```
void setup()
{
  // Mise à zéro de l'écran (Reset Routine optionnel)
  pinMode (7, OUTPUT);
  digitalWrite(7,1);
  delay(100);
  digitalWrite(7,0);
  // Fin de routine
  delay(5000);
  //SoftwareSerial DisplaySerial(1,0);
  //pinMode(0,1);
  Serial.begin(OLED_BAUDRATE);
  OLED_Init();
  ConfigJoystick();
  OLED_Clear();
  delay(5000);
}
```

Grâce à la fonction pinMode qui est utilisé de la manière suivante : pinMode(pin, mode).

Dans notre programme, on met le pin 7 en sortie.

Le digitalWrite() est déclaré de la manière suivante : digitalWrite(pin, value).

Dans notre programme, on va faire une mise à zéro de l'écran en déclarant le pin 7 qui est relié de l'écran en sortie grâce à la fonction pinMode(pin,mode). Puis on va mettre à l'écran à 1 puis à 0 grâce à la fonction digitalWrite(pin, value).

On va déclarer la vitesse de transmission à 19200 bauds qu'on a définis dans la bibliothèque via la fonction Serial.begin(Vitesse).

On va initialiser l'écran grâce à la fonction OLED_Init() qui est déclarée dans la librairie, puis nous allons configurer le fonction ConfigJoystick() puis nous effaçons l'écran. Nous avons mis des delay (des pauses) de 5 secondes car il est préconisé par le constructeur.

3.2.3.2. La fonction loop

La fonction loop() est une fonction qui sera exécutée en boucle :

```
void loop()
{
  if(i==0)
  {
    OLED_Clear();
    OLED_DrawStringText(15,0,1,WHITE,1,1,"Jeu PONG");  // Affiche la texte "Jeu PONG"
    OLED_DrawStringText(15,20,1,WHITE,1,1,"Update:30/03");
    OLED_DrawStringText(25,30,1,WHITE,1,1,"Delay = 20ms");   // Affiche le delay = 20ms
    delay(5000);
    OLED_Clear();
    i =1 ;
  }
  //DB();
  BG(NBY); // Met en paramètre le tableau
  BD(NBY); // // Met en paramètre le tableau
  NBY = DB();  //NBY est la position de la balle en y

}
```

Dans cette fonction on va coder l'intégralité de notre programme, dans un premier temps nous afficherons un message pendant 5 secondes.

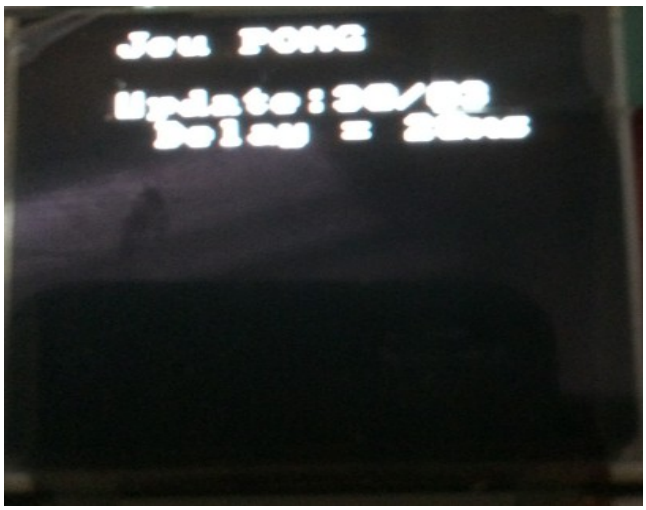


Illustration 8: Interface de démarrage personnalisé

Puis le programme va exécuter les fonctions BG et BD avec comme paramètre NBY. La variable NBY va prendre comme valeur la valeur de la fonction DB().

3.2.3.3. La fonction DB()

La fonction DB() est la fonction utilisée pour gérer le déplacement de la balle. Vous trouverez le programme complet en annexe.

Tout d'abord, on va déclarer des variables en static car celles-ci ne seront utilisées que dans la fonction.

```
static int BALL_X;
static int BALL_Y;
static int Ball_X_Black;
static int Ball_Y_Black;
```

Ici BALL_X défini la position de la balle en X et BALL_Y défini la position de la balle en Y, les variables Ball_X_Black et Ball_Y_Black sont les positions en X et Y pour écrire une balle en noir. On utilise cette technique pour effacer la balle.

Ensuite nous avons écrit l'équation pour le déplacement de la balle.

```
// Equation pour le déplacement de la balle en X et Y
BALL_Y = BALL_Y_INIT + (Y * VitY);
BALL_X = BALL_X_INIT + (X * VitX);
```

Pour dessiner la balle nous avons codé ça de cette manière :

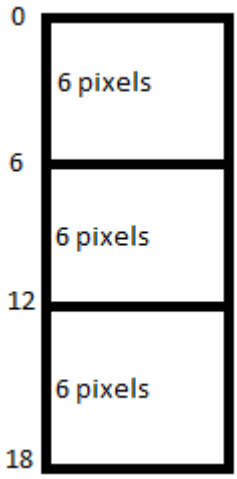
```
 // Si la balle est compris dans l'aire de jeux alors on affiche la balle
 if((BALL_X < 122) && (BALL_X >2)&&(PERDU == false))
{
  X= X + 4;
  Y= Y+4;
  OLED_DrawCircle(Ball_X_Black,Ball_Y_Black,2,1,BLACK);
  OLED_DrawCircle(BALL_X,BALL_Y,2,1,WHITE);
  Ball_X_Black = BALL_X;
  Ball_Y_Black = BALL_Y;
}
```

Si la balle est comprise dans l'aire de jeux latéral et que l'on n'a pas perdu. Alors on incrémente la valeur de la position en X et Y de 4 puis on met une balle de couleur noir et on affiche la balle en blanc à la nouvelle position en X et Y.

Dans cette partie du programme, on a codé le changement de sens lorsque la balle touche arrivera vers la position en X inférieur ou égal à 6 qui correspond à la position de la barre gauche.

```
// Si la position de la balle en X  est inférieur à 6 alors on inverse le sens de la balle en X
if (BALL_X <=6)
{
  if(flag2==0)
  {
  VitX = VitX*(-1);
  X=0;
  BALL_X_INIT = 6;
  flag2=1;
  }
}
```

Si la position de la balle est supérieure ou égal à 119, cette position correspond à la position de la barre à droite. On décidera donc de 3 conditions, qui détermineront la vitesse de la balle lorsqu'elle rebondit à certain endroit de la barre droite. Les barres de jeux sont d'une taille de 18 pixels, on va donc diviser la barre en 3 parties qui constituent nos 3 conditions comme ceci :



La première condition :

```
if(BALL_X >=119)
{
  if(flag2==1)
  {
        // Si ça tape la partie supérieux de la barre on multiplie la vitesse par 2
        if((BALL_Y >= BD_Y)&&(BALL_Y<BD_Y+6))
        {
         VitX = -1;
         VitY = 2;
         X = 0;
         BALL_X_INIT = 119;
         flag2 = 0;
        }
```

Cette condition décrit que lorsque la balle rebondit sur la partie supérieure de la barre alors la vitesse en Y est multipliée par 2 et le sens de la balle est inversé.

La seconde condition :

```
// Si ça tape la partie du milieu alors soit on garde la vitesse initiale

if((BALL_Y > BD_Y+6)&&(BALL_Y<=BD_Y+12))
{
  if(VitY == 2)
  {
     VitX = -1;
     VitY = 2;
     X = 0;
  }
  else
  {
     VitX = -1;
     VitY = 1;
     X = 0;
  }
 BALL_X_INIT = 119;
 flag2 = 0;
}
```

Si la balle rebondit sur la partie du milieu, alors elle garde la vitesse initiale avec laquelle elle rebondit.

La troisième condition est identique à la première condition.

Nous avons conditionné pour faire perdre le joueur, en effet lorsque la balle ne rebondit pas sur la barre droite alors on affiche le message « Vous Avez PERDU » et on met la variable PERDU à 1 (true) pour stopper la réécriture de la balle.

```
if(((BALL_Y < BD_Y-1) || ( BALL_Y > BD_Y+18)) && (BALL_X >=119))
{
   OLED_DrawStringText(15,0,1,WHITE,1,1,"Vous Avez PERDU");
   PERDU = true;
}
```

Et enfin nous avons codé le rebond avec les côtés haut et bas de l'écran.

```
// Conditionne le rebond sur le haut et le bas

if(BALL_Y <=4)
{
    if(flag == 0)
    {
         VitY = VitY*(-1);
         Y = 1;
         BALL_Y_INIT = 6;
         flag = 1;
    }
}
if(BALL_Y >= 120)
{
    if(flag == 1)
    {
     VitY = VitY*(-1);
     Y = 0;
     BALL_Y_INIT = 120;
     flag = 0;
    }
}
```

Une fois la fonction exécutée, nous retournons la valeur de la position de la balle en Y.

```
return BALL_Y; // Retourne le tableau
```

3.2.3.4. La fonction BG(int NB)

Cette partie du programme est incomplète car nous n'avons pas réussi à rendre l'IA (intelligence artificielle) assez forte pour ne pas perdre très rapidement. La fonction BG(int NB) sert à la création de la barre gauche.

La barre gauche est codée de manière à ce qu'elle suive le déplacement de la balle.

```
void BG(int NB)
{
  //static int BG_Y;
 // static int BG_Black;


  OLED_DrawRectangle(0,BG_Black,2,H,1,BLACK);
  OLED_DrawRectangle(0,BG_Y,2,H,1,WHITE);
  BG_Black = BG_Y;
  BG_Y = BG_Y_INIT + IBGY;
  if(BG_Y <= 1)
  {
    BG_Y = 1;
  }
  if(BG_Y >= 112)
  {
    BG_Y = 112;
  }

  if(((BG_Y >= 0) && (BG_Y <= 112)) == 1)
  {
    if(NB > (BG_Y+(H/2)) && (VitY>0)==1)
    {
      IBGY = IBGY + 8;
    }
    if(NB > (BG_Y+(H/2)) && (VitY < 0)==1)
    {
      IBGY = IBGY;
    }
  }

  if(((BG_Y >=0) && (BG_Y <= 112)) == 1)
  {
    if((NB < (BG_Y+(H/2)) && (VitY < 0))==1)
    {
      IBGY = IBGY - 8;
    }
    if(NB > (BG_Y+(H/2)) && (VitY > 0)==1)
    {
      IBGY = IBGY;
    }
  }

}
```

3.2.3.5. La fonction BD()

Cette fonction a été créée pour la barre droite qui est la barre du joueur. Elle est commandée par le joystick. Nous allons utiliser la fonction digitalRead() qui est utilisée pour lire la valeur de l'entrée analogique si celle-ci est appuyée alors elle renverra la valeur 0 donc lorsque le joueur presse le joystick vers le bas alors la barre devra s'incrémenter (se déplacer vers le bas) ou lorsque le joueur presse le joystick vers le haut alors la barre devra se décrémenter (se déplacer vers le haut).

```
void BD(int NB)
{
  if(start ==1)
  {
    start = 0;
    OLED_Clear();
    BD_Y = BD_Y_INIT;
    OLED_DrawRectangle(125,BD_Y_INIT,2,H,1,WHITE);
  }
  //OLED_DrawRectangle(125,BD_Black,2,H,1,BLACK);
  OLED_DrawRectangle(125,BD_Y,2,H,1,WHITE);
  BD_Black = BD_Y;
  if(BD_Y <= 0)
  {
    BD_Y = 0;
    BD_Y_INIT = 0;
  }
  if(BD_Y >= 110)
  {
    BD_Y = 110;
    BD_Y_INIT = 110;
  }

  if(digitalRead(JOYSTICK_UP) == 0)
  {
    OLED_DrawRectangle(125,BD_Black,2,H,1,BLACK);
    BD_Y = BD_Y_INIT -8;
    BD_Y_INIT = BD_Y;

  }
  if(digitalRead(JOYSTICK_DOWN) == 0)
  {
    OLED_DrawRectangle(125,BD_Black,2,H,1,BLACK);
    BD_Y = BD_Y_INIT +8;
    BD_Y_INIT = BD_Y;
  }
}
```

Conclusion

Au cours du projet d'Études et Réalisations du semestre 4, nous avons réalisé un jeu PONG sur une carte arduino en utilisant uniquement les connaissances acquises à l'IUT. Nous voulions créer un jeu qui se voulait au départ simple à concevoir mais qui s'est révélé ardu. En effet, PONG étant très connu il nous aurait été facile de copier le programme sur internet ; mais nous avons préféré créer le jeu par nos propres mains sans jamais utiliser le travail d'un autre même si cela s'est avéré plus difficile. Au final non seulement nous avons réussi à récréer le jeu, mais nous avons aussi pu y inclure le mode 1 joueur que nous voulions, en utilisant le joystick du shield arduino. Notre projet a donc abouti. Néanmoins avec quelques semaines de plus, il aurait été intéressant de créer un mode 2 joueurs en créant nous-même une autre carte avec 2 joysticks.

Bibliographie

Index des illustrations