Une Borne d'ARCADE *MAME*

« Qu'est-ce que le *MAME Cabinet* » ?

On connait déjà le principe de l'émulateur de jeux vidéo, un logiciel capable de simuler le fonctionnement d'une machine, afin de faire tourner des copies de jeux.
Et bien MAME, pour « Multiple Arcade Machine Emulator » est un émulateur qui à la particularité, d'émuler plusieurs types de machines d'arcade, simplifiant alors grandement les choses…

Liste des composants pour l'élaboration de la borne :

La partie informatique :

- 1 carte Mère,
- 1 Processeur,
- 1 peu de mémoire RAM,
- 1 carte Graphique,
- 1 carte Son,
- 1 Disque Dur,
- 1 Alimentation,
- 1 Système d'exploitation configuré pour fonctionner sur un Télé.
- 1 Émulateur.

La partie Mécanique :

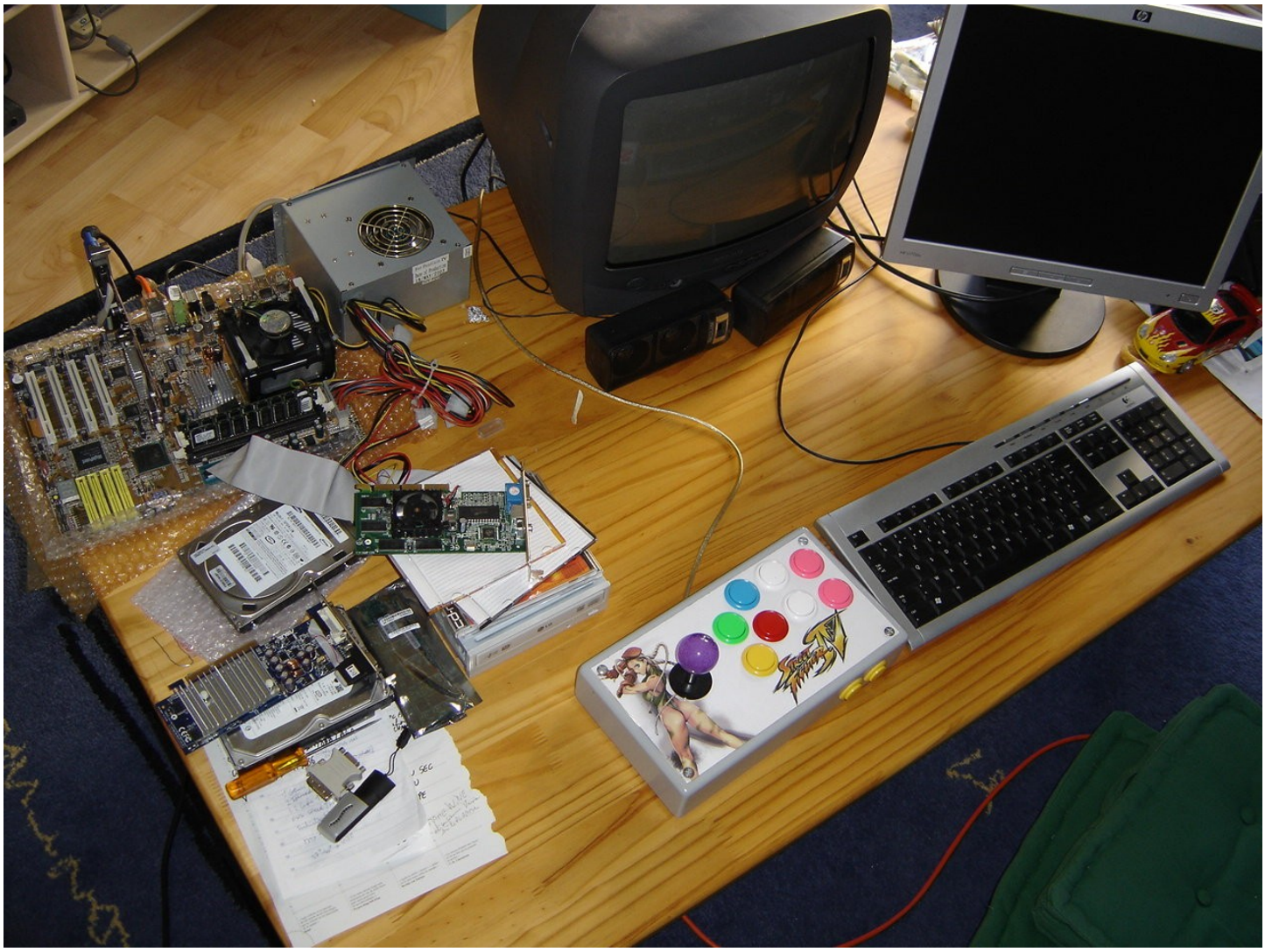
- 1 Meuble, à restaurer ou à fabriquer en bois.
- 1 Téléviseur cathodique assez gros.
- 2 Joystick USB.

Sommaire

ÉTUDES

J'ai commencé par rassembler le matériel dont je dispose pour faire un banc test et lancer des expérimentations.
Mais je ne vais pas les détailler ici, histoire d'éviter d'embrouiller les esprits.
C'est dommage, car c'est dans ces moments là qu'on apprend le plus… Je vais donc m'efforcer de faire ressortir les éléments importants.

Pour la durée des tests, la carte mère est simplement posée sur la table avec un disque dur, un lecteur CD-Rom, un écran VGA et un écran Télé.
Un câble VGA muni d'un adaptateur bidouillé permet de relier la carte vidéo au télé.



Grâce au wiki très bien fait traitant des Front-Ends MAME [http://wiki.arcadecontrols.com/wiki/Front-Ends] (couche logicielle pour accroitre les capacités techniques de *MAME [http://mamedev.org/]*) le choix de l'émulateur s'est rapidement précisé pour *AdvanceMAME* !

[http://advancemame.sourceforge.net/]

> En effet, c'est le seul qui soit capable de gérer et basculer automatiquement l'affichage des jeux dans leurs résolutions natives sur un écran TV, via la sortie VGA d'une carte Vidéo standard [http://advancemame.sourceforge.net/doc-cardwin.html]et d'être utilisé sans clavier ni souris.

(À savoir qu'il existe une carte *ArcadeVGA* [http://wiki.arcadecontrols.com/wiki/ArcadeVGA] au bios bidouillé pour fonctionner directement sur un Moniteur Arcade)

Émulateur et carte graphique ATI 9200

[validée sur Ubuntu 10.04LTS]

Introduction :

On va s'intéresser brièvement aux différences entre Téléviseur et Moniteur informatique cathodique. [https://secure.wikimedia.org/wikipedia/fr/wiki/Tube_cathodique]

- Un téléviseur PAL affiche 25 images par secondes en 625 lignes horizontales, balayées à 15,625kHz par un point lumineux en deux temps : les lignes impaires puis les lignes paires, donc 50 demi-images qu'on dit entrelacées à 50Hz.

La résolution réelle maximale de l'image atteint 720 points de large × 576 lignes de haut. (noté 576i)

- L'écran informatique affiche des images progressives (images complètes, pas d'entrelacement), fabriquées par une synchro verticale + une horizontale à des résolutions et des fréquences très élevées, permettant de travailler agréablement à faible distance de l'écran.

Par exemple mon tube 17 pouces IIyama A705MT indique comme caractéristiques une plage de Synchro Horizontale de 30 à 86kHz et Verticale de 50 à 180Hz permettant d'afficher par exemple 1024 points x 768 points @85Hz.
L'écran informatique est donc désigné 30kHz.

Ce qu'il fallait démontrer, c'est que la carte graphique de l'ordinateur est conçue pour sortir ces 30kHz, ce qui la rend en l'état, incompatible avec les téléviseurs 15kHz.
Battu ? On abandonne ?
Meuhh non, Xorg est là, paramétrable à souhait !

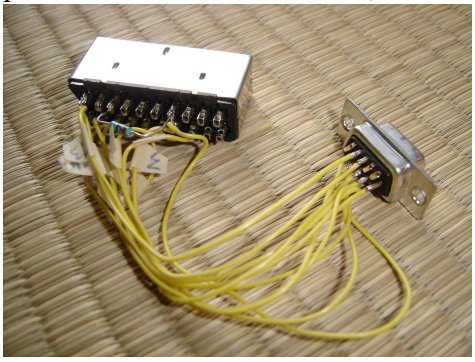
1 - L'adaptateur Péritel/VGA :

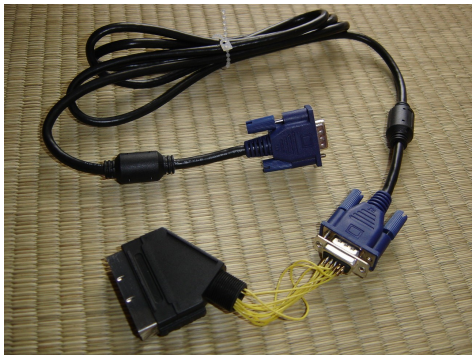
Moyennant une adaptation, on va exploiter les signaux **R**ouge, **V**ert, **B**leu et les Synchro sortants de la prise VGA [https://secure.wikimedia.org/wikipedia/fr/wiki/Connecteur_VGA] de la carte vidéo à destination de l'entrée péritel [http://fr.wikipedia.org/wiki/Péritel] du TV, Xorg s'occupant de balancer cela en 15kHz.
L'adaptateur doit faire correspondre ces N° de pattes :

```
        Prise VGA                 —        Prise Péritel

Pin N°1: Vidéo Rouge              —     Pin N°15: Rouge
Pin N°2: Vidéo Vert               —     Pin N°11: Vert
Pin N°3: Vidéo Bleu               —     Pin N°7:  Bleu
Pin N°6: Retour rouge             —     Pin N°13: Masse Rouge
Pin N°7: Retour vert              —     Pin N°9:  Masse Verte
Pin N°8: Retour bleu              —     Pin N°5:  Masse Bleue
Pin N°9: +5 V                     —     Pin N°8:  SWITCH
Pin N°10: Masse                   —     Pin N°17: Masse Synchro
Pin N°13: HSync et N°14: VSync —     Pin N°20: Synchro.
```

Ajouter une résistance de 75 Ohms entre la Pin 8 et 16 de la prise péritel, permet, grâce au 5V en provenance de la carte vidéo, de basculer le TV en RVB sur l'entrée AV.





2 - Paramétrage d'Xorg pour le TV :

- Sur Ubuntu 10.04LTS, la carte vidéo ATI Radeon 9200 est automatiquement exploitée par un pilote libre, donc rien à installer.

Cela fait aussi quelques temps que le fichier de configuration */etc/X11/xorg.conf* n'est plus utilisé pour la configuration du pilote par le système, mais il peut toujours être utilisé par nous autres humains, pour amender la configuration par défaut et ainsi adapter l'affichage à nos besoins.

Ainsi on va devoir y spécifier les sections élémentaires que sont *Section "Device"*, *Section "Monitor"*, et *Section "Screen"*, avec des modificateurs pour l'utilisation du Téléviseur :

```
### spécifie un nom de carte vidéo (à définir) et le nom du pilote.
Section "Device"
    Identifier  "ATI"
    Driver      "radeon"
EndSection

### spécifie le nom de l'appareil (à définir), les fréquences de balayage et de
synchro du TV, ainsi que le modeline 15kHz.
Section "Monitor"
        Identifier   "TV"
## Idéalement ces valeurs  :
#    HorizSync    15.625
#    VertRefresh  50.0
## mais on peut mettre une plage plus souple pour s'adapter au modèle de TV dont
on dispose :
        HorizSync    15.0 - 20.0
        VertRefresh  50.0 - 60.0
## le modeline TV @15kHz
    # 720x576x50.00 @ 15.625kHz
    Modeline "720x288x50.08"  13.875000  720 744 808 888  288 293 296 312
-HSync -VSync
EndSection

### Rappel des noms précédents et du mode dans lequel on veut fonctionner.
Section "Screen"
    Identifier "Default Screen"
    Device     "ATI"
    Monitor    "TV"
    DefaultDepth     24
    SubSection "Display"
    Depth     24
    Modes    "720x288x50.08"
    EndSubSection
EndSection
```

Mais attention, DANGER !!

- Chose importante également, une fois ce fichier *xorg.conf* mis en place IL NE FAUT PAS allumer le TV durant le démarrage de l'ordinateur, ni tenter de passer en console tty.

Et oui, la carte vidéo continue d'envoyer du 30kHz en permanence, et donc durant le boot de l'ordi ou en tty… Ce n'est qu'une fois qu'X est lancé que du 15kHz est envoyé.

- Et vice-versa, quand X est lancé, il ne faut pas brancher de moniteur informatique.

Vu comme ça, ça complique un peu les tests, mais Ubuntu est une chose bien faite et on va pouvoir modifier la configuration d'xorg grâce à un second ordinateur via ssh ou en partage de bureau à distance.

3 - Installation d'*AdvanceMAME* :

Sur la page de téléchargement [http://advancemame.sourceforge.net/download.html] , prendre la dernière version des sources d'*AdvanceMAME* en date (11/01/2009)
- Extraire les sources et se rendre à leurs emplacements :

```
cd /home/makoto/Download/AdvanceMame/advancemame-0.106.1
```

Le fichier BUILD nous indique de quoi on aura besoin, à savoir les compilateur suivants :

- GNU gcc C/C++ 3.2.3 or 3.3.4 (or newer)
- GNU make 3.79.1 (or newer)

Et les sources des logiciels suivant :

- NASM 0.98.33 (or newer) **[edit :** suite à des pb de son et d'affichage, ne pas mettre ce composant**]**
- SVGALIB 1.9.14 (or newer)
- SDL 1.2.4 (or newer)
- S-Lang 1.4.3 (or newer)
- ncurses 5.4 (or newer)
- FreeType 2.1.7 (or newer)
- zlib 1.1.4 (or newer)
- expat 1.95.6 (or newer)

J'ai donc ajouté *libsdl1.2-dev*, *libslang2-dev*, *libncurses5-dev*, *libfreetype6-dev*, *zlib1g-dev* et *libexpat1-dev*.

```
sudo apt-get install libsdl1.2-dev libslang2-dev libncurses5-dev libfreetype6-dev
zlib1g-dev libexpat1-dev
```

À noter que la version de *libsvga1-dev*, est incompatible avec *AdvanceMAME*, mais on s'en passera.
S'il manque certains éléments, des fonctionnalités d'*AdvanceMAME* ne fonctionneront pas; ici, l'affichage des jeux par le driver SVGA ne fonctionnera pas, mais heureusement SDL prendra la relève.

Quand tout est prêt on peut lancer :

```
./configure
```

À la fin, un résumé des opérations est affiché, indiquant si toutes les dépendances indiquées par BUILD sont satisfaites. Si tel n'est pas le cas, il est temps d'y remédier en cherchant et ajoutant via synaptic, les versions DEV, par exemple *libsdl1.2-dev* si *sdl* n'est pas indiqué en face de *Video*, *Sound*, *Keyboard*, *Joystick* et *Mouse*.
puis relancer ./configure jusqu'à obtenir quelque chose de quasi parfait, comme ça :

```
== Host ==
Environment : i686-pc-linux-gnu
Endianess : little
Bits : 32
Assembler for Pentium : no
Assembler MIPS3 emulator for Pentium : no

== Drivers/Libraries ==
Video : fb slang ncurses sdl
Sound : alsa oss sdl
Keyboard : sdl raw event
Joystick : sdl raw event
Mouse : sdl raw event
Misc : zlib expat pthread freetype2

== Compiler ==
CC : gcc
CXX : g++
CFLAGS : -march=native -O2 -fomit-frame-pointer -fno-merge-constants -Wall -Wno-sign-compare -Wno-unused
LDFLAGS : -s

== Configuration ==
Emulator : mame
Debugger : no
```

Bien, il ne devrait plus y avoir d'obstacle au bon déroulement des prochaines opérations, on peut donc fabriquer le binaire :

```
make
```

Puis l'installer :

```
sudo make install
```

4 - Installation d'AdvanceMENU :

- Procéder de la même manière pour *AdvanceMENU* [http://advancemame.sourceforge.net/menu-download.html]

Une fois les deux logiciels installés, on retrouve les binaires dans */usr/local/bin* :

```
advmame - The AdvanceMAME emulator.
advmenu
advcfg - The automatic video configurator.
advv - The manual video configurator and tester.
advm - The mouse tester.
advk - The keyboard tester.
advs - The sound tester.
advj - The joystick tester.
```

5 - Il est temps de jouer !!

Au premier lancement de l'émulateur,

```
advmame
```

Le fichier de configuration est crée

> Creating a standard configuration file...
> Configuration file `/home/makoto/.advance/advmame.rc' created with all the default options.
>
> The default rom search path is `/home/makoto/.advance/rom:/usr/local/share/advance/rom'.
> You can change it using the `dir_rom' option in the configuration file.

Idem, au premier lancement du menu,

```
advmenu
```

Le fichier de configuration est crée

> Creating a standard configuration file...
> Adding emulator `advmame'...
> Configuration file `/home/makoto/.advance/advmenu.rc' created with all the default options.

Ce sont ces fichiers *advmame.rc* et *advmenu.rc* qui vont être édités manuellement pour peaufiner les réglages.
Tout est documenté dans de dossier */usr/local/share/doc/advance* ou depuis les sources */home/makoto/Download/AdvanceMame/advancemame-0.106.1/doc* ou encore, sur le site du projet *AdvanceMAME*.

- *advmame.rc*

  device_sound sdl [car je n'avais pas de son dans les jeux]
  device_video auto [en fait c'est *sdl* qui est utilisé, donc on pourrait le forcer]
  display_resize no [pas besoin d'étirer l'image, puisqu'ils sont lancés dans leurs résolutions natives]
  display_resizeeffect no [idem]
  display_rgbeffect none [on veut se rapprocher du rendu Arcade, donc besoin de rien]

- *advmenu.rc*

  device_sound sdl [car advmenu freezait sans raison]
  sound_backgroud_loop none [pour virer le mp3 de feu qui crépite tiré de DukeNukem3D]
  device_joystick auto [pour se passer du clavier lors de la sélection et lancement des jeux]
  device_video_output fullscreen [pour lancer advmenu en plein écran]

Bref, pour jouer il suffit de copier des jeux dans le dossier */home/makoto/.advance/rom*, puis de taper :

```
advmame NomDuJeux
```

Ou alors, pour accéder au menu graphique, suffit de lancer la commande :

```
advmenu
```

Ou de créer un lanceur sur le bureau.

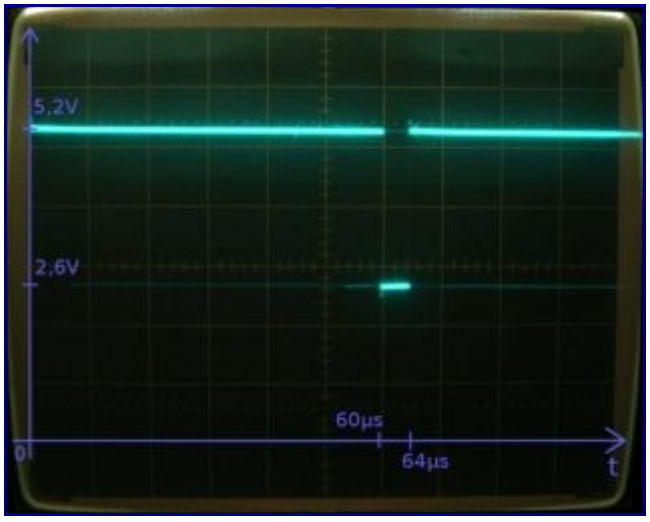
Dispositif de protection de l'écran TV durant le démarrage et l'arrêt de l'ordinateur

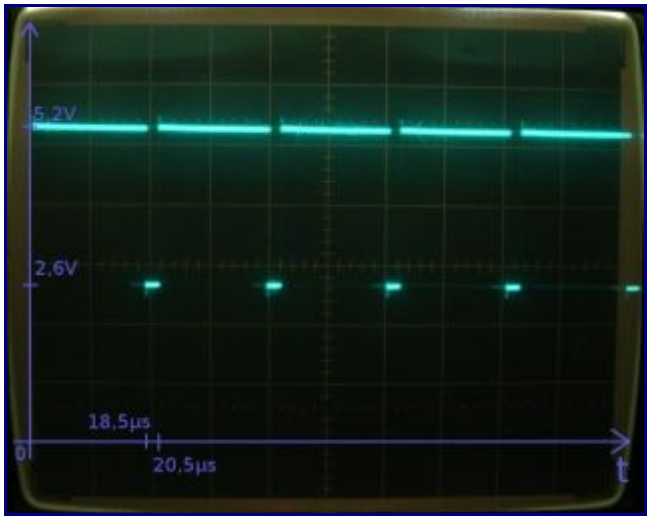
Introduction :

Petit rappel du pourquoi cette nécessité :

Une fois l'ordinateur démarré et les logiciels lancés, l'écran de TV reçoit un signal de synchronisation @15,625kHz, car Xorg utilise un modeline réglé pour cela.
Tant qu'X n'est pas lancé, c'est à dire durant les phases de boot et halt de l'ordinateur, le TV va recevoir un signal de fréquence au moins deux fois plus élevé de la part de la carte vidéo !!
Cette situation pouvant à terme se révéler destructive pour le tube cathodique, il est impératif de protéger l'écranTV

Commençons par regarder la tronche du signal de synchro envoyé au télé;
Il s'agit donc des signaux HSync et VSync de la carte vidéo reliés ensembles sur la Synchro du télé.
J'ai donc branché mon oscilloscope sur le connecteur VGA de la carte graphique, pour mesurer ce signal sortant…

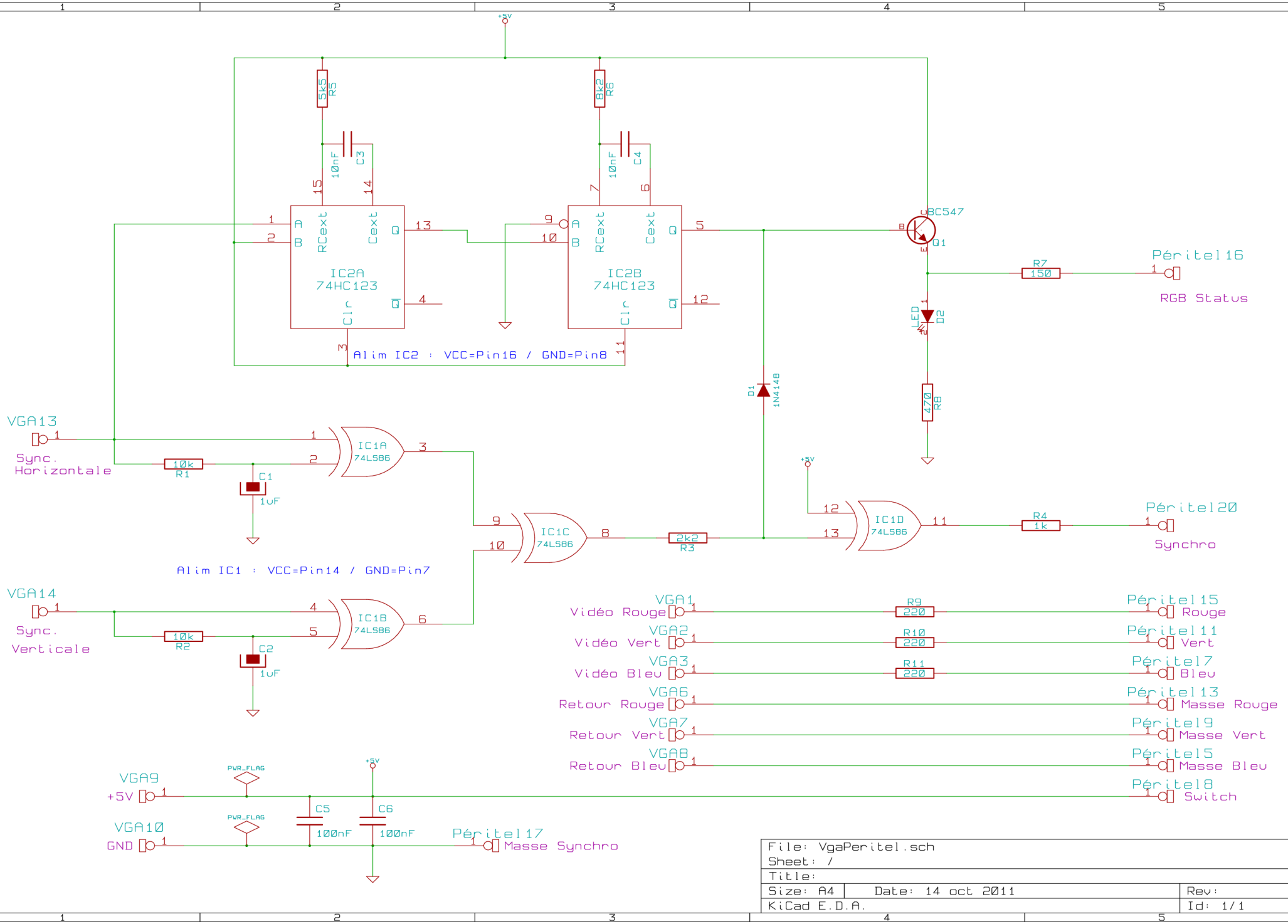





```
Signal de synchro avec X                Signal de synchro sans X
  calcul avec F = 1/T  [F:fréquence en Hertz, T:période en secondes]
             F = 1/64µs                                F = 1/20,5µs
```

Sans surprise, sous X la fréquence du signal est bien de 15,625kHz.
En mode console, donc sans X, la fréquence du signal plus élevée avec 48,780kHz.

À quoi bon réinventer la roue lorsqu'elle existe déjà !
Depuis les liens du site *AdvanceMame* [http://advancemame.sourceforge.net/video-link.html] on trouve suffisamment d'info pour satisfaire une solution [http://members.optusnet.com.au/eviltim/scart.htm] malgré le nombre important de liens brisés…
J'ai donc élaboré ce schéma, avec quelques modifications :


+5V
5k5
R5
8k2
R6
10nF
C3
10nF
C4
IC2A
74HC123
IC2B
74HC123
RCext
Cext
Clr
Alim IC2 : VCC=Pin16 / GND=Pin8
BC547
Q1
R7
150
Péritel16
RGB Status
LED
D2
1N4148
D1
470
R8
VGA13
Sync.
Horizontale
10k
R1
C1
1uF
IC1A
74LS86
IC1C
74LS86
2k2
R3
IC1D
74LS86
R4
1k
Péritel20
Synchro
Alim IC1 : VCC=Pin14 / GND=Pin7
VGA14
Sync.
Verticale
10k
R2
C2
1uF
IC1B
74LS86
VGA1
Vidéo Rouge
R9
220
Péritel15
Rouge
VGA2
Vidéo Vert
R10
220
Péritel11
Vert
VGA3
Vidéo Bleu
R11
220
Péritel7
Bleu
VGA6
Retour Rouge
Péritel13
Masse Rouge
VGA7
Retour Vert
Péritel9
Masse Vert
VGA8
Retour Bleu
Péritel5
Masse Bleu
Péritel8
Switch
VGA9
+5V
PWR_FLAG
C5
100nF
C6
100nF
VGA10
GND
PWR_FLAG
Péritel17
Masse Synchro
File: VgaPeritel.sch
Sheet: /
Title:
Size: A4
Date: 14 oct 2011
Rev:
KiCad E.D.A.
Id: 1/1

Le but de ce circuit est donc de détecter si la fréquence de synchro est bien d'environ 15kHz, et sinon de la couper pour qu'elle ne soit pas envoyé au télé.

Comment ça marche ?

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité du «Ou Exclusif».

- Les deux première portes «Ou Exclusif» du circuit intégré **74LS86** vont ajuster la polarité des signaux de synchro. Il vont ensuite être mélangés par la troisième porte, et enfin inversé par la quatrième porte pour former le signal de synchro destiné au TV

Inputs			Outputs	
$\overline{\text{Clear}}$	A	B	Q	$\overline{\text{Q}}$
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⎍	⎌
H	↓	H	⎍	⎌
↑	L	H	⎍	⎌

Truth Table

H = HIGH Level
L = LOW Level
↑ = Transition from LOW-to-HIGH
↓ = Transition from HIGH-to-LOW
⎍ = One HIGH Level Pulse
⎌ = One LOW Level Pulse
X = Irrelevant

- Le circuit **74HC123** est un double générateur d'impulsions re-déclenchables.

Il va servir à détecter si le signal de synchro est d'environ 15kHz. On règle la largeur de chacune des impulsions par la formule donnée par le constructeur PW = (Rext)x(Cext)

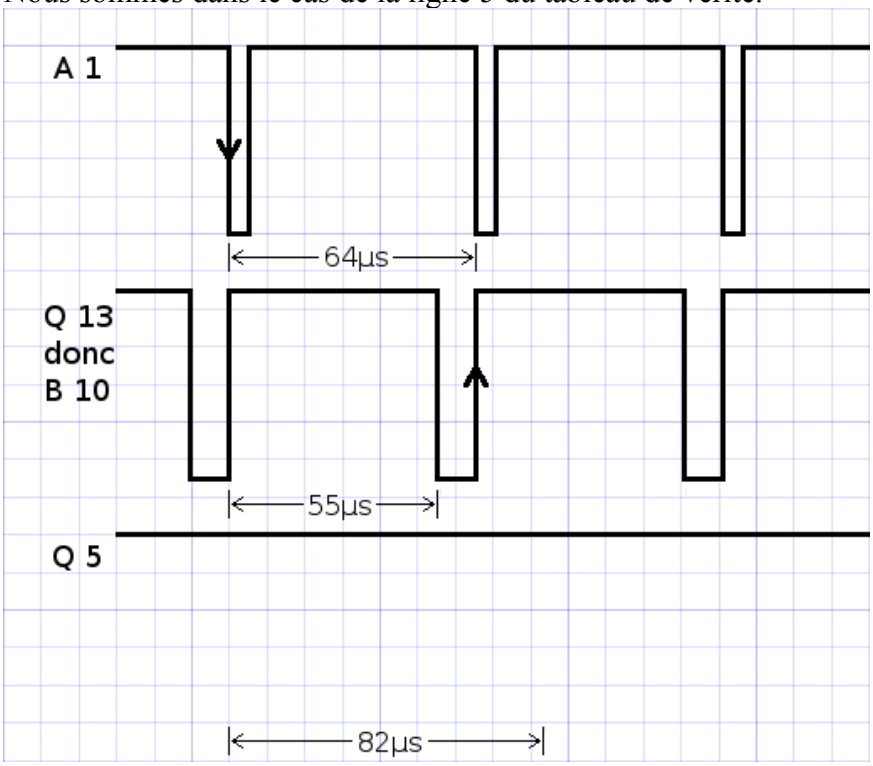
Ainsi en fixant Cext à 10nF, on détermine R5 = 5,5kΩ , pour une première impulsion de 18kHz (T=55µs) et R6 = 8,2kΩ (T=82µs).

Attention, en fonction de la marque du circuit intégré 74xx123, la formule peut-être différente.

Table de vérité du74HC123.

1 - <u>Premier cas</u> :

Voici ce qu'il se passe **lorsque le signal de 15kHz sort de la carte vidéo** pour alimenter le circuit :
Le niveau logique en B 2 est Haut.
Nous sommes dans le cas de la ligne 5 du tableau de vérité.

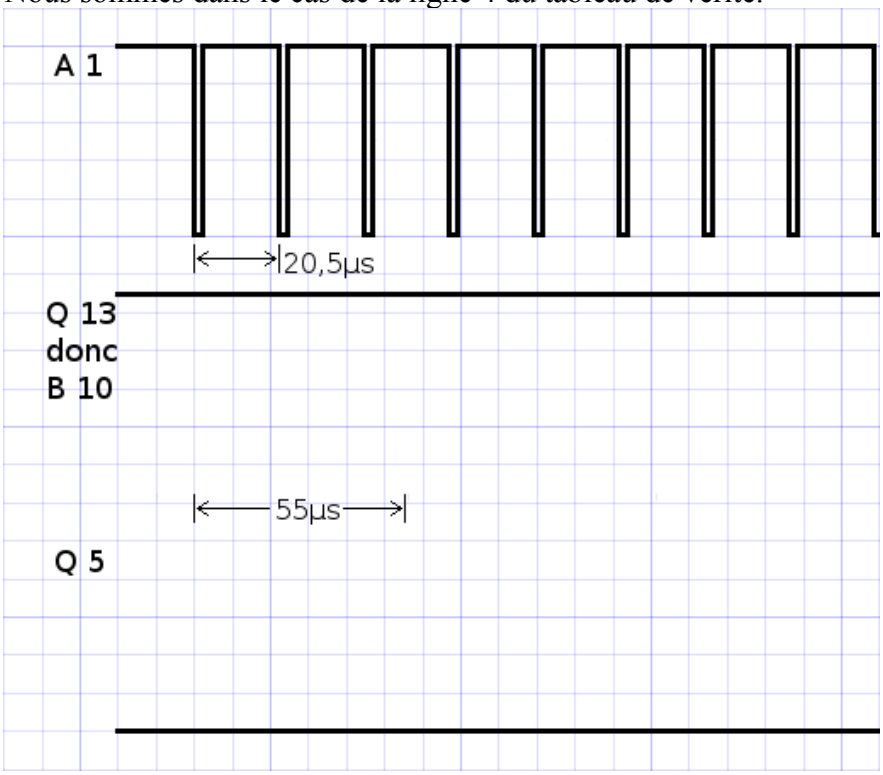


- Sur le front du signal de synchro horizontale [A 1], IC2A est déclenché et génère une impulsion de 55µS [Q 13].
Cette impulsion qui dure donc moins longtemps que la période du signal de synchro, va être re-déclenchée au front suivant du signal de synchro, soit à 64µs… Et ainsi de suite.
- Ce signal entrant en [B 10] va à son tour déclencher IC2B à l'infini puisqu'il doit générer une impulsion de 82µs et qu'il se voit re-déclenché avant sa fin, toutes les 64µs.
- Ce qui se traduit en sortie [Q 5] par un état logique Haut.

- La sortie d'IC2B va donc piloter le transistor Q1, pour allumer la Led témoin D2 et activer la prise en charge des signaux RVB par le TV via la patte 16 de la prise péritel.
- La diode D1 laisse passer le signal de synchro vers la patte 20 de la prise péritel… L'image s'affiche sur l'écran TV !!

2 - <u>Second cas</u> :

Voici ce qu'il se passe maintenant **lorsque le signal de 48kHz sort de la carte vidéo** pour alimenter le circuit :
Le niveau logique en A 9 est bas.
Nous sommes dans le cas de la ligne 4 du tableau de vérité.



- Sur le front du signal de synchro horizontale [A 1], IC2A est déclenché à l'infini puisqu'il doit générer une impulsion de 55µs et qu'il se voit re-déclenché avant sa fin, toutes les 20,5µs.
- Ce qui se traduit en sortie [Q 13] par un état logique Haut.
- Comme il n'y a pas d'impulsions pour déclencher IC2B sa sortie est à l'état logique bas.

- La sortie d'IC2B ne pilote pas le transistor Q1, la Led témoin D2 reste donc éteinte et les signaux RVB du TV sont désactivés par la patte 16 de la prise péritel.
- La diode D1 empêche le signal de synchro de passer, la patte 20 de la prise péritel ne reçoit pas de synchro … L'écran TV reste noir !!

Mission complete !

En plus de tout ça, comparé au précédent câble, j'ai ajouté 3 résistances de faibles valeurs (R9, R10, R11) sur les signaux RVB.
Elles sont tout à fait facultatives pour la grande majorité des télés.
Je n'en avais pas parlé, mais j'ai eu une mauvaise surprise avec la TV 29" qui m'affichait des couleurs surexposées… Genre le rouge s'affichait orangé…
Il semblerait donc que sur certains TV (1 sur 4 TV testés) l'amplitude des signaux RVB soit trop importante.
Pour ce cas, la valeur de 220Ω à fait l'affaire, mais il faudra expérimenter pour trouver la valeur qui restituera au mieux les couleurs et ainsi corriger leur surexposition, mais sans trop les enterrer !

Quant à l'audio, il suffit de confectionner ou récupérer un câble Jack 3,5mm mâle pour relier la sortie de la carte sonore de l'ordi aux pattes audio de la prise péritel :

Péritel 2 sur son droit.
Péritel 6 sur son gauche.
Péritel 4 pour la masse.

Joystick d'Arcade USB

Principe :

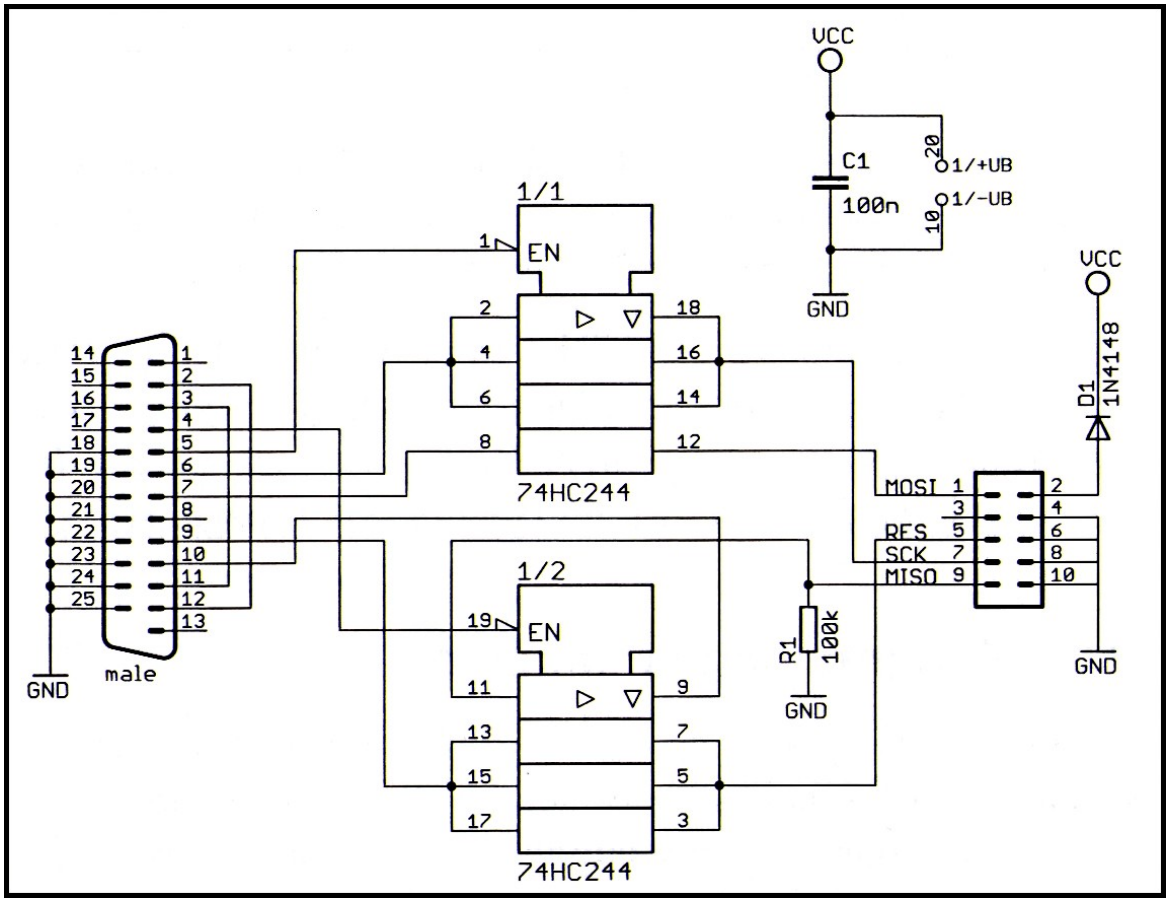
Une architecture USB est construite autour d'une puce Atmel (µC atmega8) contenant un descripteur USB HID (c'est à dire qui ne requiert pas de driver pour l'ordinateur) et les fonctions d'une manette de jeux.

Comment est-ce possible ??
Tout simplement grâce à un programme chargé dans la puce… Aie, et c'est là que c'est compliqué… Je sais pas programmer !!! Et pi programmer quoi ?? Ça n'a aucun rapport avec la programmation d'un logiciel pour ordinateur… Mis à part le langage de programmation qui est en commun, puisque c'est du «C».

En fait le µC contient plein de fonctions (port, oscillateur, registre, eeprom… voir le «Block Diagram» page 3 de la datasheet en fichier attaché à ce billet) qu'il faut programmer en leurs demandant de se causer entres-elles pour effectuer une tâche particulière…
J'en dirais pas plus, c'est des trucs d'électronicien hardcore… Peut-être qu'un jour je m'y mettrais sérieusement afin de faire mes propres programmes.
Fort heureusement, le programme qui nous intéresse est fournit avec le schéma, et il est compilé en plus !

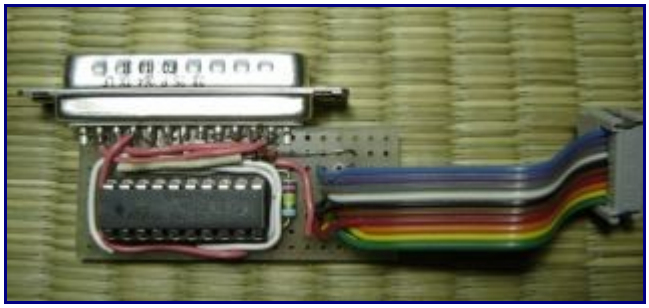
1 - Réalisation du programmateur :

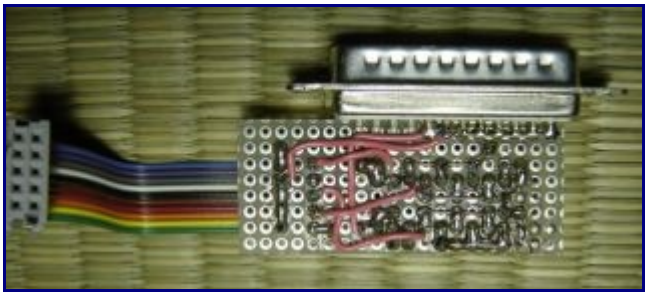
Avant toute chose donc, il va nous falloir de quoi programmer le µC avec le fichier compilé.
Pour cela il nous faut un programmateur qu'on va devoir fabriquer (à défaut d'en acheter un) et un logiciel qu'on va installer sur un ordinateur muni d'un port parallèle (port imprimante).
Voici le schéma du programmateur **stk200** (interface port parallèle-ISP) et la liste des principaux composants :



- 1 Connecteur DB25 mâle
- 1 Circuit intégré 74HC244
- 1 Condensateur 100nF
- 1 Résistance 100k Ohm
- 1 Diode 1N4148

Avec ce genre de bidouille, il vaut mieux tout câbler au plus court, ainsi j'ai coincé une carte à trou directement sur le connecteur DB25 mâle et relié tout le circuit avec des fils…





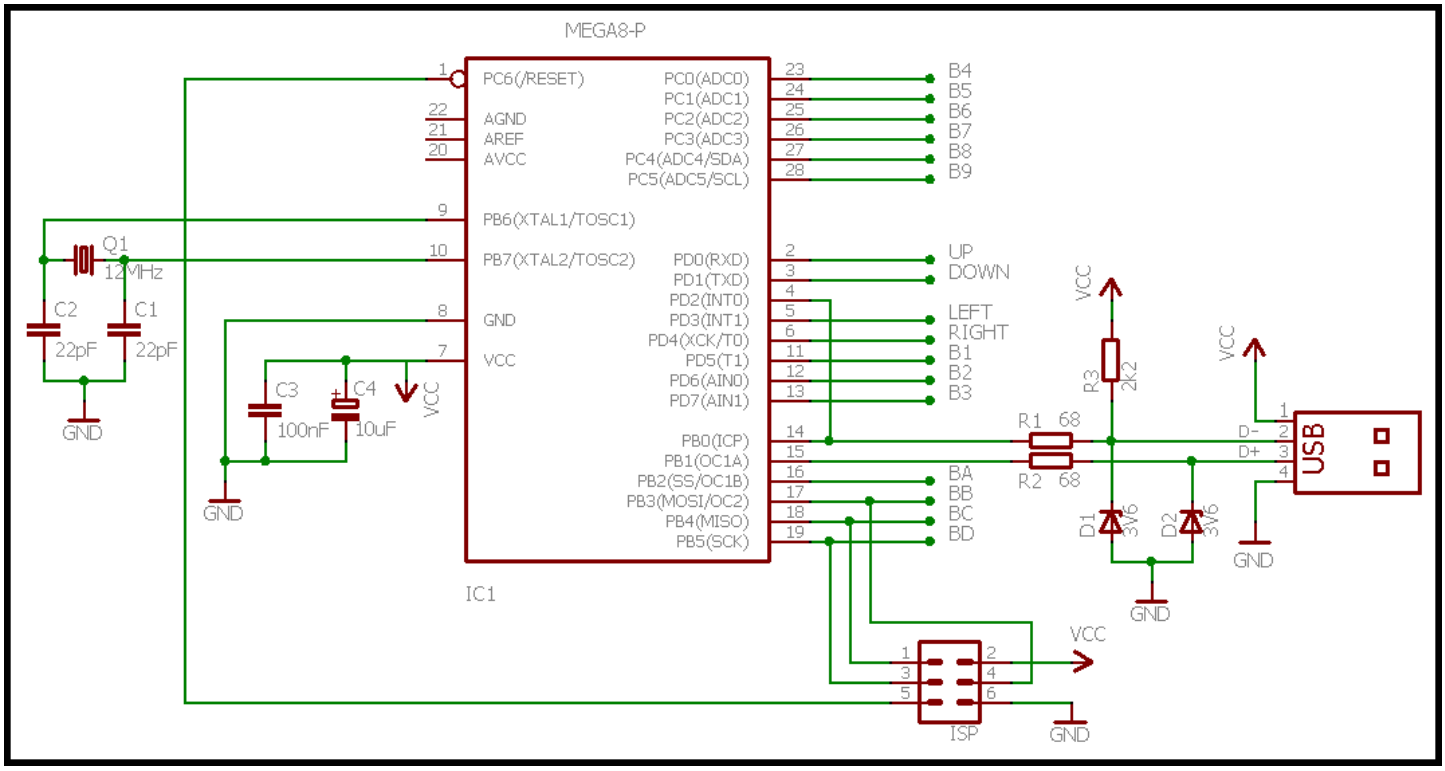
Le programmateur est terminé, reste à installer le logiciel de programmation sur Ubuntu.

- Il s'agit d'AVRdude

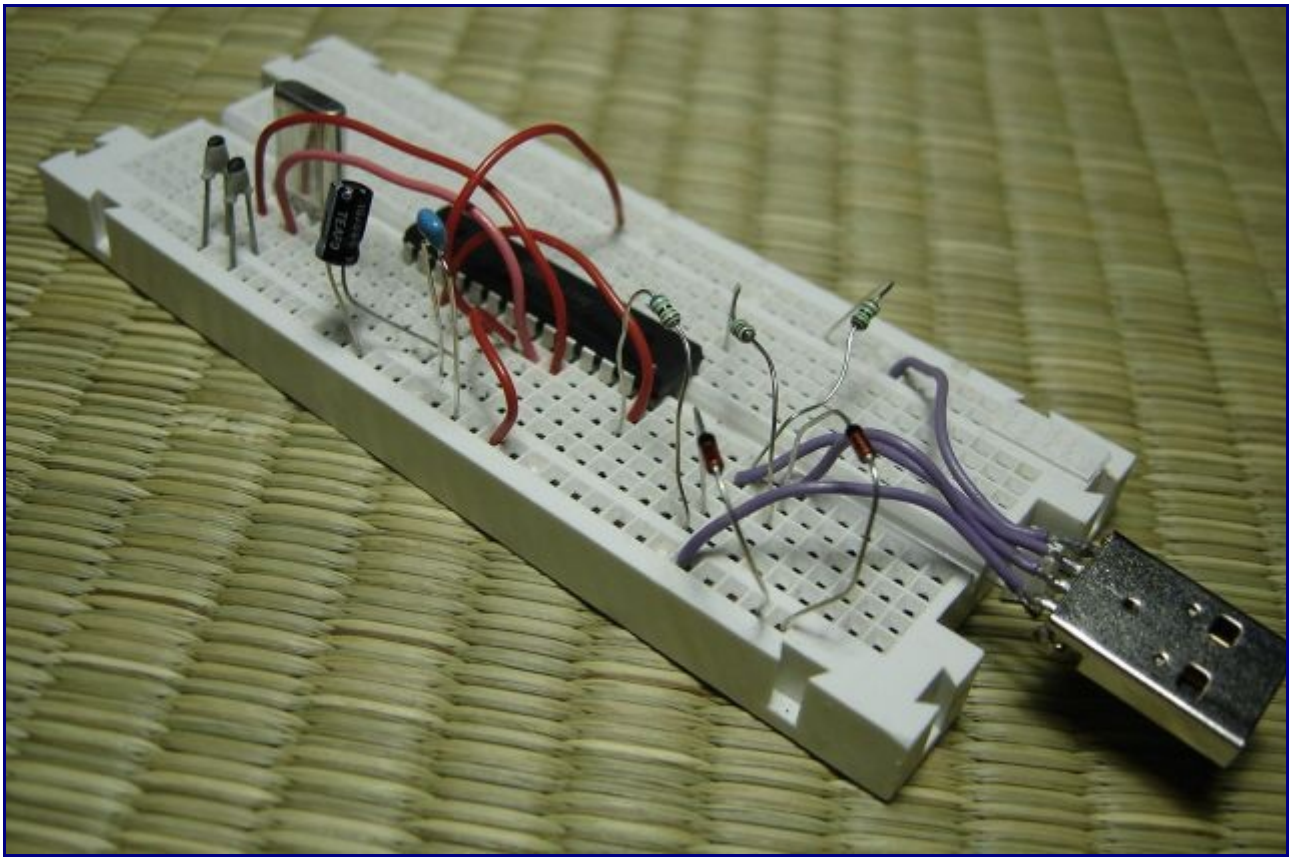
```
sudo apt-get install avrdude
```

2 - Réalisation du montage sur plaquette labo :

Il est temps d'effectuer le montage qui nous intéresse.
Ici celui de denki.world3, que j'ai pris soins de modifier, le connecteur USB dessiné pouvant prêter à confusion !

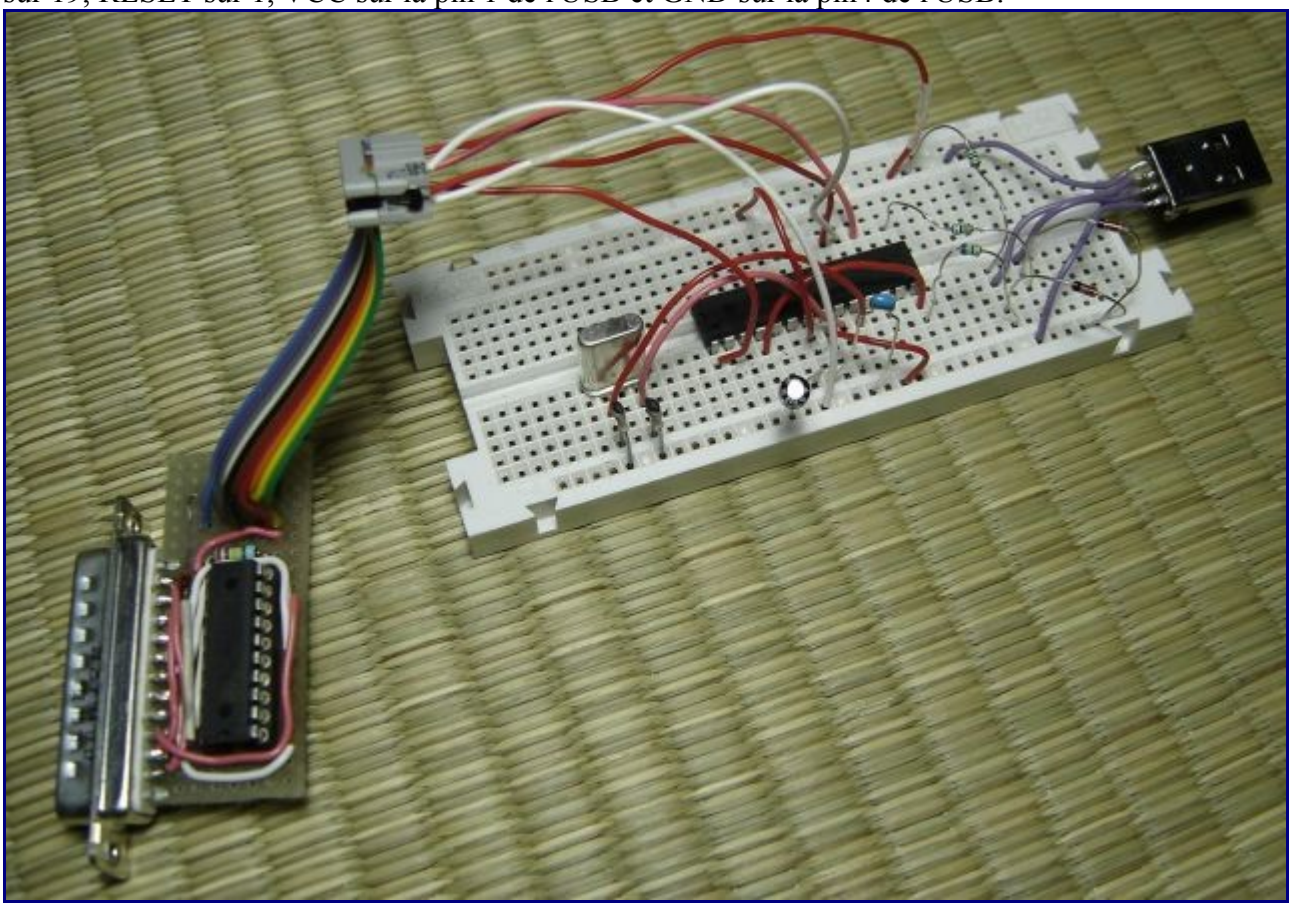


Suffit de câbler tous les composants sur la plaquette et de faire les connexions avec des fils



3 - Premiers tests :

Pour valider le programmateur, on va le connecter sur le µC comme indiqué par le schéma, en utilisant le connecteur ISP.
Il s'agit donc de connecter le fil MISO du programmateur sur la pin 18 de l'atmega8, MOSI sur 17, SCK sur 19, RESET sur 1, VCC sur la pin 1 de l'USB et GND sur la pin4 de l'USB.



Puis de brancher de connecteur DB25 du programmateur sur le port imprimante de l'ordinateur, et… d'alimenter le µC et le programmateur !!
Pour faire simple on va brancher le connecteur USB sur l'ordinateur pour qu'il alimente le µC et au travers du connecteur ISP, le programmateur.
On peut alors vérifier entre les pin 10 et 20 du CI (74HC244) du programmateur si on a bien 5V, et entre les pin 7 et 8 du µC (atmega8) également.

- Dans un terminal la commande suivante va afficher les dernier évènements de l'ordinateur.

```
dmesg
```

Elle donne :

```
[  466.216025] usb 5-1: new low speed USB device using uhci_hcd and address 2
[  466.336065] usb 5-1: device descriptor read/64, error -71
[  466.560034] usb 5-1: device descriptor read/64, error -71
[  466.776021] usb 5-1: new low speed USB device using uhci_hcd and address 3
[  466.896020] usb 5-1: device descriptor read/64, error -71
[  467.120060] usb 5-1: device descriptor read/64, error -71
[  467.336023] usb 5-1: new low speed USB device using uhci_hcd and address 4
[  467.744041] usb 5-1: device not accepting address 4, error -71
[  467.856024] usb 5-1: new low speed USB device using uhci_hcd and address 5
[  468.264027] usb 5-1: device not accepting address 5, error -71
[  468.264055] hub 5-0:1.0: unable to enumerate USB device on port 1
```

Le montage électronique est bien vu par l'ordi, mais il lui est inconnu !
Normal, le µC ne contient pas encore de programme.

Maintenant on va tester si le programmateur fonctionne, et donc s'il voit le µC qui lui est connecté par l'ISP.

- Par exemple en lisant la mémoire flash de l'atmega8

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -U flash:r:-:i -E noreset
```

Ce qui devrait répondre ainsi :

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9307
avrdude: reading flash memory:

Reading | ################################################## | 100% 2.66s

avrdude: writing output file "<stdout>"
:00000001FF

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

Super !! Ça fonctionne !

Un peu d'explications :

sudo, car on a besoin d'accéder à la ressource matérielle du port parallèle.
-p atmega8, indique le modèle de µC.
-P /dev/parport0, spécifie le port sur lequel est connecté le programmateur (port parallèle).
-c stk200, indique le nom du programmateur.
-U flash:r:-:i -E noreset, pour s'adresser à la mémoire flash du µC en mode lecture.

4 - <u>Programmation de la puce</u> :

Reste à programmer le µC pour y mettre le fichier *.hex* donné par le site denki.world3.net.
(fichier contenu dans *JoystickControllerV1.1.zip*)

- Programmer un fichier HEX dans la mémoire flash

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -Uflash:w:/chemin vers le
fichier/Arcade_Joystick.hex
```

Ce qui renvoie :

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9307
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "/home/makoto/Arcade_Joystick.hex"
avrdude: input file /home/makoto/Arcade_Joystick.hex auto detected as Intel Hex
avrdude: writing flash (2226 bytes):

Writing | ################################################## | 100% 0.93s

avrdude: 2226 bytes of flash written
avrdude: verifying flash memory against /home/makoto/Arcade_Joystick.hex:
avrdude: load data flash data from input file /home/makoto/Arcade_Joystick.hex:
avrdude: input file /home/makoto/Arcade_Joystick.hex auto detected as Intel Hex
avrdude: input file /home/makoto/Arcade_Joystick.hex contains 2226 bytes
avrdude: reading on-chip flash data:
```

```
Reading | ################################################## | 100% 0.76s

avrdude: verifying ...
avrdude: 2226 bytes of flash verified

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

Un nouveau *dmesg* montrera que le montage USB est toujours inconnu…
Il manque une étape cruciale, **la programmation des bits fusibles**.
En effet, sans cela l'oscillateur @12Mhz du montage ne sera pas pris en compte par le µC qui continuera d'utiliser son oscillateur interne.

denki.world3 nous indique en fichier texte que les bit fusibles doivent recevoir les valeurs 0xc9 et 0x9f.

On fait cela avec la commande suivante :

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -Uhfuse:w:0xc9:m
-Ulfuse:w:0x9f:m
```

Ce qui donne :

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9307
avrdude: reading input file "0xc9"
avrdude: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of hfuse written
avrdude: verifying hfuse memory against 0xc9:
avrdude: load data hfuse data from input file 0xc9:
avrdude: input file 0xc9 contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of hfuse verified
avrdude: reading input file "0x9f"
avrdude: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against 0x9f:
avrdude: load data lfuse data from input file 0x9f:
avrdude: input file 0x9f contains 1 bytes
avrdude: reading on-chip lfuse data:

Reading | ################################################## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of lfuse verified

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

5 - Test final :

- Sans rien toucher, le montage devrait être instantanément reconnu comme un périphérique USB.

```
dmesg
```

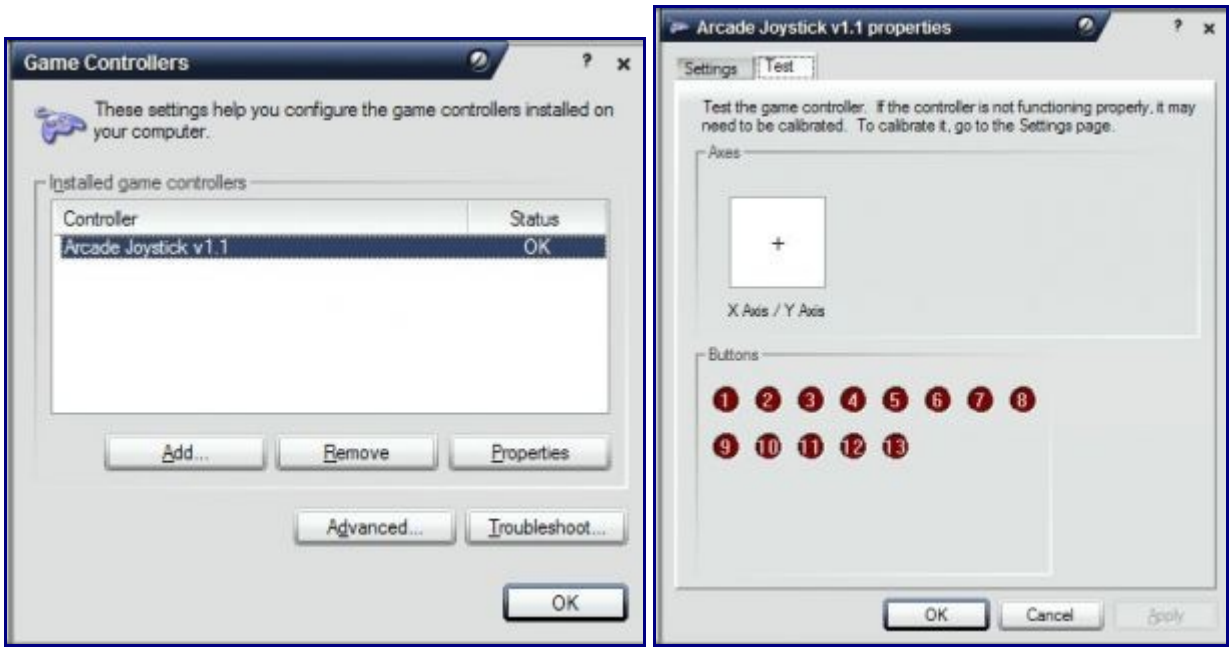
Dit :

```
[ 2573.624031] usb 5-1: new low speed USB device using uhci_hcd and address 42
[ 2573.799441] usb 5-1: configuration #1 chosen from 1 choice
[ 2573.843349] input: Paul Qureshi Arcade Joystick v1.1 as
/devices/pci0000:00/0000:00:08.0/0000:01:09.1/usb5/5-1/5-1:1.0/input/input8
[ 2573.846966] generic-usb 0003:8282:8001.0004: input,hidraw0: USB HID v1.01
Gamepad [Paul Qureshi Arcade Joystick v1.1] on usb-0000:01:09.1-1/input0
```

Reste à utiliser son logiciel préféré pour tester le «joystick» en reliant à la masse successivement les sorties du µC, pour simuler l'appuie sur les boutons.

```
jstest /dev/input/js0
```

Sur La console de jeux qu'est Windows®, ça marche aussi !



Quelques commandes utiles :

- Lire l'état des bits de sécurités

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -U lock:r:-:i -E noreset
```

- Lire l'état des bits fusibles

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -Uhfuse:r:-:i -E noreset
```

- Effacement total du µC

```
sudo avrdude -p atmega8 -P /dev/parport0 -c stk200 -e
```

RÉALISATIONS

INSTALLER le système Arcade sur l'ordinateur

1 - Installation du système de base Ubuntu minimal :

Télécharger l'image iso d'**Ubuntu 10.04 Alternate**, graver le cd et booter l'ordi dessus, au menu :

1. Choisir la langue.
2. Appuyer sur F4 et sélectionner *installer un système en ligne de commande*, Valider.
3. Valider sur *installer Ubuntu*.

http://releases.ubuntu.com/lucid/ubuntu-10.04.3-alternate-i386.iso.torrent
ftp://ftp.free.fr/mirrors/ftp.ubuntu.com/releases/lucid/ubuntu-10.04.3-alternate-i386.iso

2 - Préparatifs :

Télécharger **AdvanceMameInstall.tar.gz** [http://burogu.makotoworkshop.org/public/ordi/softs/AdvanceMameInstall.tar.gz] et le copier via un média de stockage externe dans le *home*.
Puis l'extraire :

```
tar -zxvf AdvanceMameInstall.tar.gz
```

Rendre le script d'installation exécutable :

```
chmod 777 ~/AdvanceMameInstall/install.sh
```

3 - Installation automatique :

Lancer le script d'installation :

```
./install.sh
```

Redémarrer l'ordinateur,
C'est terminé :) !

Restera à copier vos roms dans *~/.advance/rom*
et copier vos snapshot dans *~/.advance/snap* pour avoir les photos d'écran de chaque jeux via le menu de l'émulateur.
Pour arrêter le système, suffit d'appuyer sur le bouton power.

4 - Quelques explications sur les commandes du script :

- Installation de l'environnement graphique *Xorg* et de *SDL* :

```
sudo apt-get update && sudo apt-get install xserver-xorg-video-all xfonts-base xinit
x11-xserver-utils alsa-base libsdl1.2debian sshfs mingetty acpid
```

- Copie des fichiers de configuration pour permettre à X de fonctionner sur une TV via une carte graphique ATI

```
sudo cp xorg.conf /etc/X11/xorg.conf
```

- Copie des fichiers pour login automatique, configuration et démarrage automatique de l'émulateur.

```
sudo cp tty1.conf /etc/init/tty1.conf
cp profile ~/.profile
cp xinitrc ~/.xinitrc
mkdir ~/.advance
cp advmame.rc ~/.advance
cp advmenu.rc ~/.advance
```

- Installation de l'émulateur par compilation :

```
sudo apt-get install g++ checkinstall libsdl1.2-dev libslang2-dev libncurses5-dev
libfreetype6-dev zlib1g-dev libexpat1-dev
tar -zxvf advancemame-0.106.1.tar.gz
cd ./advancemame-0.106.1
./configure
make
sudo make install
cd ..
tar -zxvf advancemenu-2.5.0.tar.gz
cd ./advancemenu-2.5.0
./configure
make
sudo make install
```

- Nettoyage des paquet téléchargés :

```
sudo apt-get clean
```

- Config manuelle pour régler les niveaux sonores de la carte son, démuter les tranches et régler les faders… Valider avec la touche *Échap*

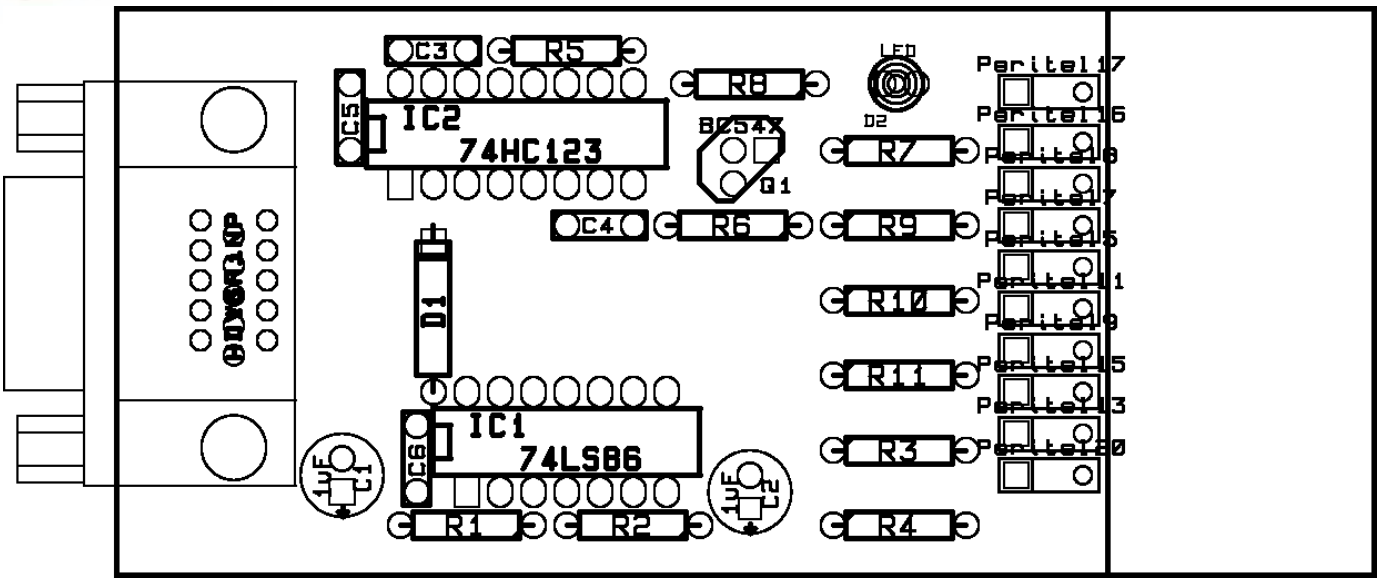
```
sudo alsamixer
sudo alsactl store 0
sudo gpasswd -a makoto audio
```

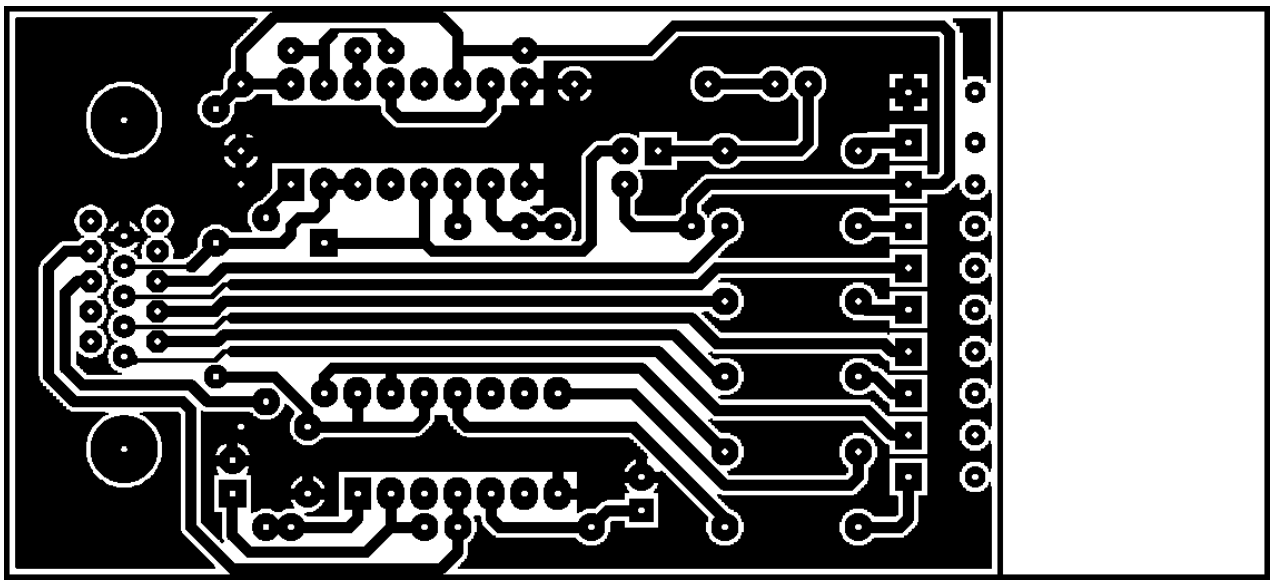
Conclusion :

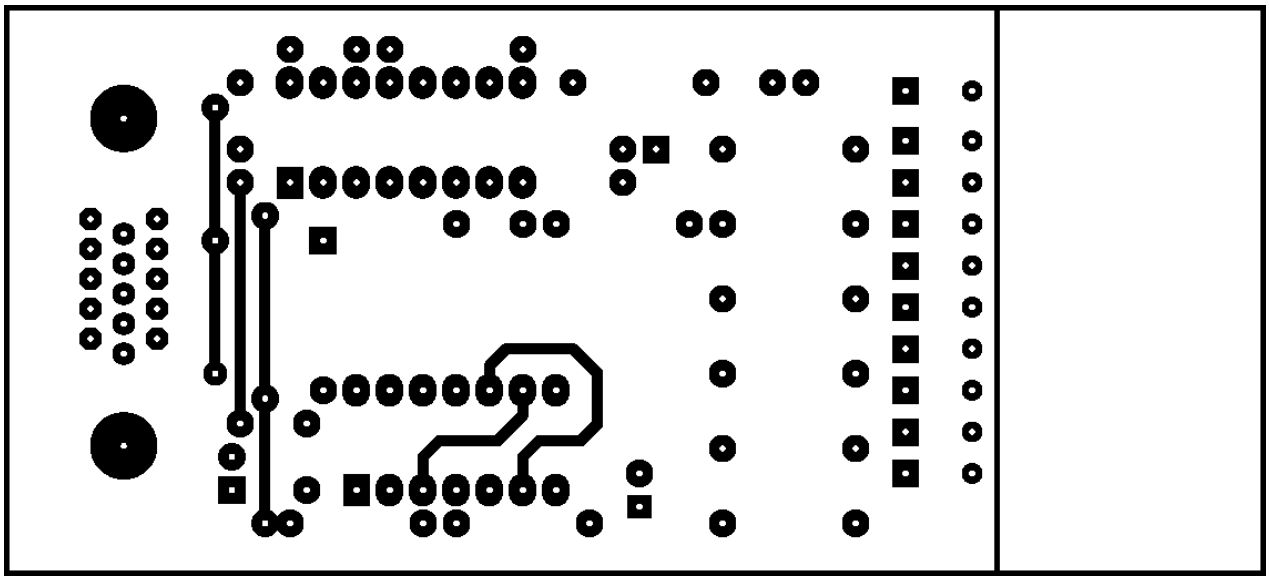
- Je propose aussi une alternative d'installation qui contient l'émulateur déjà compilé : **AdvanceMameInstall-Compile.tar.gz [http://burogu.makotoworkshop.org/public/ordi/softs/AdvanceMameInstall-Compile.tar.gz]**, le fichier est plus lourd, mais elle sera plus rapide d'exécution, la compilation pouvant durer jusqu'à 1h sur une machine de 10 ans d'âge.
- Ce système pèse environ 800 Mio, sans superflu, il démarre très vite, idéal pour être utilisé en borne, sans clavier ni souris.
- Les 5 boutons que je n'ai pas câblé dans mon joystick vont alors pouvoir servir à donner les ordres au menu et à l'émulateur… En réalisant deux joystick pour la borne ça fera donc 10 boutons assignables ^^
- Pour info, le *modeline* contenu dans le fichier *xorg.conf* semble relativement universel, puisqu'il a fonctionné sur 4 télés différents, et plusieurs modèles de cartes graphique… avec un peu de bol donc, y'a même pas à se casser la tête à l'adapter, mais faites y gaffe quand même !
- Astuce ! Pour jouer en *tate*, ajouter *vertical/display_rol yes* dans le fichier *advmame.rc*. On pourra alors basculer l'écran sur le côté pour jouer tous les jeux type *Shoot'em up* vertical. Les couleurs du Télé vont délirer un peu, pour arranger ça, suffit d'éteindre l'écran quelque minute (équivalent du dégauss).

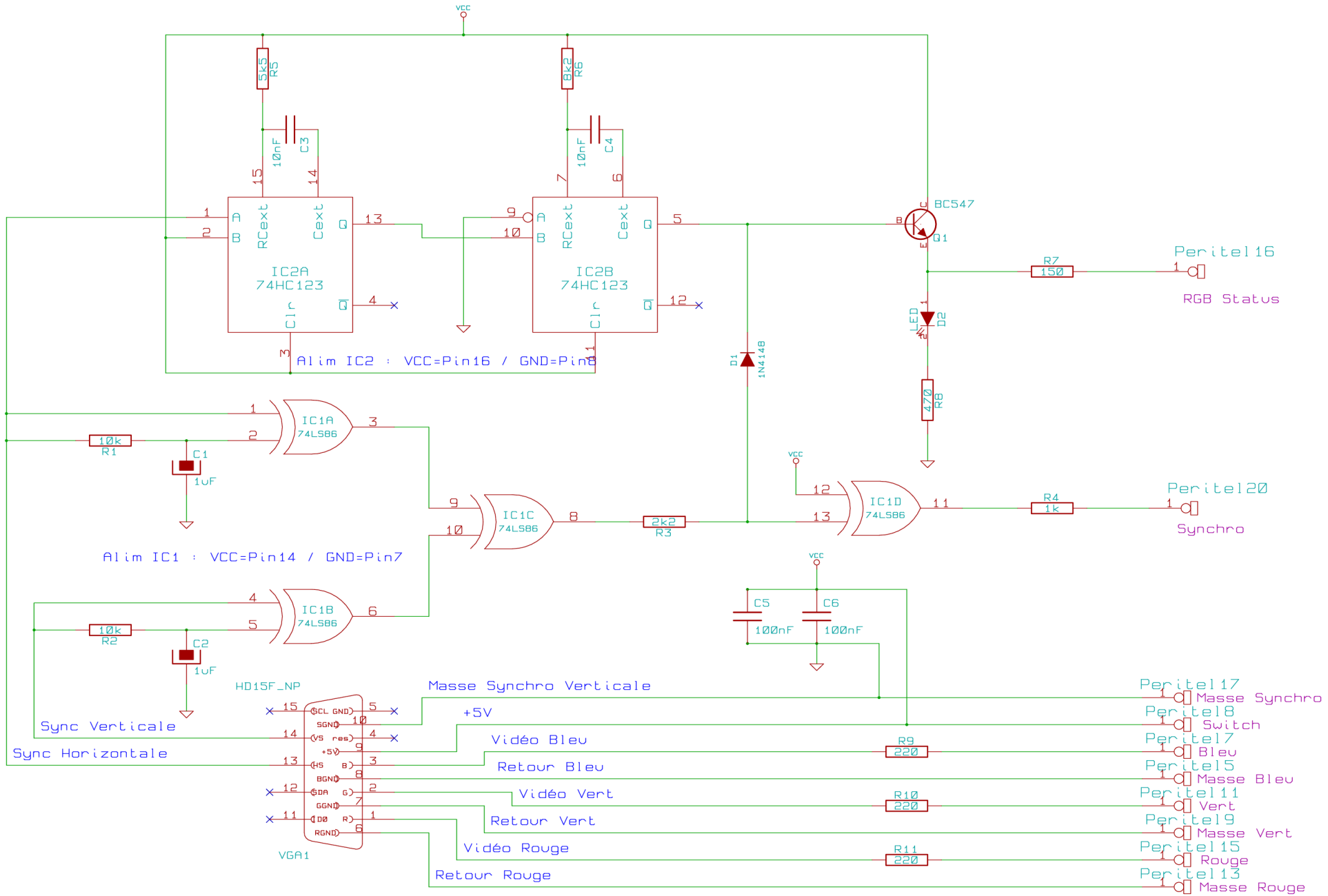
Câble adaptateur VGA vers Péritel protégé

J'ai modifié le schéma structurel afin d'y ajouter l'empreinte d'une prise VGA (HD15 points femelle).









File: VgaPeritel2.sch		
Sheet: /		
Title:		
Size: A4	Date: 14 oct 2011	Rev:
KiCad E.D.A.		Id: 1/1

Ici réalisé en simple face, les quelques pistes manquantes sont remplacées par des fils.
Donc une fois le typon imprimé et gravé sur la plaque d'époxy [http://wiki.jelectronique.com/realisation_de_circuits_imprimes] , il nous reste à rassembler tout le matériel nécessaire :

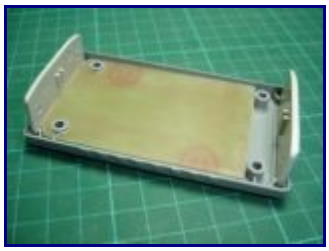
1 LED faible consommation - 5mm - Verte
1 Diode de signal et de commutation 1N4148
1 Circuit logique HC - 74HC123
1 Circuit logique LS - 74LS86 ou 74HC86
1 Transistors bipolaire NPN - BC547B
1 Résistance à couche métallique 5 % 1/2W de 5,6 kohm
1 Résistance à couche métallique 5 % 1/2W 8,2 kohm
1 Résistance à couche métallique 5 % 1/2W de 2,2 kohm
1 Résistance à couche métallique 5 % 1/2W de 470 ohm
1 Résistance à couche métallique 5 % 1/2W de 150 ohm
2 Résistance à couche métallique 5 % 1/2W de 10 kohm
1 Résistance à couche métallique 5 % 1/2W de 1 kohm
2 Condensateur MKT LCC 5mm 100nF 63V
2 Condensateur MKT LCC 5mm 10nF 63V
2 Condensateur électrochimique polarisé sortie radiale - 1µF - 63V
1 Fiche SCART (péritel) Mâle
1 1m de câble péritel PRO
1 Connecteur SUB-D circuit imprimé coudé HD 15 Femelle
1 Coffret série 400 profilés en ABS G-403 - 90x50x24mm

Ce qui devrait couter environ 5€ hors prix de la plaque d'époxy et du câble/connecteur Péritel qu'on peut trouver facilement de récupération.

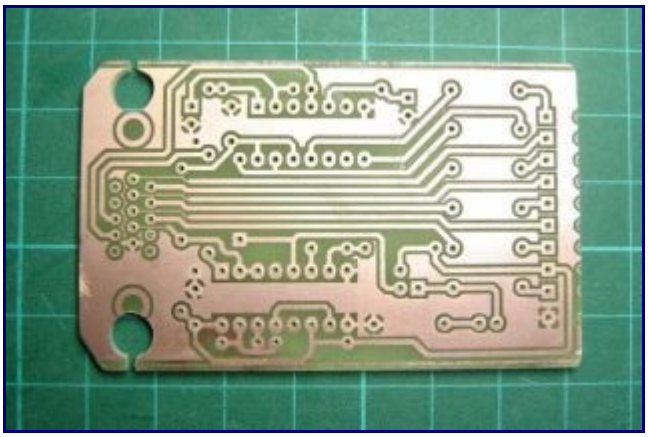
Sur ce modèle, j'ai choisi d'utiliser un câble VGA de récupération qui sera soudé en lieu et place du connecteur VGA, et d'utiliser un boitier pour y loger le circuit :

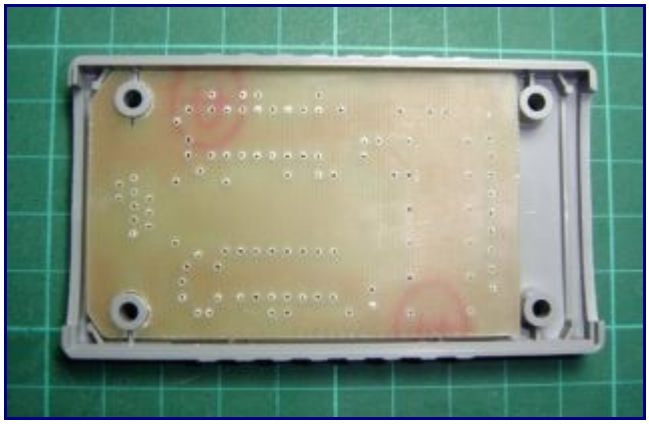

Il faut commencer par adapter un peu le boitier et usiner l'époxy pour pouvoir le rentrer dans le logement.



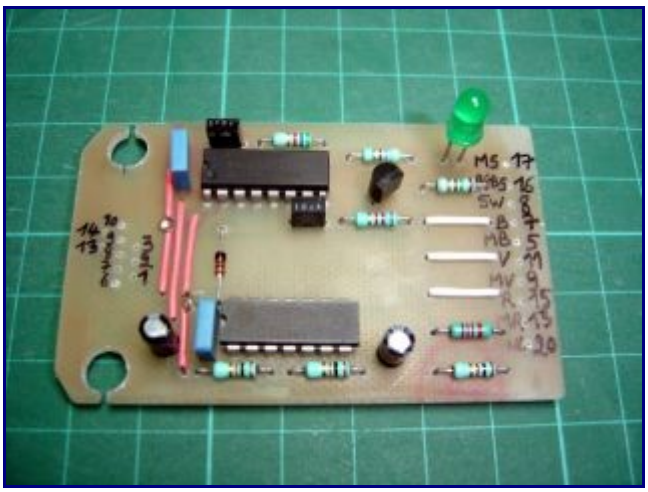



Puis on peut procéder au perçage du circuit…

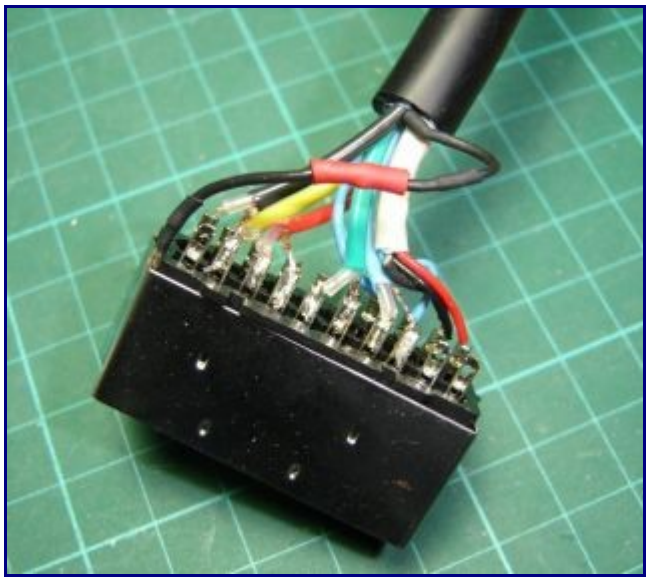




Voilà, le circuit est prêt et le boitier a été usiné pour accueillir la led, les câbles péritel et VGA, ainsi qu'une prise Jack 3,5 mm femelle pour connecter l'audio, si on désire faire passer le son en provenance de la carte son dans les haut-parleurs du TV plutôt que d'utiliser des enceintes amplifiées externes.



Tous les éléments avant assemblages…



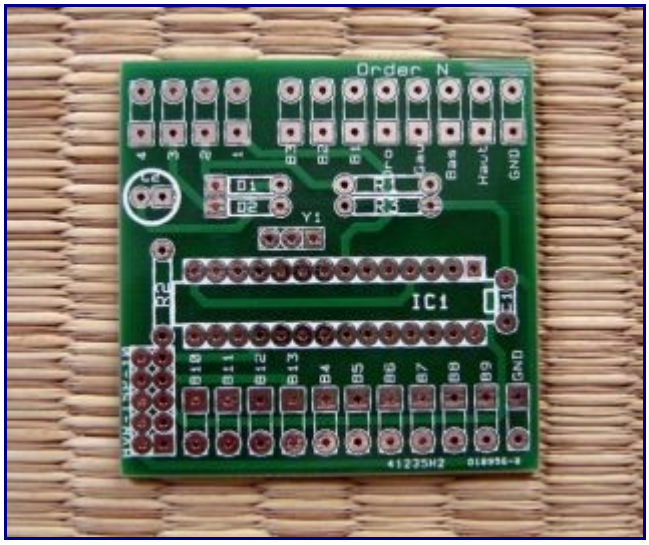
Le plus chiant étant de souder les fils sur la plaque époxy, en s'efforçant le moins de manipulations possible en vue d'éviter trop de torsion des câbles, provoquant alors la rupture des fils à raz du circuit… Les serres câbles sont là pour éviter l'arrachement des câbles, et même leurs rotations à l'intérieur du boitier, si on a pris soins de les placer judicieusement.

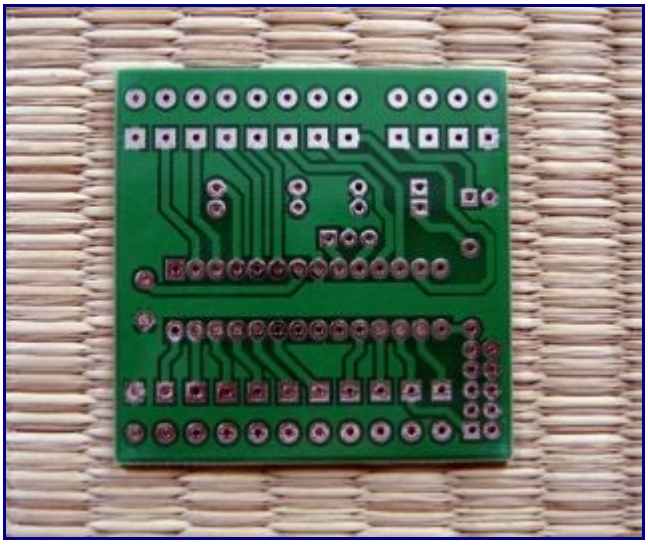
Et c'est… fini !

Joystick d'Arcade USB

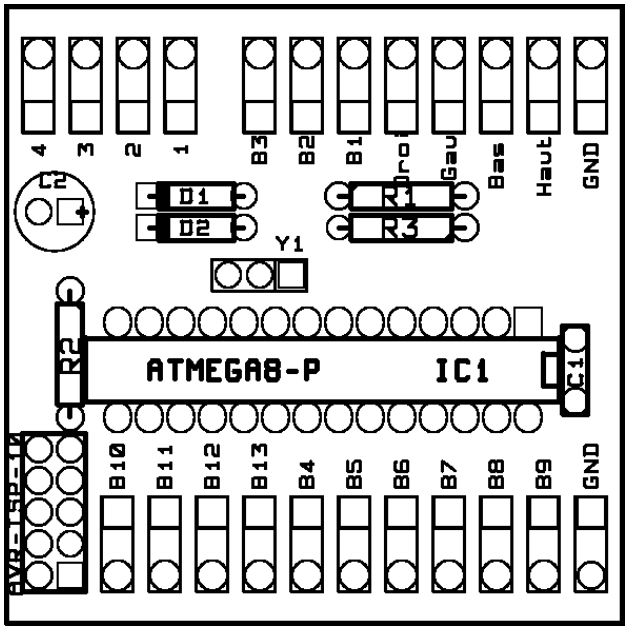
Pour le tirage de la carte électronique, j'ai décidé de faire appel aux services en ligne de Seeedstudio.com



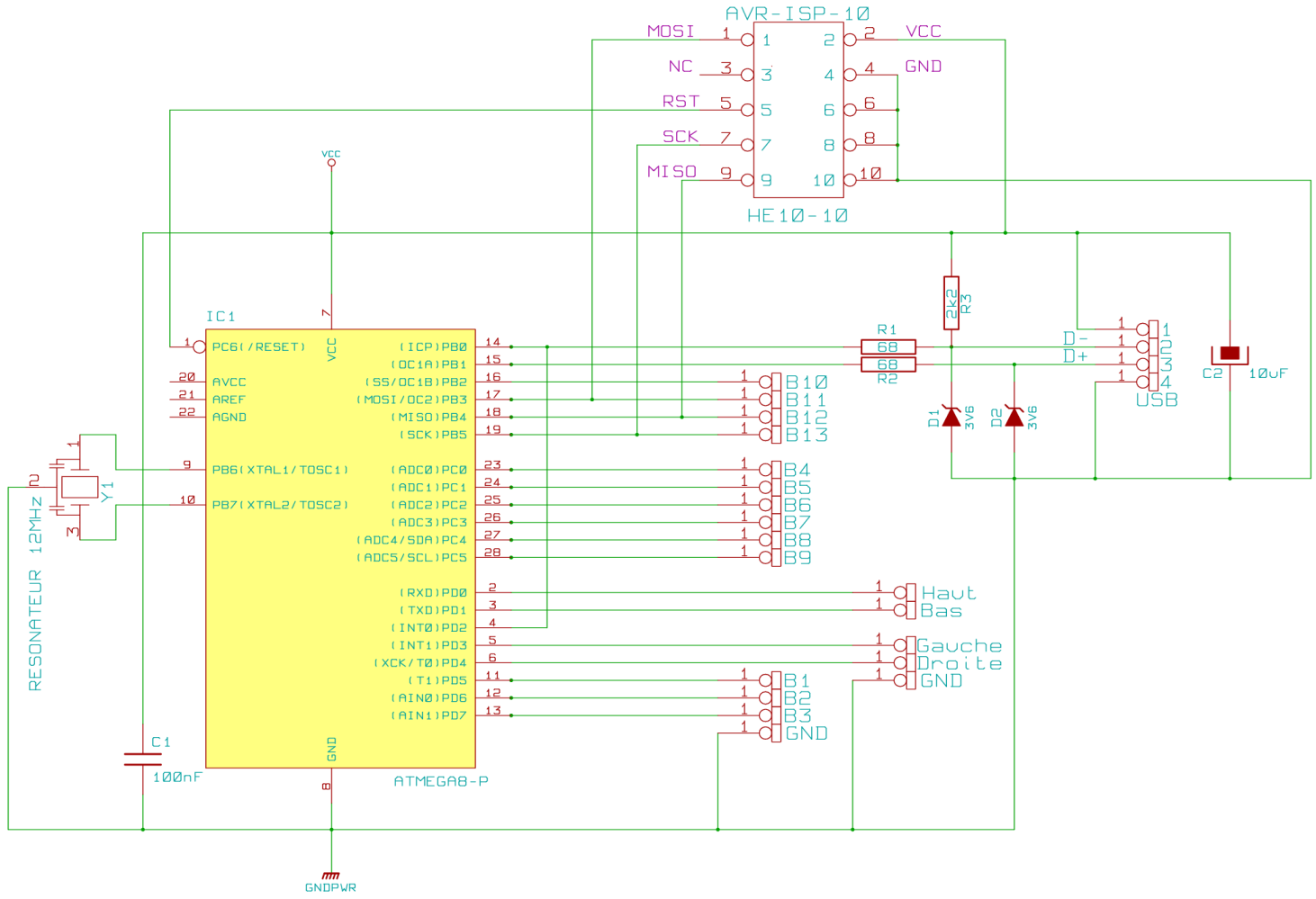




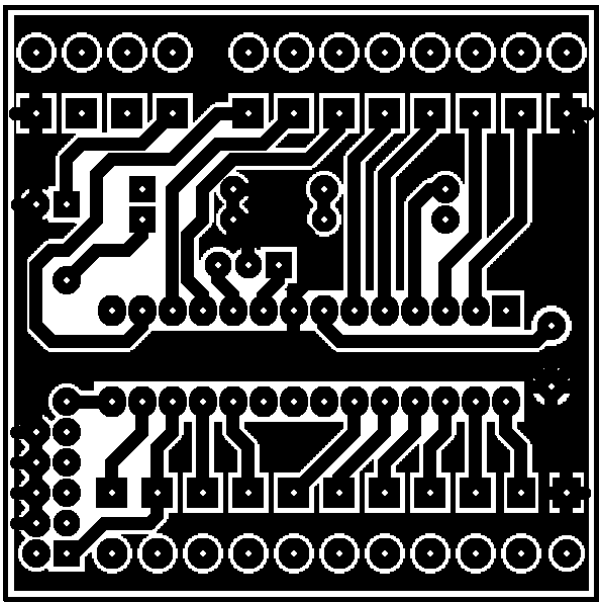
Reste alors à souder les composants en se référant au schéma structurel et à l'implantation :

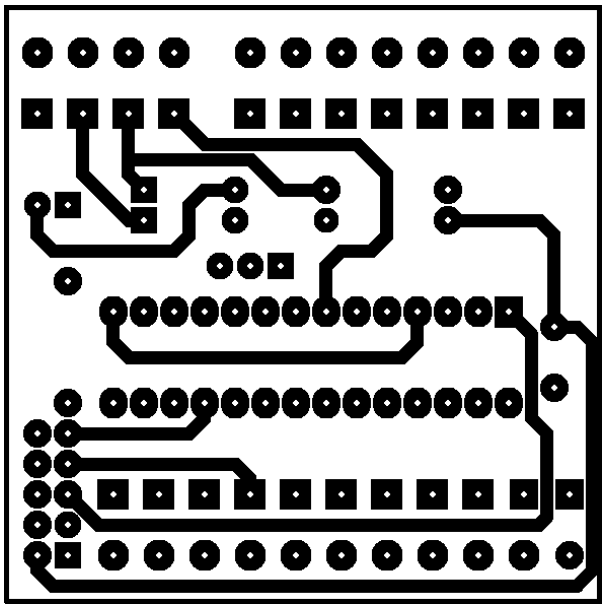


- 2 Résistances à couche métallique 5 % 1/2W de 68 ohm
- 1 Résistance couche métal 1/2W 2,2 kohm
- 1 Résonateur céramique THT CSTLF 12.00MHz
- 1 Condensateur MKT LCC 5mm 100nF 63V
- 1 Condensateur radial 10µF 50V
- 2 Diode zener standard BZX55C - 500mW - 3,6V
- 1 Barette Mâle 2x10
- 1 cordon USB de récup… genre une vieille souris ?
- 1 ATMEGA8 -16PU

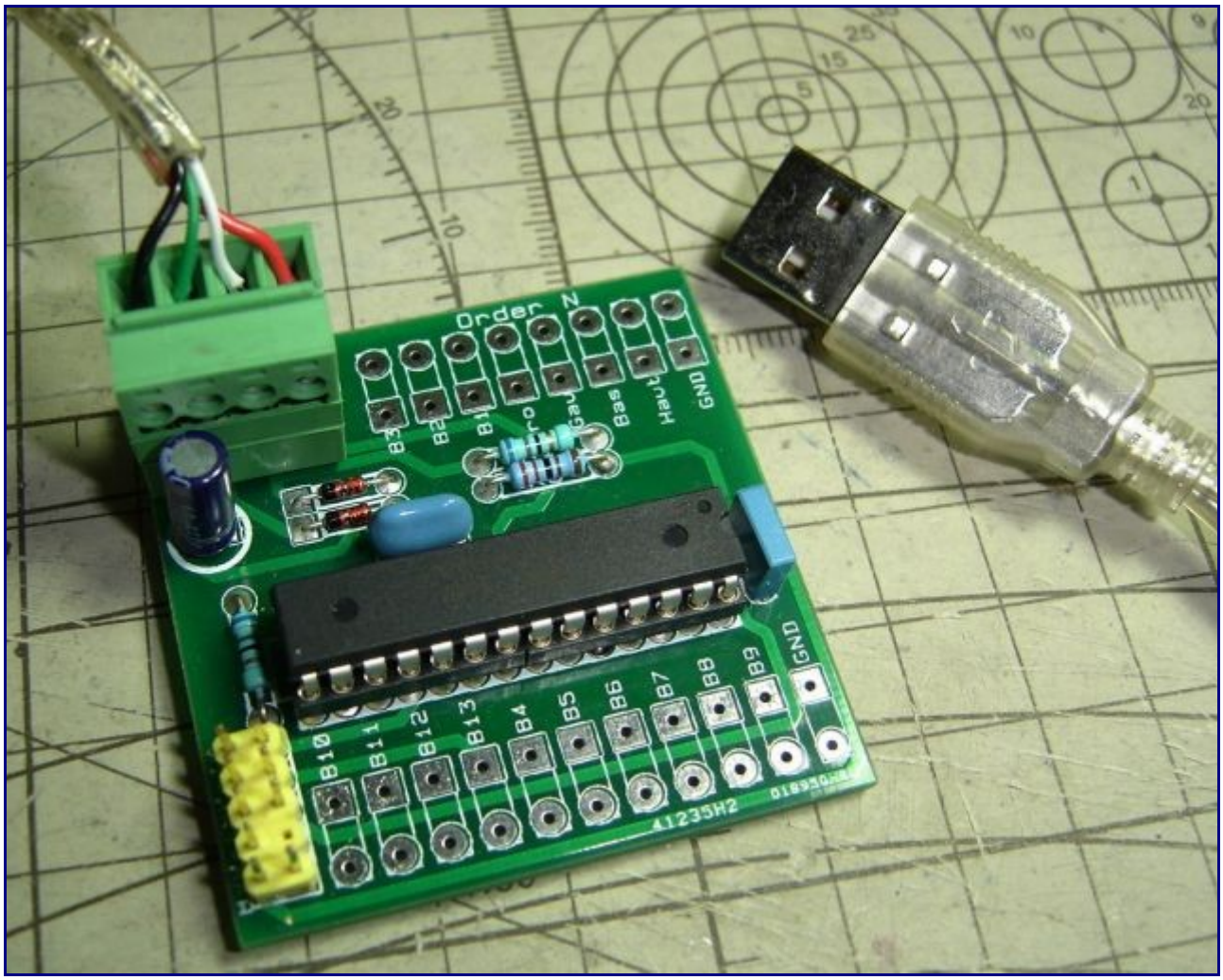


File: joystick.sch		
Sheet: /		
Title:		
Size: A4	Date: 30 sep 2011	Rev:
KiCad E.D.A.		Id: 1/1



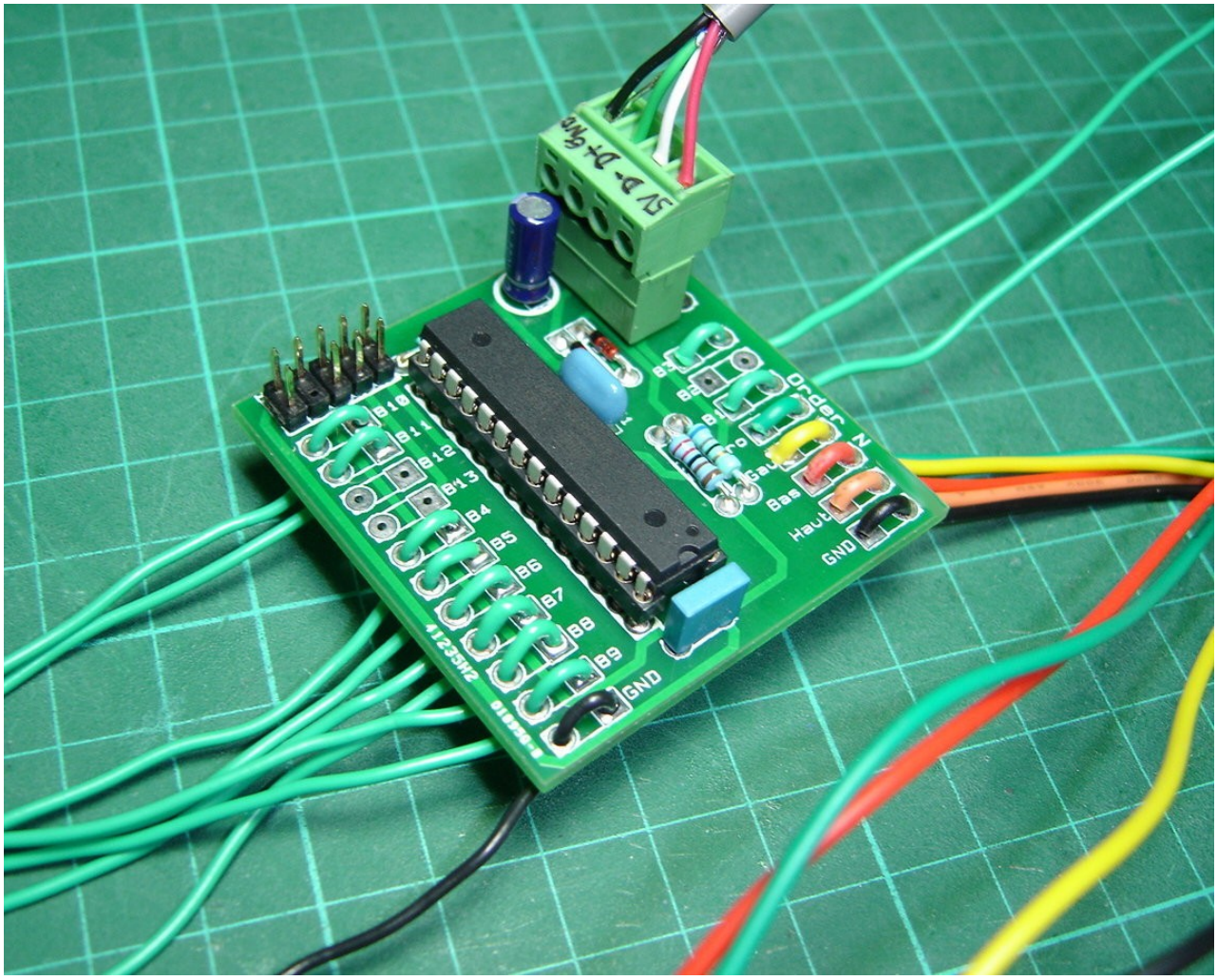


Et voila !!



Les trous carrés vont recevoir soit des borniers ou des connecteurs, soit des fils à destinations des boutons et du joystick.

Dans ce cas, les trous ronds pourront être agrandis à la perceuse, afin d'y passer le fil (gaine incluse) par le dessous de la carte, et le passer dans le trou carré correspondant pour l'y souder … ceci histoire de rendre plus solide l'attache du fil. Rien de plus emmerdant que de se retrouver avec un fil qui se coupe et se détache à force de contraintes mécaniques et manipulations.



- À titre indicatif, un exemplaire de carte à souder de chez seeedstudio revenant à environ 1€, le prix de revient d'une carte complète est d'environ 9€, auquel il faut ajouter le programmateur USBASP pour programmer le µC Atmel.

Il est bien entendu possible de faire baisser ce coût en fonction de la récup qu'on aura pu faire en canibalisant d'autres appareils…
Sur la photo, les picots viennent d'une carte mère d'ordi, le câble USB d'un Clavier, le connecteur et le support du µC (facultatif mais bien pratique) d'une carte électronique.