

---

INSTITUT SAINT JEAN BERCHMANS – SAINTE MARIE  
Section Transition Sciences – Informatique

# **PROGRAMMATION GAME TOY**

Travail de fin d'études  
réalisé par  
VAUSE Romain

---

Année académique 2014 - 2015

## Sommaire

Introduction.....	7
GameToy.....	7
LineRun.....	7
Space Invaders.....	7
Scénario principal.....	7
GameToy.....	7
LineRun.....	8
Jeu.....	8
Records.....	8
Options.....	8
Quitter.....	9
Space Invaders.....	9
Jeu.....	9
Quitter.....	10
Manuel d'utilisation.....	11
GameToy.....	11
LineRun.....	12
Menu principal.....	12
Jouer.....	13
Options.....	14
Records.....	15
Space Invaders.....	16
Menu principal.....	16
Jouer.....	16
Développement.....	18
Méthode de développement.....	18
GameToy.....	18
Composants.....	18
Raspberry PI Model B+.....	18
Écran TFT touchscreen.....	19
Bouton poussoirs.....	19
Batterie Li-po 1S.....	20
Chargeur de batteries Li-po.....	21
Booster de tension.....	21
Push Button ON/OFF.....	22
Coque.....	22
Câble Ribbon.....	23
Outils.....	23

Fer à souder.....	23
Gaine thermorétractable.....	24
Logiciel de lancement.....	25
Classe.....	25
System.....	25
Graphical.....	27
Fonctions externes.....	28
Conception de la console.....	29
Rechargement / alimentation.....	29
Bouton d'allumage.....	29
Utilisation des boutons pousoirs.....	31
Communiquer avec la carte.....	35
Fixation de la Raspberry PI.....	38
Sortie HDMI.....	39
Mise en couleur de la coque.....	40
Jeux.....	40
PyGame.....	41
Menu, jeu, records et options.....	42
Méthodes de base.....	42
Les classes externes.....	43
Les fonctions externes.....	43
Phases de développement.....	44
GameToy.....	44
Achat ✕.....	44
Objectif 1 ✕.....	44
Objectif 2 ✕.....	44
Objectif 3 ✕.....	44
Objectif 4 ✕.....	44
Objectif 6 ✕.....	44
Design de la coque ✕.....	44
Objectif 1 ✕.....	44
Objectif 2 ✕.....	44
Objectif 3 ✕.....	44
Objectif 4 ✕.....	44
Affichage ✕.....	45
Objectif 1 ✕.....	45
Objectif 2 ✕.....	45
Circuit d'alimentation ✕.....	45
Objectif 1 ✕.....	45
Objectif 2 ✕.....	45
Objectif 3 ✕.....	45
Objectif 4 ✕.....	45
Touches fonctionnelles ✕.....	45

Objectif 1 <input checked="" type="checkbox"/>	45
Software <input checked="" type="checkbox"/>	45
Objectif 1 <input checked="" type="checkbox"/>	45
Objectif 2 <input checked="" type="checkbox"/>	45
LineRun.....	46
Étape 0 <input checked="" type="checkbox"/>	46
Chemin aléatoire <input checked="" type="checkbox"/>	46
Objectif 1 <input checked="" type="checkbox"/>	46
Objectif 2 <input checked="" type="checkbox"/>	46
Point (Joueur) <input checked="" type="checkbox"/>	46
Objectif 1 <input checked="" type="checkbox"/>	46
Objectif 2 <input checked="" type="checkbox"/>	46
Gestion du score <input checked="" type="checkbox"/>	46
Objectif 1 <input checked="" type="checkbox"/>	46
Objectif 2 <input checked="" type="checkbox"/>	46
Objectif 3 <input checked="" type="checkbox"/>	46
Objectif 4 <input checked="" type="checkbox"/>	46
Space Invaders.....	47
Étape 0 <input checked="" type="checkbox"/>	47
Vaisseau <input checked="" type="checkbox"/>	47
Objectif 1 <input checked="" type="checkbox"/>	47
Objectif 2 <input checked="" type="checkbox"/>	47
Objectif 3 <input checked="" type="checkbox"/>	47
Alien <input checked="" type="checkbox"/>	47
Objectif 1 <input checked="" type="checkbox"/>	47
Objectif 2 <input checked="" type="checkbox"/>	47
Objectif 3 <input checked="" type="checkbox"/>	47
Objectif 4 <input checked="" type="checkbox"/>	47
Objectif 5 <input checked="" type="checkbox"/>	47
Temps bonus <input checked="" type="checkbox"/>	48
Objectif 1 <input checked="" type="checkbox"/>	48
Objectif 2 <input checked="" type="checkbox"/>	48
Historique.....	49
[ Semaine 1 - 26/01/2015 ].....	49
[ Semaine 3 - 09/02/2015 ].....	49
[ Semaine 4 - 16/02/2015 ].....	49
[ Semaine 5 - 23/02/2015 ].....	49
[ Semaine 6 - 02/03/2015 ].....	49
[ Semaine 7 - 09/03/2015 ].....	49
[ Semaine 8 - 16/03/2015 ].....	49
[ Semaine 9 - 23/03/2015 ].....	49
[ Semaine 10 - 30/03/2015 ].....	50
[ Semaine 11 - 06/04/2015 ].....	50

[ Semaine 12 - 13/04/2015 ].....	50
[ Semaine 13 - 20/04/2015 ].....	50
[ Semaine 13 - 27/04/2015 ].....	50
[ Semaine 14 - 04/05/2015 ].....	50
[ Semaine 15 - 11/05/2015 ].....	50
[ Semaine 16 - 18/05/2015 ].....	50
[ Semaine 17 - 25/05/2015 ].....	51
Problèmes rencontrés.....	52
Pièces non-disponible.....	52
Problème.....	52
Solution.....	52
Certains jeux rament.....	52
Problème.....	52
Solution.....	52
Pas d'écran.....	52
Problème.....	52
Solution.....	52
Booster peu puissant.....	53
Problème.....	53
Solution.....	53
Batterie morte.....	53
Problème.....	53
Solution.....	53
Peu de mémoire.....	53
Problème.....	53
Solution.....	53
Conclusion.....	54
Bibliographie.....	55
Ressources Internet.....	55
Software.....	55
Hardware.....	56

## **Remerciements**

*Je tiens à remercier premièrement M. Carrera qui m'a autorisé à développer ce TFE. Il m'a appris à ne jamais abandonner.*

*Je tiens également à remercier mon entourage. Comme mon papa, ma maman ou encore mon frère qui m'ont aidé à la réalisation matérielle du TFE.*

*Et bien évidemment merci aux autres, avec leurs petites idées, leurs précieux conseils, ...*

## **Introduction**

### ***GameToy***

La **GameToy** est une console inspirée de l'originale **GameBoy**. Elle permet la lecture par USB de jeux externes pouvant être développés par d'autres développeurs en python, utilisant le module de jeu, **PyGame**, ainsi que le module de communication avec la carte électronique, **GPIO**. Les développeurs doivent cependant respecter une certaine norme.

La programmation, l'électronique et le bricolage sont mes plus grandes passions et peut-être aussi un peu le jeu. En essayant de trouver une idée en mixant les 4, j'en suis arrivé à vouloir fabriquer une console me permettant de jouer à mes propres jeux.

### ***LineRun***

**LineRun** est un jeu d'obstacle qui se joue grâce aux touches de la **GameToy**. On contrôle un point qui suit un trajet et qui doit zigzaguer entre deux lignes pour éviter des zones de danger.

Ce jeu m'inspire tant par sa fluidité que par sa complexité à le maîtriser. Il est inspiré du jeu original **WaveRun**.

### ***Space Invaders***

**Space Invaders** est un jeu de shoot qui se joue grâce aux touches de la **GameToy**. Tout se joue sur un plan dans l'espace où le but est d'anéantir les vaisseaux ennemis apparaissant à l'écran. Chaque ennemi abattu rapporte un certain nombre de points.

Ce jeu représente la base des jeux vidéos. Il est sortit en 1978 et reste l'un des jeux les plus connus, je l'ai un peu modifié à ma sauce pour en faire quelque chose d'un peu plus original.

### ***Scénario principal***

#### ***GameToy***

L'utilisateur doit utiliser une clé **USB** qu'il aura renommé auparavant (« **GAMETOY** »).

L'utilisateur allume ensuite la console grâce à un bouton situé sur le bas de la console. Après le chargement de ses composants (**Linux**),

un programme se lance automatiquement et fait apparaître les jeux contenu sur la clé **USB**.

Un menu composé d'une liste déroulante apparaît. L'utilisateur utilise les touches « **HAUT** » et « **BAS** » pour se déplacer dans la liste déroulante pour sélectionner le jeu voulu en appuyant sur la touche « **A** ».

### ***LineRun***

Le jeu débute sur un menu où l'utilisateur choisit s'il veut jouer, afficher les records, accéder aux options ou quitter le jeu. L'utilisateur se déplace dans le menu grâce aux touches fléchées.

#### ***Jeu***

S'il sélectionne le jeu, il doit déplacer un point sur deux lignes. En appuyant sur le bouton « **A** », il fait déplacer le point sur l'autre ligne et ainsi de suite.

Des obstacles d'une certaine longueur apparaissent aléatoirement sur les deux lignes.

S'il touche un obstacle, le joueur perd et le score final est affiché. Le score augmente de manière linéaire et continue tout au long de la partie. En appuyant sur « **SELECT** », si la partie est terminée, le joueur retourne au menu principal.

Le joueur peut quitter le jeu à tout moment en appuyant sur « **SELECT** » et le mettre en pause en appuyant sur « **START** ».

#### ***Records***

Les records s'affichent sous forme de trois catégories. Ces dernières représentent le niveau de difficulté.

#### ***Options***

Le joueur peut changer certains paramètres du jeu comme la gestion de la musique ou des bruitages. Il peut aussi changer la difficulté du jeu ce qui influencera la vitesse de déplacement du point.

Pour choisir l'option souhaitée, il utilise les touches directionnelles. Pour changer la valeur de l'option sélectionnée, il doit appuyer sur le bouton « **A** ».

***Quitter***

L'application s'arrête et retourne au menu principal de la **GameToy**.

***Space Invaders***

Le jeu débute sur un menu où l'utilisateur choisit s'il veut jouer, afficher les records où accéder aux options. L'utilisateur se déplace dans le menu grâce aux touches fléchées.

***Jeu***

S'il sélectionne le jeu, il doit déplacer son vaisseau sur l'axe des abscisses grâce aux touches directionnelles « **GAUCHE** » et « **DROITE** ». En appuyant sur la touche « **A** », le vaisseau effectue un tir.

Le vaisseau possède 140 points de vie et chaque tir qu'il effectue enlève 20 points de vie à un alien lorsqu'il le touche.

Des aliens apparaissent et tentent de tuer le vaisseau mais en suivant un trajet complètement aléatoire. Ils changent de direction lorsqu'ils entrent en collision avec le bord du plateau ou avec un autre alien. Ils tirent tous en même temps toutes les secondes.

Chaque alien possède 100 points de vie mais leur dégât de tir est progressif tout au long du jeu.

Le jeu est composé de deux phases qui se répètent à l'infini :

1. Les aliens arrivent par groupes. D'abord par 1, puis par 2, puis par 3.
2. Une fois après avoir vaincu le groupe de 3 aliens, une pause s'intercale dans le jeu, permettant au vaisseau de récupérer de la vie en ramassant les caisses de soin. Il doit cependant éviter, en même temps, les astéroïdes qui passent. Chaque astéroïde enlève 20 point de vie et chaque caisse de soin en rend également 20.

À la fin de la 2ème phase de jeu, les caractéristiques des aliens changent : ils sont plus puissant et donnent plus de point.

Au début de la partie, les aliens font des dommages de 10 points de vie, mais au fur et à mesure des fins des phases de jeu, elle augmente à chaque fois de 10.

Pour le score, ils donnent de base 10 points. Leur valeur de score augmente de 5 chaque fois qu'un nouveau groupe apparaît.

Si la vie du vaisseau tombe à 0, la partie se termine et le score final s'affiche. En appuyant sur « **SELECT** », si la partie est terminée, le joueur retourne au menu principal.

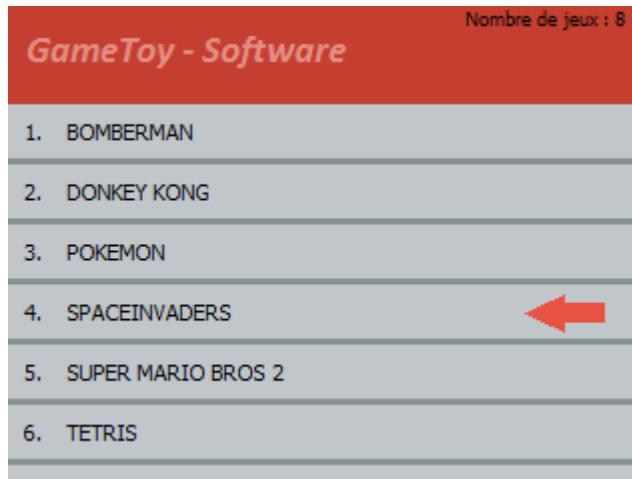
Le joueur peut quitter le jeu à tout moment en appuyant sur « **SELECT** », et le mettre en pause en appuyant sur « **START** ».

***Quitter***

L'application s'arrête et retourne au menu principal de la **GameToy**.

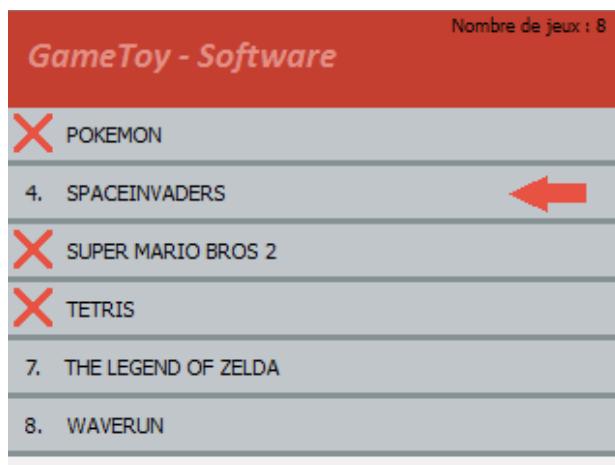
## Manuel d'utilisation

### ***GameToy***



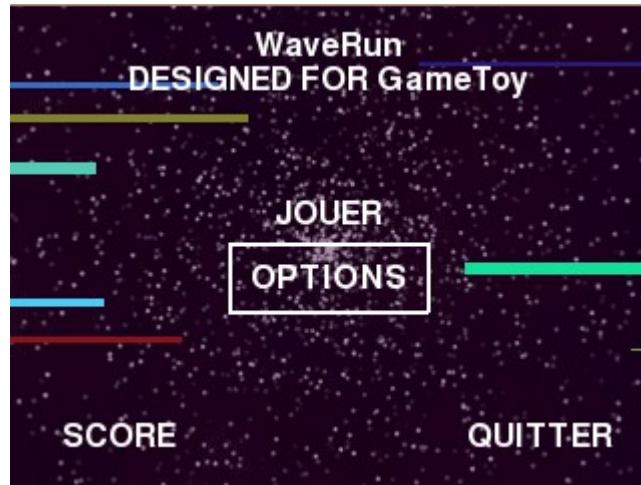
L'utilisateur choisit le jeu désiré grâce aux boutons « HAUT » et « BAS ». La flèche indique la sélection du jeu.

Si le programme détecte des jeux qui ne sont pas valides d'utilisation, il affiche une croix à la place de la numérotation :



## ***LineRun***

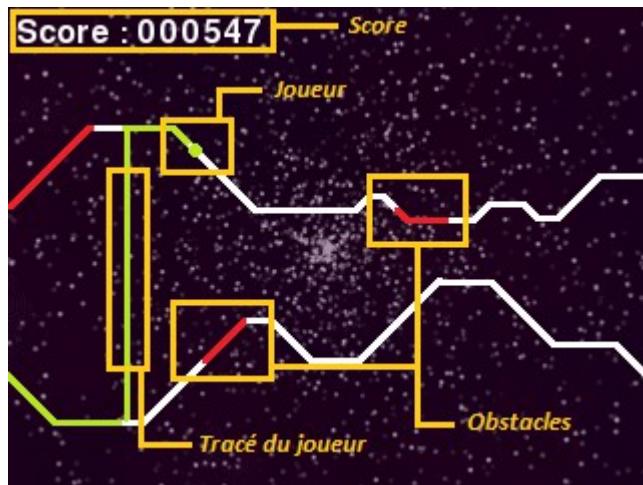
### **Menu principal**



L'utilisateur se déplace dans le menu grâce aux touches fléchées. Les 4 choix possibles sont :

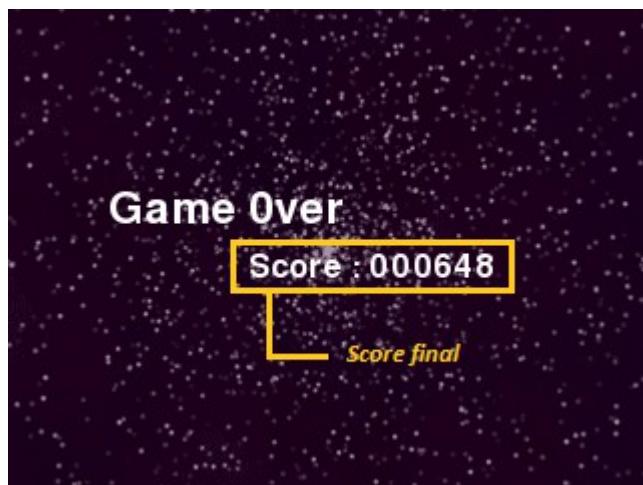
- Jouer (Lancer une partie)
- Options (Accéder aux options du jeu)
- Score (Afficher les records du jeu)
- Quitter (Quitter le jeu)

## Jouer



L'utilisateur a lancé une partie, le voici avec plusieurs éléments :

- Le point vert représente la position actuelle du joueur
- Les tracés blancs sont les chemins sur lesquels le joueur doit se déplacer
- Les tracés rouges sont les obstacles que je le joueur doit éviter
- Les tracés verts sont les tracé du joueur tout le long de son parcours



Si le joueur heurte un tracé rouge (obstacle), la partie se termine directement. L'écran « Game Over » s'affiche et montre le score final du joueur.

## Options



L'utilisateur à sélectionné les paramètres. Il a ici 3 paramètres modifiables :

- Le niveau de difficulté
- L'activation/désactivation de la musique
- L'activation/désactivation des bruitages

Il doit se déplacer dans ce menu annexe pour sélectionner l'option voulue et appuyez sur « A » pour en changer la valeur.

La difficulté se caractérise par 3 niveaux :

- Facile (Vitesse du joueur lente et parcours non-croisé)
- Moyen (Vitesse du joueur lente et parcours croisé)
- Difficile (Vitesse du joueur accélérée et parcours croisé)

## Records



L'utilisateur a sélectionné les records. Ils sont affichés sous forme de 3 catégories représentant les 3 niveaux de difficulté.

Il peut aussi remettre les records à zéro en sélectionnant le bouton « **RESET** » et en appuyant sur « **A** » pour confirmer.

## ***Space Invaders***

### **Menu principal**



L'utilisateur se déplace dans le menu grâce aux touches fléchées. Les 2 choix possibles sont :

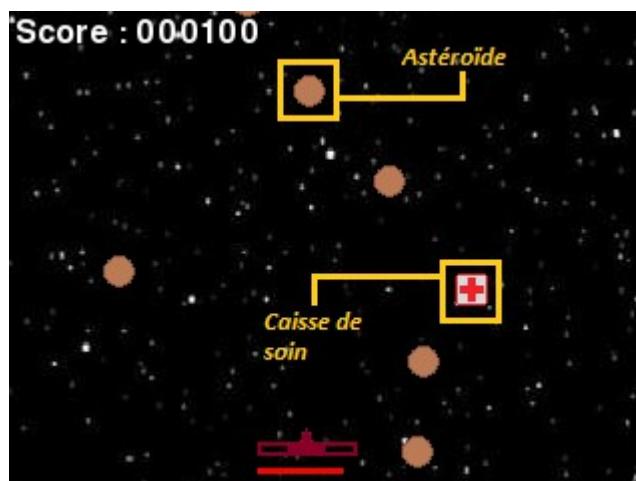
- Jouer (Lancer une partie)
- Quitter (Quitter le jeu)

### **Jouer**

Ci-dessous, la première phase de jeu, celle où le vaisseau doit tuer les 3 groupes d'aliens. Cette image représente un groupe de deux aliens :



Si dessous, la deuxième phase de jeu, celle où le vaisseau peut récupérer de la vie tout en essayant d'éviter les astéroïdes :



# Développement

## Méthode de développement

### GameToy

#### Composants

La *GameToy* est créée grâce à plusieurs éléments chacun aussi important que les autres. Mes choix des composants sont inspirés du projet présent sur le site [www.adafruit.com](http://www.adafruit.com).

#### Raspberry PI Model B+

La **Raspberry PI** est un nano-ordinateur permettant de réaliser de petits projets sans passer par l'achat de grosse carte électronique. Son choix pour le modèle B+ est basé sur plusieurs avantages :

- Peu coûteuse (+- 40€)
- Tourne sous le système d'exploitation **Linux** et est donc facilement adaptable à mes besoins. Dans ce cas-ci, c'est l'utilisation facile et rapide de Python qui est intéressante
- Taille d'une carte de crédit
- Dispose de plusieurs ports **USB** pour la lecture des jeux
- Puissance correcte en proportion à mes besoins
  - **CPU : 700 MHz ARM1176JZF-S core (ARM11)**
  - **RAM : 512 Mo**
- Tourne sous **5-6v** continu.



### Écran TFT touchscreen

Pour l'utilisation d'une interface graphique, la carte à besoin d'un écran directement relié à celle-ci. Dans ce cas, j'ai choisis un écran de petite taille (3,2 pouces) ainsi qu'une basse résolution (320x240) mais qui à l'avantage d'être tactile, ce qui offre la possibilité de réaliser des jeux utilisant la souris (Avec le doigt, évidemment).

Cet écran coûte dans les alentours de 40€.



### Bouton poussoirs

Pour une jouabilité plus agréable et plus réaliste, des boutons de jeux sont indispensables !

Ces boutons poussoirs ne coûtent presque rien.



### Batterie Li-po 1S

Pour alimenter tous les composants nécessaires à la réalisation de la **GameToy**, il faut une batterie pour ne pas devoir être branché sur secteur tout le temps. C'est grâce à elle que nous avons un avantage de portabilité.

Celle-ci fonctionne sous **3,7v (1s)** continu et offre une capacité de **1300mAh**. Et comme toute batterie, elle est rechargeable.

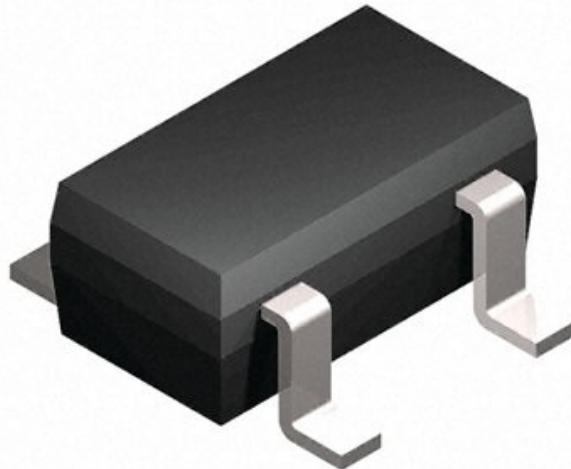
Cette batterie coûte dans les alentours de **15€**.



### Chargeur de batteries Li-po

Vu que la batterie est rechargeable, il faut penser à un système de recharge. Le plus simple et le plus accessible reste un module de recharge car pour recharger une batterie Li-po, il faut passer par une tension bien précise pour ne pas endommager la batterie, dans ce cas (pour une batterie 1s) il nous faut précisément une sortie de **4,2v** qui seront générée à partir d'un câble USB.

Ce module de recharge coûte dans les alentours de 10€.



### Booster de tension

Cet élément est intéressant car il est capable de transformer un courant continu de minimum **3v** à un courant continu de **5,2v**. Il s'emboîte donc parfaitement avec la batterie qui tourne donc en **3,7v** continu.

Il possède également un indicateur de batterie faible lorsque le courant passe en dessous des **3,2v** continu.

Ce module d'augmentation de courant coûte dans les alentours de 10€.



**Push Button ON/OFF**

Pour un effet encore plus réalise, j'ai décidé d'ajouter un bouton de type **push** à la console. Ce bouton est simple, il permet juste de passer entre l'état 0 et l'état 1 et d'y rester avant que l'utilisateur n'appuie à nouveau dessus.

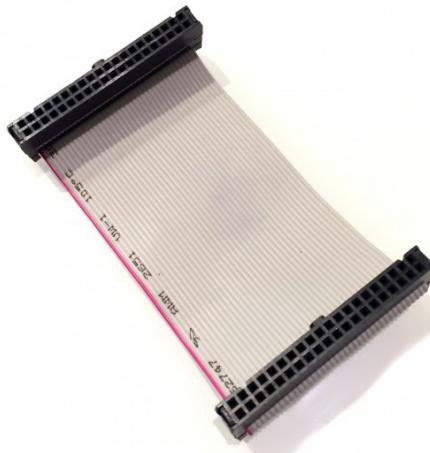
Ce bouton coûte dans les alentours de 2€.

**Coque**

Pour loger tous les éléments, il faut une petite coque. Dans mon cas, j'ai opté pour une boîte de **PLA** (plastique) de 10,5cm x 19cmx 3cm pour bien avoir la place pour les composants et pour aussi éviter toute surchauffe.

### Câble Ribbon

Pour avoir une communication propre avec les boutons, l'utilisation d'un câble **Ribbon** était très intéressante. En effet, il correspond bien avec les **PIN** de sortie de la carte électronique.



### Outils

Pour la bonne création de la console, j'ai du me mettre à disposition plusieurs outils.

#### Fer à souder

Pour la liaison de certains composants, j'ai du créer une soudure assez propre pour que les éléments restent solide entre eux.



### Gaine thermorétractable

Une méthode intéressante pour la soudure est sa couverture de gaine thermorétractable. Cette dernière permet d'isoler la soudure d'un morceau de plastique que l'on fait chauffer pour que celui-ci s'adapte parfaitement à l'allongement du câble soudé.



### **Logiciel de lancement**

Lorsque la console démarre, elle lance directement un programme qui permet de sélectionner un jeu, écrit sur la mémoire de la clé **USB** précédemment insérée, grâce à un logiciel développé par moi-même.

Le logiciel se charge d'analyser tout un dossier prédéfini (**gametoy\_games**) se trouvant sur la clé **USB** pour rechercher tous les fichiers **main.py** qui, par convention définie, sont les fichiers de base sur lesquels le jeu développé doit se lancer.

Le fichier **main.py** doit également contenir la classe **Master**, classe qui se chargera de lancer le jeu. C'est l'utilisateur qui décide de comment développer cette classe.

Après avoir récupéré les jeux, une interface graphique tournant avec la librairie **PyQT** affiche le nom du jeu.

Le logiciel est développé sur base d'un système d'import et de dé-import de modules. Il se présente sous la forme de deux classes, une première pour la partie traitement système, et l'autre pour la partie interface graphique.

#### **Classe**

#### **System**

##### **\_\_init\_\_(self)**

- Défini le chemin absolu du dossier où les jeux se situent.
- Récupère les modules de base nécessaire au fonctionnement principal du programme grâce au module **sys** : **sys.modules.keys()**.

##### **check\_existence(self)**

- Test l'existence du chemin absolu du dossier où les jeux se situent.
- Si une erreur survient, appel immédiatement la méthode **error** avec un **perm\_exit** à **True**.

##### **say(self, message, begin='>>')**

- Affiche un message sous forme de console commençant par **begin**.
- Cette méthode n'est utilisée que lorsque le programme est en mode console.

```
Error(self, code_error, perm_exit=False)
```

- Affiche une erreur précise grâce au numéro contenu dans `code_error`.
- Si `perm_exit` est définie à `True`, le programme s'arrête immédiatement.
- Cette méthode n'est utilisée que lorsque le programme est en mode console.

```
get_game_folder(self)
```

- Récupère tout les jeux se trouvant dans le dossier réservé aux jeux et respectant les règles de lancement :
  1. Un fichier `main.py` doit exister.
  2. Le fichier `main.py` doit posséder une classe `Master` car c'est celle qui sera utilisé pour démarrer le programme.

```
select_game(self)
```

- Propose à l'utilisateur un menu en console qui affiche les jeux respectant les règles de lancement.
- Ajoute les modules nécessaires au bon fonctionnement du jeu au `path` du programme.
  - Le `path` du programme est une variable globale qui contient tous les fichiers liés à l'utilisation du programme. On y retrouve éventuellement les librairies externes, les images, etc.
- Appelle la méthode `launch_game`.

```
launch_game(self)
```

- Importe le fichier main du jeu et toutes ses dépendances.
- Initialise le module `pygame` et lance le programme grâce à l'appel de la classe `Master`.
- Une fois que l'utilisateur quitte le jeu, la méthode quitte le module `pygame` pour décharger le programme des données graphiques utilisées.
- Appelle la méthode `refresh_modules`.

```
refresh_modules(self)
```

- Récupère la nouvelle liste des modules. Celle-ci a changé depuis

le lancement du programme car elle a chargé tous les modules et dépendances nécessaire au bon fonctionnement du jeu.

- Grâce à l'attribut ayant stocké les modules de base, en crée une nouvelle liste ne contenant que les modules importés lors du chargement du jeu.
- Avec la nouvelles liste contenant les nouveaux modules importé, le programme la parcourt et, pour chaque module, le supprime grâce à la fonction `delete_module`.

### **Graphical**

#### `__init__(self, system)`

- Se charge de définir plusieurs attributs de la future interface graphique :
  - Sa résolution (320x240)
  - Le titre de sa fenêtre
- Elle appelle la méthode `initUI` pour initialiser les widgets visuels.
- Elle se charge d'afficher la fenêtre pour que celle-ci soit visible au yeux de l'utilisateur.
- La variable `system` envoyée contient l'autre classe `System` permettant la gestion des fichiers/dossiers ainsi que le lancement du jeu sélectionné.
- Elle contient toutes les sorties `GPIO`, pour l'utilisation des boutons, sous forme d'un dictionnaire. À chaque nom de touche correspond son numéro de sortie.

#### `connect_inputs(self)`

- Récupère toutes les sorties `GPIO` pour ajouter un `trigger`. Lorsque l'on appuie sur un bouton, `PyQt` se charge directement d'appeler une fonction définie. Dans ce cas-ci, c'est la fonction `callback` qui sera appelée.

#### `disconnect_inputs(self)`

- Récupère toutes les sorties `GPIO` pour enlever le `trigger` précédemment ajouté.

#### `find_channel_name(self)`

- Lorsque l'on appuie sur un bouton, cette fonction est appelée

pour retrouver le véritable nom de la sortie **GPIO** utilisée.

**callback(self, channel)**

- **PyQt** récupère automatiquement l'appui sur un bouton, il se charge de récupérer le numéro de la sortie **GPIO**.
- Dans ce cas-ci elle permet de définir si l'utilisateur veut monter dans le menu, descendre ou encore, de lancer un jeu.
- Elle appelle la méthode **find\_channel\_name** pour connaître le nom de la sortie **GPIO** utilisée.

**keyPressEvent(self, e)**

- Permet de récupérer et de tester si l'utilisateur appuie sur l'une des touches du clavier.
- Cette partie fait exactement la même chose que **callback** mais avec les touches du clavier.

**refresh\_menu(self)**

- Permet de rafraîchir la position du sélecteur pour ne pas que celui-ci « sorte » de l'écran et qu'il permette le défilement de la liste des jeux si celle-ci est supérieur à 6.
- Elle appelle la méthode **refresh\_games**.

**refresh\_game(self)**

- Est utilisée pour redéfinir la position des jeux à l'écran.

**initUI(self)**

- Cette méthode contient toutes les initialisations des widgets permettant un bon effet visuel. On y retrouve des **labels** (Qui peuvent contenir soit du texte, soit une image).

## Fonctions externes

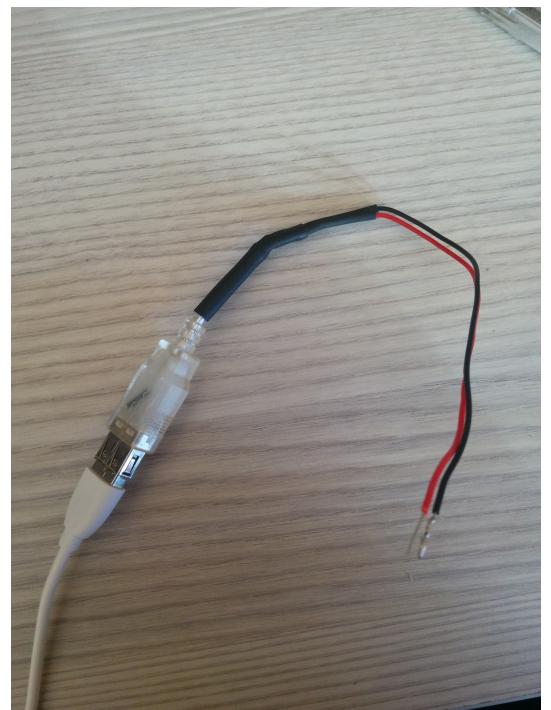
**delete\_module(modname)**

- Fonction trouvée sur internet permettant de retirer de la mémoire courante, un module précédemment importé.

### **Conception de la console**

#### **Rechargement / alimentation**

J'ai décidé de mettre à disposition de l'utilisateur un système de rechargement et d'alimentation simple. Cela se traduit par une entrée **USB** (femelle) qui s'occupera d'aller alimenter la carte et de recharger la batterie si celle-ci en a besoin.



#### **Bouton d'allumage**

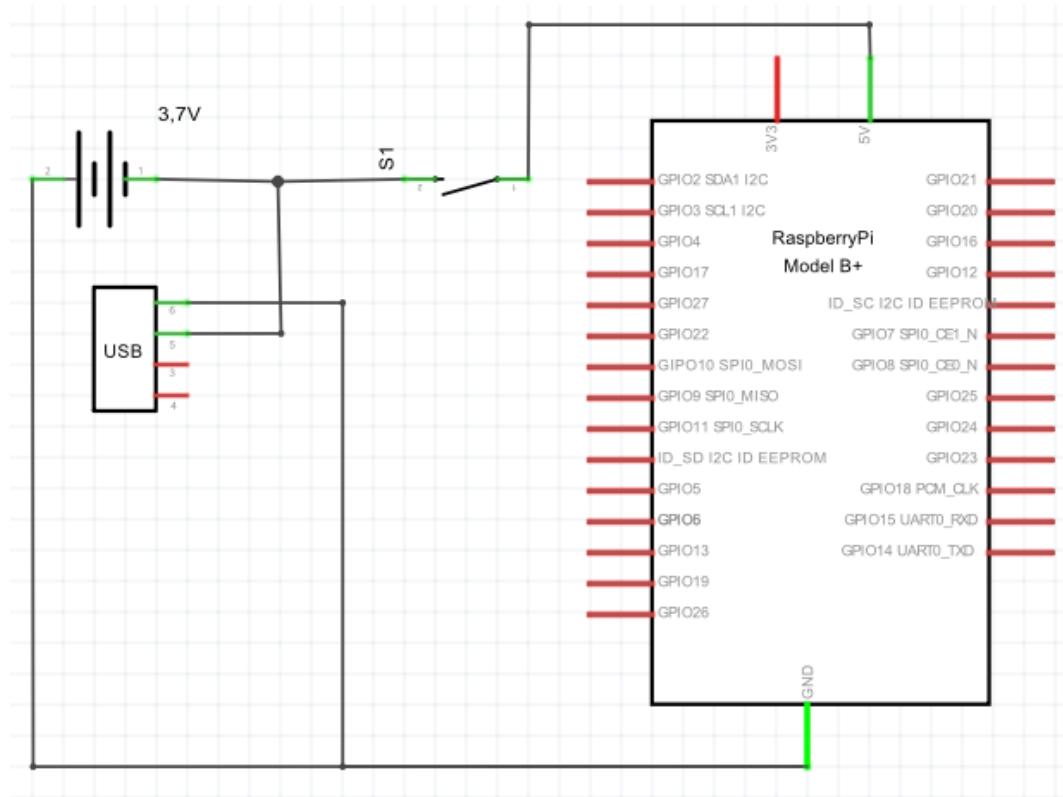
J'ai positionné le bouton d'allumage juste après l'entrée d'alimentation et juste après la batterie.

L'allumage de la batterie se résume par une équation simple :

```
(Batterie || USB) && Button_ON = 1
```

Si l'on a une entrée d'alimentation (soit batterie, soit par USB) et que le bouton d'allumage est à l'état 1, alors la console s'allume.

Voici la partie du schéma électrique correspondante :



### Utilisation des boutons poussoirs

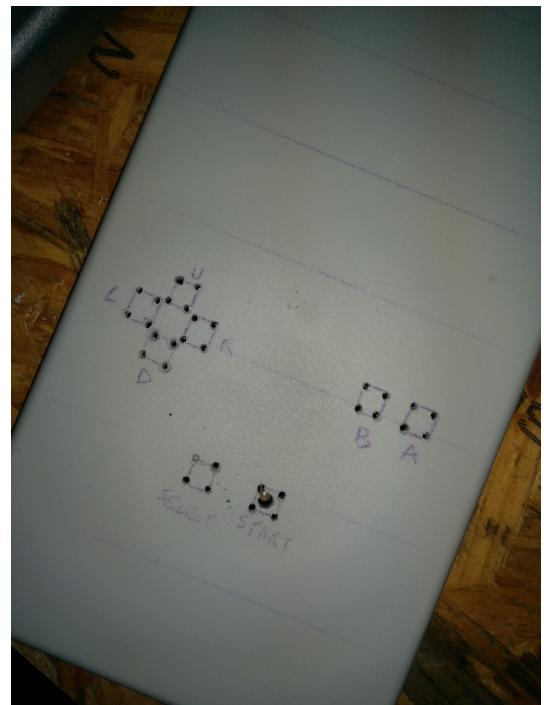
Pour respecter une certaine convention, la manette comporte 8 boutons poussoirs.

- 4 utilisés pour la direction (**HAUT**, **BAS**, **GAUCHE**, **DROITE**)
- 2 utilisés pour les actions (**A**, **B**)
- 2 utilisés pour les options (**START**, **SELECT**)

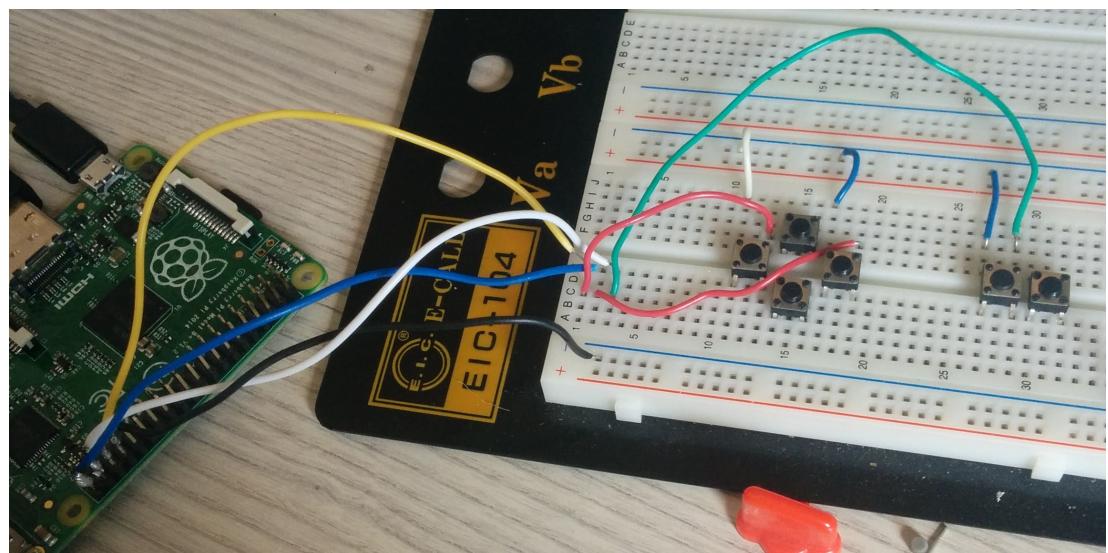
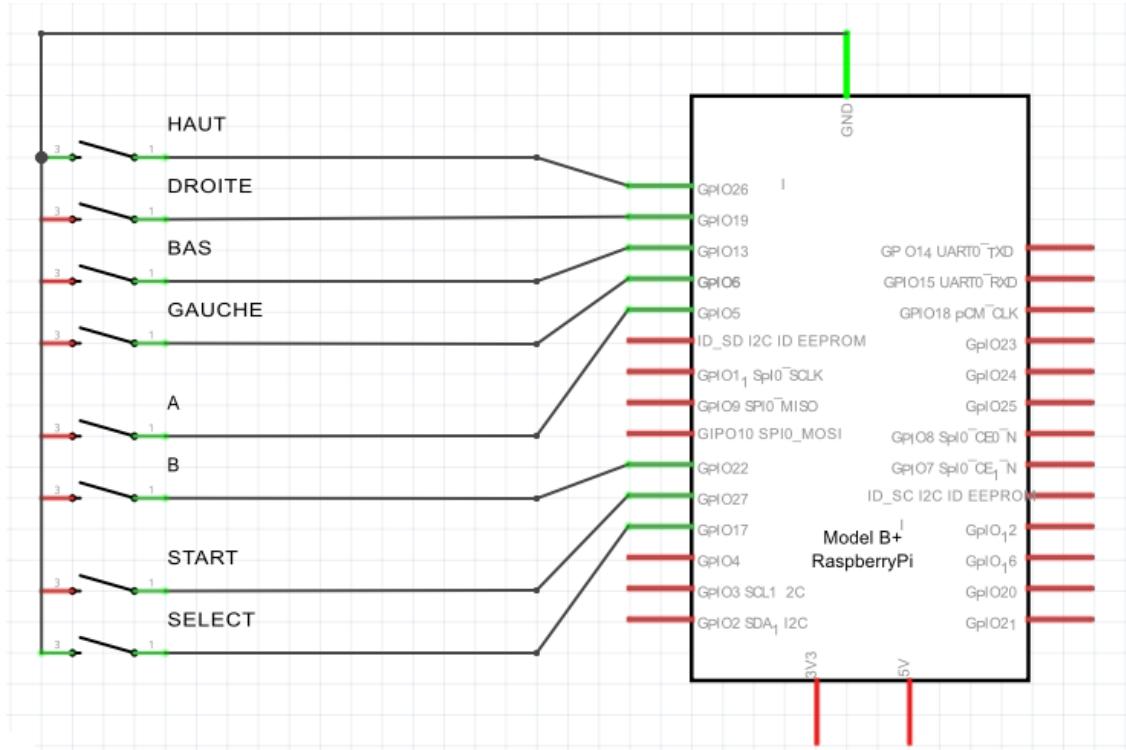
Pour reproduire la réalité, je les ai positionné comme sur la vraie GameBoy :



Pour qu'il tienne sur la coque, j'ai effectué 4 trous pour chaque bouton pour laisser passer les pattes. Je n'avais ensuite qu'à faire passer les câbles par les trous pour les souder. La fixation finale est effectuée grâce à un simple point de colle en dessous du bouton, sur la coque.



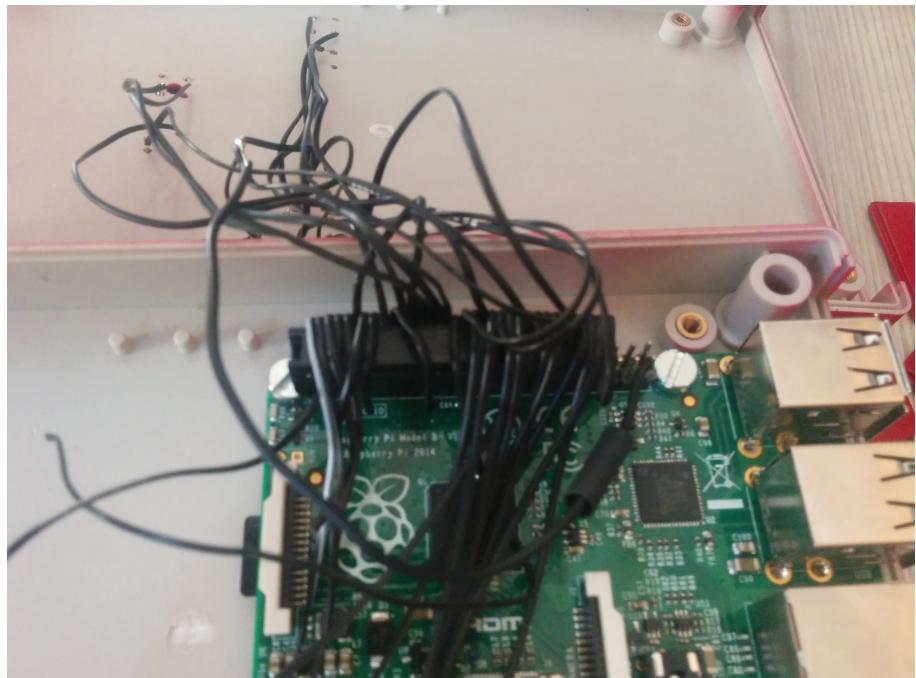
Voici la partie du schéma électrique correspondante :



La partie la plus compliquée à été la soudure des boutons avec la carte. Pour avoir un système de communication propre avec la carte, l'utilisation d'un câble **Ribbon**, utilisé normalement pour la communication avec les plus anciens disques durs, m'a été très pratique.

En effet, les **PIN** sont disposés sur la carte de la même manière que sur un disque dur. Il ma donc suffit de séparer chaque filament de ce câble pour les répartir vers les boutons correspondant.

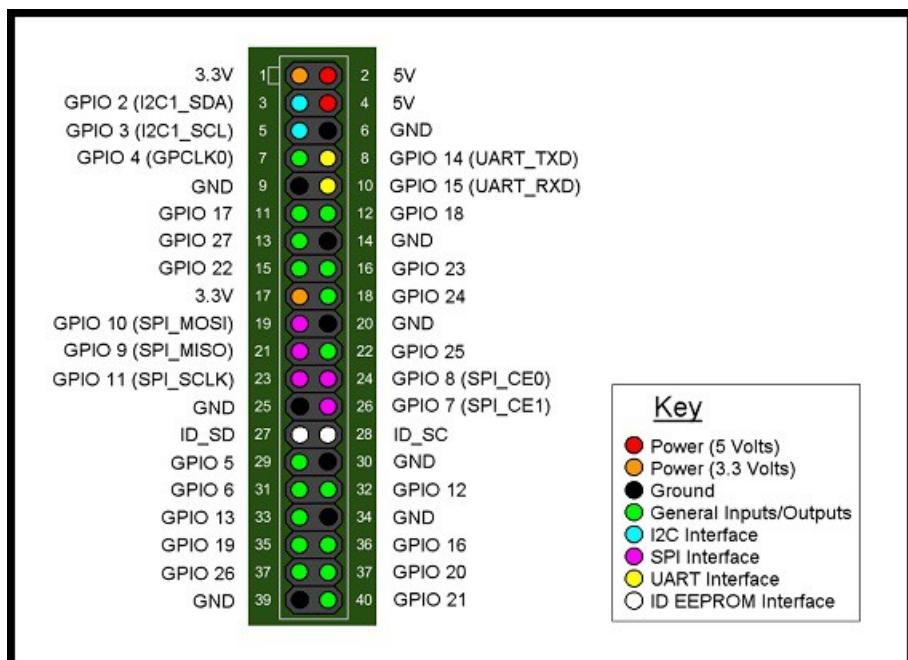
Cette partie est la plus compliquée et aussi la plus longue car il faut bien faire attention à ne pas mélanger les câbles entre eux, cela pourrait provoquer un court circuit. Chaque câble correspond à son PIN de la carte. Ce qui nous fait 2 câbles par boutons, multipliés par 8 boutons, nous avons donc un total de 16 câbles rien que pour la communication.



### Communiquer avec la carte

Pour que les boutons puissent communiquer avec la carte électronique et avoir un impact sur le jeu lancé, il faut utiliser un module python disponible par défaut sur la carte électronique : **GPIO**.

Ce module permet la liaison entre les sorties de la carte électronique (les **PIN GPIO**) et d'autres composants. L'idée ici est de tester si un courant passe entre deux PIN définie. Voici un schéma expliquant les sorties de la carte :



Ci-dessous, la liste des boutons avec les sorties **PIN GPIO** correspondante utilisé pour la **GameToy** :

<i>Nom du bouton</i>	<i>Numéro de PIN</i>	<i>Numéro GND</i>	<i>Numéro GPIO</i>
A	11	9	GPIO17
B	12	6	GPIO18
START	13	14	GPIO27
SELECT	15	20	GPIO22
BAS	16	25	GPIO23
HAUT	18	30	GPIO24
GAUCHE	22	34	GPIO25
DROITE	40	39	GPIO21

Par exemple, pour utiliser le bouton « A », je vais tester si un courant existe entre le **GPIO17** (Ou une autre sortie **GPIO**) qui équivaut au **PIN 11** et le **GND9** (cf. schéma électrique ci-dessus).

Coté électronique, il n'y a qu'à relier un bouton entre les 2 **PIN** par deux câbles pour avoir un peu de longueur pour l'utiliser.

Coté programmation, cela reste également très simple. Il faut d'abord initialiser le module de sorties, c'est-à-dire dire à la carte électronique que l'on veut utiliser ses sorties :

**GPIO.setmode(GPIO.BCM)**

**Attention, pour utiliser les PIN de la carte électronique, il faut les droits super-utilisateur !**

Ensuite, il suffit de préciser quel PIN est-ce que l'on veut utiliser et préciser le type de sortie dont on a besoin :

- **GPIO.IN** : spécifie que l'on veut tester si un courant passe.
- **GPIO.OUT** : permet de contrôler la sortie. C'est-à-dire si l'on veut que du courant passe ou ne passe pas par cette sortie. On peut contrôler facilement une **LED** de cette manière.

Dans notre cas, c'est **GPIO.IN** qui va nous être utile :

**GPIO.setup(17, GPIO.IN)**

Le numéro 17 représente le numéro **GPIO** pour le bouton « A ».

Son numéro de **PIN** sur la carte est le 11.

Pour récupérer l'état du bouton, c'est-à-dire si l'utilisateur pousse dessus ou non, il existe la fonction **GPIO.input** qui prend comme paramètre le numéro de la PIN que l'on veut tester :

```
etat_button = GPIO.input(11)
```

Pour donner un exemple concret d'utilisation de ce module très utile, voici un bout de code qui explique comment allumer une LED via un bouton :

```
1. GPIO.setmode(GPIO.BCM)
2.
3. pin_button = 1
4. pin_led = 2
5.
6. GPIO.setup(pin_led, GPIO.OUT)
7. GPIO.setup(pin_button, GPIO.IN)
8.
9. while 1:
10.     if (GPIO.input(pin_button) == False):
11.         GPIO.output(pin_led, HIGH)
12.     elif (GPIO.input(pin_button) == True):
13.         GPIO.output(pin_led, LOW)
```

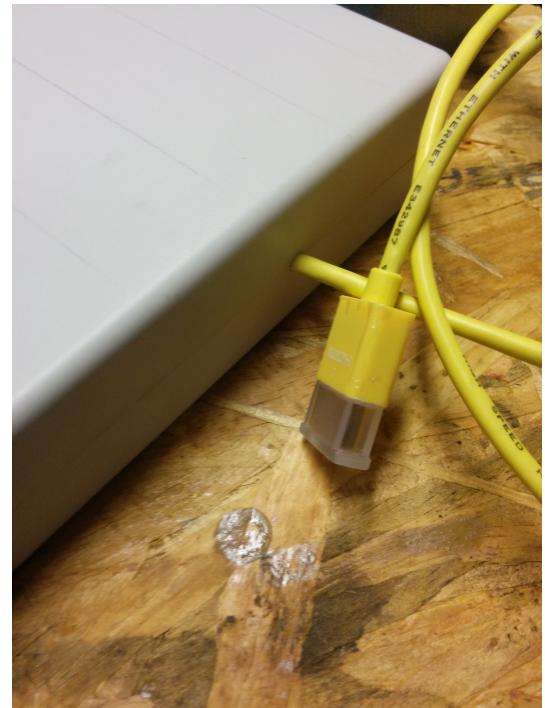
### Fixation de la Raspberry PI

Pour fixer la carte, j'ai simplement fait 4 trous dedans ainsi que dans la coque pour ensuite passer une vis avec un écrou. J'ai décidé de la fixer au bord de la coque pour avoir facilement accès aux sorties **USB** et **ETHERNET**.



### Sortie HDMI

Pour pouvoir directement brancher la console à un écran **HDMI**, j'ai fait un petit trou sur le côté juste au niveau de l'encoche pour laisser passer le câble.



### Mise en couleur de la coque

La sélection de la couleur « Rouge » est simple. C'est une couleur que l'on retrouve rarement dans le domaine de la console. Ici, l'utilisation d'un rouge « brillant » fait penser à l'esprit maquette, ce que je préfère.



## Jeux

Les jeux sont développés sur base d'un squelette, identique pour chacun. Chaque jeu contient 6 grosses parties :

- Le menu (`menu.py`)
- Le jeu (`jeu.py`)
- Les records (`records.py`)
- Les options (`settings.py`)
- Les classes externes (`Dossier - classes.py`)
- Les fonctions externes (`fonctions.py`)
- Un fichier de configuration (`data.txt`)

C'est pour cette raison qu'il existera au minimum 6 fichiers python.

## **PyGame**

Le but de ce TFE n'étant pas de présenter les aspects techniques de l'utilisation de **PyGame**, je ne m'attarderai pas énormément sur l'explication de cette librairie.

**PyGame** est une librairie qui permet le développement aisément d'application contenant des images et du son.

Le type de squelette utilisé pour le développement des jeux tourne de paire avec **PyGame**. Plus bas dans ce rapport, sont expliquées les méthodes contenue dans l'architecture de mes jeux.

**PyGame** fonctionne sous plusieurs phases. Certaines ne sont utilisée qu'une fois, d'autres se répètent :

1. Initialisation graphique :
  - Définition de la taille de la fenêtre,
  - Définition d'une icône, d'un titre ou encore la spécification d'un affichage **FullScreen** ou non. Tout cela encore pour la gestion de la fenêtre,
2. Chargement des composants :
  - Chargement des tous les éléments externes de la programmation,
  - On y retrouve les images, les sons, les vidéos, les polices, etc.

### 3. Boucle principale :

- C'est ici que toutes les actions qui doivent être répétées se trouvent,
- On y retrouve les déplacements des objets (Par exemple, un joueur se déplace à gauche).

### 4. Attente d'événements :

- Cette phase se trouve dans la boucle principale,
- Elle attend que l'utilisateur fasse quelque chose pour interagir avec le programme :
  - Appui sur une touche du clavier,
  - Fermeture de la fenêtre,
  - Mouvement de la souris.

### 5. Rendu graphique

- Cette phase se trouve dans la boucle principale,
- Elle permet le rafraîchissement de la fenêtre (Par exemple, mettre à jour la position d'une image).

## ***Menu, jeu, records et options***

Les jeux respectent chacun la même architecture. Ils sont fait d'une grosse classe qui permet de suivre le même fonctionnement de **PyGame**.

Cette architecture permet aussi au programmeur de mieux s'y retrouver dans son code, tout est décortiqué. C'est donc beaucoup plus facile d'effectuer des modifications, il y a donc un gain de temps.

## **Méthodes de base**

### **\_\_init\_\_**

- Chargement des variables dites « systèmes » qui serviront au bon fonctionnement de la partie du jeu,
- Lancement des méthodes de chargements,
- Lancement de la méthode de boucle principale.

### **on\_load\_item**

- Charge les images nécessaires au bon fonctionnement de la partie du jeu,

- Charge les classes nécessaires au bon fonctionnement de la partie du jeu.
- Charge les musiques nécessaires au bon fonctionnement de la partie du jeu.

**on\_load\_font**

- Crée toutes les polices nécessaires pour l'affichage de divers éléments de la partie du jeu.

**on\_run**

- Appelle la méthode qui gère les événements,
- Applique toutes les modifications qui doivent être effectuées. C'est dans cette partie du programme que l'on retrouve toutes les conditions nécessaires au bon fonctionnement de la partie du jeu,
- Appelle la méthode qui gère le rendu graphique.

**on\_render**

- Calcule toutes les positions des éléments qui doivent être affichés,
- Effectue le rendu de tous les éléments grâce aux positions précédemment calculées.

***Les classes externes***

Ici deux solutions sont possibles.

La première est d'avoir un seul fichier contenant toutes les classes si le jeu n'est pas trop gros.

La deuxième est d'avoir un dossier contenant un fichier par classe externe.

Aucune architecture n'est respectée dans le cadre de développement de classes externes pour la simple et bonne raison que beaucoup d'éléments changent entre chaque classe. Il est donc difficile de trouver des points communs entre ces dernières, et d'en faire un système modulable.

***Les fonctions externes***

Ce fichier contient toutes les fonctions un peu plus grosses qui permettent d'alléger encore un peu plus le programme principal dans sa lecture.

## **Phases de développement**

### **GameToy**

#### **Achat**

##### **Objectif 1**

Acheter l'écran.



##### **Objectif 2**

Acheter la coque.



##### **Objectif 3**

Acheter la batterie.



##### **Objectif 4**

Acheter les boutons.



##### **Objectif 6**

Acheter les modules de rechargement.



#### **Design de la coque**

##### **Objectif 1**

Dessiner et découper dans la coque les emplacements destinés au boutons



##### **Objectif 2**

Dessiner et découper dans la coque l'emplacement destiné à l'écran



##### **Objectif 3**

Dessiner et découper dans la coque les emplacements destinés aux entrées de la carte comme l'alimentation, un câble réseau, un câble USB.



##### **Objectif 4**

Peindre la coque avec une bombe couleur.



**Affichage****Objectif 1**

Relier l'écran à la carte grâce au câble **GPIO**.

**Objectif 2**

Offrir la possibilité de brancher un écran **HDMI**.

**Circuit d'alimentation****Objectif 1**

Relier tous les éléments à la batterie en suivant en schéma électrique bien précis.

**Objectif 2**

Permettre à l'utilisateur d'allumer ou d'éteindre la console grâce à un petit bouton **ON/OFF**.

**Objectif 3**

Mettre en place une petite LED qui permet de dire si la console est allumée.

**Objectif 4**

Permettre le rechargement et l'alimentation de la console par un câble USB Mâle-Mâle relié à un chargeur de smartphone ou à un ordinateur.

**Touches fonctionnelles****Objectif 1**

Soudre les pins des boutons aux bonnes **PIN** de la carte électronique.

**Software****Objectif 1**

Le logiciel se lance automatiquement au démarrage de la console.

**Objectif 2**

L'utilisateur peut se déplacer dans un menu « déroulant » et lancer son jeu.

## LineRun

### Étape 0



Initialisation de `PyGame`, création de l'écran de jeu avec une résolution de 320x240.

### Chemin aléatoire



#### **Objectif 1**



Créer un chemin prenant la première moitié supérieur de l'écran comme dimension.

#### **Objectif 2**



Créer deux chemins qui peuvent se croiser ou non selon la demande.

### Point (Joueur)



#### **Objectif 1**



Faire défiler le point sur l'un des deux chemins selon la demande.

#### **Objectif 2**



Faire changer le point de chemin lorsque le joueur appuie sur la touche « A ».

### Gestion du score



#### **Objectif 1**



À chaque tour de boucle, le joueur gagne un point.

#### **Objectif 2**



Affichage du score à l'écran.

#### **Objectif 3**



Gestion d'un système d'affichage des 5 meilleurs scores, enregistrés dans un fichier.

#### **Objectif 4**



Lorsque le joueur meurt, son score final est affiché au milieu de l'écran.

## Space Invaders

### Étape 0



Initialisation de **PyGame**, création de l'écran de jeu avec une résolution de 320x240.

### Vaisseau



### **Objectif 1**



Le joueur peut déplacer le vaisseau à gauche ou à droite en utilisant les boutons « **GAUCHE** » et « **DROITE** ».

### **Objectif 2**



Lorsque le joueur appuie sur le bouton « **A** », le vaisseau tire en direction des aliens.

### **Objectif 3**



Lorsque le vaisseau n'a plus de vie, effectue une petite animation de mort.

### Alien



### **Objectif 1**



Les aliens se déplacent diagonalement de manière aléatoire.

### **Objectif 2**



Si les aliens entre en collision entre eux, changent de direction.

### **Objectif 3**



Les aliens tir en direction du vaisseau chaque seconde.

### **Objectif 4**



Lorsque l'alien n'a plus de vie, disparaît.

### **Objectif 5**



Apparaissent par groupe de 1, de 2 et puis de 3 avant d'offrir au joueur la possibilité d'avoir un temps bonus.

**Temps bonus**



**Objectif 1**



Des astéroïdes apparaissent aléatoirement et se dirigent vers le vaisseau. Lorsqu'ils entrent en collision avec le vaisseau, lui enlève 10 points de vie.

**Objectif 2**



Des caisses de soin apparaissent aléatoirement et se dirigent vers le vaisseau. Lorsqu'elles entrent en collision avec le vaisseau, lui donne 10 points de vie.

## **Historique**

### **[ Semaine 1 - 26/01/2015 ]**

- Mise en place du cahier de charge et écriture des informations sur le jeu **LineRun**
- Début de la récupération des points communs de développement entre chaque jeux

### **[ Semaine 2 - 02/02/2015 ]**

- Suite du développement du jeu **LineRun**

### **[ Semaine 3 - 09/02/2015 ]**

- Suite du développement du jeu **LineRun**

### **[ Semaine 4 - 16/02/2015 ]**

- Suite du développement du jeu **LineRun**

### **[ Semaine 5 - 23/02/2015 ]**

- Début du développement du jeu **Space Invaders**

### **[ Semaine 6 - 02/03/2015 ]**

- Suite et fin du développement du jeu **LineRun**
- Suite du développement du jeu **Space Invaders**
- Mise en place d'un système de développement commun et léger pour la création des jeux.

### **[ Semaine 7 - 09/03/2015 ]**

- Suite du développement du jeu **Space Invaders**

### **[ Semaine 8 - 16/03/2015 ]**

- Suite du développement du jeu **Space Invaders**

### **[ Semaine 9 - 23/03/2015 ]**

- Mise en place du cahier de charge et écriture des informations sur le jeu **Space Invaders**

**[ Semaine 10 - 30/03/2015 ]**

- Début du développement du programme de la **GameToy**

**[ Semaine 11 - 06/04/2015 ]**

- Réflexion sur le choix des composants et calcul des différentes valeurs

**[ Semaine 12 - 13/04/2015 ]**

- Dessins des schémas électriques
- Réflexion sur le choix des composants et calcul des différentes valeurs

**[ Semaine 13 - 20/04/2015 ]**

- Rédaction du cahier de charge sur la partie de développement de la **GameToy**.
- Commande des composants pour la console
- Recherche d'explications sur le fonctionnement de la batterie

**[ Semaine 13 - 27/04/2015 ]**

- Début du montage des composants et premier tests de comptabilité
- Test et apprentissage du module python **GPIO**.

**[ Semaine 14 - 04/05/2015 ]**

- Création du système d'alimentation

**[ Semaine 15 - 11/05/2015 ]**

- Installation des boutons en rapport avec le **software** et dessin de la boîte
- Fixation de la carte dans la coque

**[ Semaine 16 - 18/05/2015 ]**

- Suite de la rédaction du rapport
- Fin du développement et correction de bugs mineurs sur l'utilisation du software de la **GameToy**

**[ Semaine 17 - 25/05/2015 ]**

- Mise en couleur de la coque
- Placement des boutons et soudure avec la carte
- Tests finaux sur l'utilisation des **softwares**

## **Problèmes rencontrés**

### ***Pièces non-disponible***

#### **Problème**

En voulant commander les pièces sur le site [www.adafruit.com](http://www.adafruit.com). Je me suis rendu compte que le prix était assez élevé et certains composants n'étaient plus disponibles.

#### **Solution**

Il a fallut que je me débrouille pour trouver une alternative, c'est-à-dire :

- Designer un nouveau boîtier
- Rechercher après des équivalences au niveau des composants

### ***Certains jeux sont lent***

#### **Problème**

Certains jeux se mettent à « ramer » lorsqu'ils sont joués sur la GameToy.

#### **Solution**

Aucune solution n'est facilement acceptable. D'une part car la carte limite vraiment beaucoup le développeur au niveau de ses caractéristiques et d'autres part je n'ai pas beaucoup de connaissance en terme d'optimisation de jeux vidéos.

### ***Pas d'écran***

#### **Problème**

Le fournisseur n'a pas su assumer ses délais de livraisons et ne peut me délivrer l'écran de la console en temps et en heure.

#### **Solution**

L'affichage de la console s'effectuera donc via un écran **HDMI**.

## ***Booster peu puissant***

### **Problème**

Le booster que j'ai acheté ne permet pas une assez bonne conversion de la tension. En effet, c'est un composant assez bon marché pour le résultat que cela doit produire.

Le booster ne peut donc pas alimenter la **GameToy**.

### **Solution**

Mettre de coté l'utilisation de la batterie et utiliser un système d'alimentation classique grâce à un câble **USB** Mâle-Mâle qui était sensé recharger la batterie.

## ***Batterie morte***

### **Problème**

La batterie que j'ai acheté a été morte au niveau de son recharge et décharge après 3-4 utilisation. Cela est dû à la basse qualité de la batterie. La batterie que j'ai acheté est à la base prévue pour l'alimentation d'un Coyote (Détecteur de radar), elle est donc prévue pour une utilisation à plus faible consommation.

### **Solution**

Mettre de coté le système d'alimentation par batterie avec le système de recharge. Tout sera géré par alimentation classique par **USB** (5,0v).

## ***Peu de mémoire***

### **Problème**

De temps à autre, après la fermeture d'un jeu, la console plante car je lui demande une trop grande utilisation de **RAM**. Un message d'erreur apparaît « **Segmentation fault** ». D'après mes recherches, aucune méthode ne peut contrer ce problème.

### **Solution**

Il n'y a aucune solution possible si ce n'est que l'achat d'une **Raspberry PI** plus puissante (un modèle plus récent).

## **Conclusion**

Ce travail ne correspond pas complètement à mes attentes de base. En effet, je m'y suis pris un peu tard dans le montage de la console en elle-même. C'est à ce moment que tous les problèmes sont apparus, problème de fournisseur pour recevoir les bonnes pièces, problèmes de communication avec les composants externes, ....

Malgré tous les problèmes, j'ai quand même réussi à finir le projet. Cela m'a appris qu'avec de la motivation et de l'envie, rien ne peut arrêter le développement.

J'ai trouvé ce projet tellement intéressant qu'il m'a permis l'apprentissage de plusieurs choses dans différents domaines. Comme la maîtrise de la soudure, l'utilisation d'outils divers, le dessin de schémas électroniques, ....

J'ai aussi appris, et je pense que c'est la partie la plus importante, à pouvoir compter sur mon entourage. Il est très important, je pense, d'avoir pas mal de connaissances gérant chacun son domaine. J'ai ainsi pu faire appel à des personnes qualifiées en électronique, en développement, en design, etc....

Même si le projet n'est pas à 100% terminé, il me tâte d'avoir un peu de temps pendant les grandes vacances pour le finir. Je n'aime pas rester sur un échec.

# Bibliographie

## **Ressources Internet**

### **Software**

#### **Olivier Pfeiffer,**

- <http://olivierpfeiffer.net/raspberry-pi/>

#### **OpenClassRoom,**

- <http://openclassrooms.com/forum/sujet/reloading-module-et-ses-dependances>

#### **PyGame,**

- <http://www.pygame.org>

#### **PyMOTW,**

- <http://pymotw.com/2/sys/imports.html>

#### **Python,**

- <https://docs.python.org/2/library/functions.html>
- <https://mail.python.org/pipermail/tutor/2006-August/048596.html>
- <https://wiki.python.org/moin/PyQt>

#### **Raspberry PI,**

- <https://www.raspberrypi.org/downloads/>
- <http://store.raspberrypi.com/projects>
- <https://www.raspberrypi.org/learning/quick-reaction-game/worksheet/>
- <https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=111955&p=767433#p767433>

#### **Raspbian,**

- <http://www.raspbian.org/RaspbianInstaller>

#### **SourceForge,**

- <http://pyqt.sourceforge.net/Docs/PyQt4/classes.html>
- <http://sourceforge.net/p/raspberry-gpio-python/wiki/Inputs/>

**StackOverflow,**

- <http://stackoverflow.com/questions/684171/how-to-re-import-an-updated-package-while-in-python-interpreter>
- <http://stackoverflow.com/questions/12152060/how-does-the-keypressevent-method-work-in-this-program>

**Wikibooks,**

- <http://fr.wikibooks.org/wiki/PyQt>

**Hardware****Adrafruit,**

- <https://learn.adafruit.com/pigrrl-raspberry-pi-gameboy/overview>

**Amazon,**

- [http://www.amazon.fr/MB-Raspberry-Pi-Type-512MB/dp/B009RIOXFE/ref=sr\\_1\\_4?ie=UTF8&qid=1429437722&sr=8-4&keywords=raspberry+pi+b](http://www.amazon.fr/MB-Raspberry-Pi-Type-512MB/dp/B009RIOXFE/ref=sr_1_4?ie=UTF8&qid=1429437722&sr=8-4&keywords=raspberry+pi+b)
- [http://www.amazon.fr/Adafruit-PiTFT-Mini-Kit-Touchscreen/dp/B00H9B1DTA/ref=sr\\_1\\_2?ie=UTF8&qid=1429437839&sr=8-2&keywords=raspberry+touchscreen](http://www.amazon.fr/Adafruit-PiTFT-Mini-Kit-Touchscreen/dp/B00H9B1DTA/ref=sr_1_2?ie=UTF8&qid=1429437839&sr=8-2&keywords=raspberry+touchscreen)
- [http://www.amazon.fr/TotalConsole-EG-C7029-SNES-Controller-Tomee/dp/B00AZJWSII/ref=sr\\_1\\_7?ie=UTF8&qid=1429437882&sr=8-7&keywords=snes+controller](http://www.amazon.fr/TotalConsole-EG-C7029-SNES-Controller-Tomee/dp/B00AZJWSII/ref=sr_1_7?ie=UTF8&qid=1429437882&sr=8-7&keywords=snes+controller)
- [http://www.amazon.fr/Expansion-chemin-plat-C%C3%A2ble-Raspberry/dp/B00LQD9ES4/ref=sr\\_1\\_4?ie=UTF8&qid=1429437962&sr=8-4&keywords=GPIO+Ribbon+Cable+for+Raspberry+Pi](http://www.amazon.fr/Expansion-chemin-plat-C%C3%A2ble-Raspberry/dp/B00LQD9ES4/ref=sr_1_4?ie=UTF8&qid=1429437962&sr=8-4&keywords=GPIO+Ribbon+Cable+for+Raspberry+Pi)
- [http://www.amazon.fr/Adafruit-Mini-Lipo-Mini-B-Jack/dp/B00SK8J94M/ref=sr\\_1\\_1?ie=UTF8&qid=1429438093&sr=8-1&keywords=mini+lipo+adafruit](http://www.amazon.fr/Adafruit-Mini-Lipo-Mini-B-Jack/dp/B00SK8J94M/ref=sr_1_1?ie=UTF8&qid=1429438093&sr=8-1&keywords=mini+lipo+adafruit)
- [http://www.amazon.fr/PowerBoost-500-Basic-Boost-500mA/dp/B00PY2YRL2/ref=sr\\_1\\_1?ie=UTF8&qid=1429438155&sr=8-1&keywords=powerboost+500+adafruit](http://www.amazon.fr/PowerBoost-500-Basic-Boost-500mA/dp/B00PY2YRL2/ref=sr_1_1?ie=UTF8&qid=1429438155&sr=8-1&keywords=powerboost+500+adafruit)

**Artesyn,**

- [https://www.artesyn.com/power/power-supplies/product-docs/168/ldo03c/an\\_ldo03c\\_ldo06c\\_ldo10c\\_1298442035\\_appnote.pdf](https://www.artesyn.com/power/power-supplies/product-docs/168/ldo03c/an_ldo03c_ldo06c_ldo10c_1298442035_appnote.pdf)

**Ebay,**

- [http://www.befr.ebay.be/item/3-5-inch-Display-Touch-Screen-TFT-LCD-for-Raspberry-Pi-2-B-RP02007-/271834567052?pt=LH\\_DefaultDomain\\_77&hash=item3f4a9a518c](http://www.befr.ebay.be/item/3-5-inch-Display-Touch-Screen-TFT-LCD-for-Raspberry-Pi-2-B-RP02007-/271834567052?pt=LH_DefaultDomain_77&hash=item3f4a9a518c)

**FaitMain,**

- <http://faitmain.org/volume-1/cable-gpio.html>

**Ni-Cd,**

- <http://www.ni-cd.net/accusphp/theorie/lithium/charge.php>
- <http://www.ni-cd.net/accusphp/chargeur/realisation/lithium.php>
- <ftp://ftp2.ni-cd.net/nicd/datasheet/LM3420.pdf>

**OpenClassRoom,**

- <http://openclassrooms.com/forum/sujet/convertisseur-dc-dc>

**PiHW,**

- <https://pihw.wordpress.com/guides/direct-network-connection/>

**RS,**

- <http://befr.rs-online.com/web/p/dc-dc-converters/8213370/>
- <http://befr.rs-online.com/web/p/graphics-display-development-kits/8417850/>
- <http://befr.rs-online.com/web/p/battery-charge-controller-ics/7386367/>
- <http://befr.rs-online.com/web/p/non-isolated-dc-dc-converters/0461947P/>

**SuperPat,**

- <http://blog.superpat.com/2012/06/08/raspberry-pi-fix-for-hdmi-to-dvi-cable-issue/>

**4dSystems,**

- [http://www.4dsystems.com.au/productpages/4DPi-32/downloads/4DPi-32\\_datasheet\\_R\\_1\\_2.pdf](http://www.4dsystems.com.au/productpages/4DPi-32/downloads/4DPi-32_datasheet_R_1_2.pdf)