http://fr.pinout.xyz/pinout/wiringpi

- BCM - numérotation Broadcom, plus communément apellée "GPIO"; utile pour travailler avec la librarie RPi.GPIO
- WiringPi - numérotation Wiring Pi; utile pour travailler avec la librarie Wiring Pi
- Physique - numérotation correspondante à la position physique des broches sur le connecteur GPIO

Installation de WiringPi

Téléchargement et installation

WiringPi est maintenu dans un GIT pour facilité le suivi des modifications. Si GIT n'est pas installé, vous pouvez l'installer avec la commande suivante :

```
sudo apt-get install git-core
```

Si vous rencontrez une erreur sur cette étape, vous devrez alors faire une mise à jour de Raspbian à l'aide de la commande suivante:

```
sudo apt-get update
sudo apt-get upgrade
```

Vous pouvez obtenir WiringPi en utilisant GIT:

```
git clone git://git.drogon.net/wiringPi
```

Si vous avez déjà utilisé une opération "clone" lors d'une précédente installation, vous pouvez exécuter la commande suivante:

```
cd wiringPi
git pull origin
```

Qui rapatrie une version mise-à-jour que vous pouvez finalement recompiler à l'aide du script ci-dessous.

Vous trouverez un script simplifié pour compiler/installer WiringPi:

```
cd wiringPi
./build
```

Tester l'installation de wiringPi

Exécutez les commandes du GPIO pour vérifier l'installation de WiringPi:

```
gpio -v
gpio readall
```

Initialisation

There are four ways to initialise wiringPi.

•int wiringPiSetup (void) ;
•int wiringPiSetupGpio (void) ;
•int wiringPiSetupPhys (void) ;
•int wiringPiSetupSys (void) ;

Une de ces fonctions doit être appelée au démarrage de votre programme sinon votre programme ne fonctionnera pas correctement.

Les fonctions d'initialisation

Voici les différentes méthodes d'initialisation disponibles.

wiringPiSetup

```
wiringPiSetup(void);
```

Initialise wiringPi et par du principe que le programme appelant va utilisé le principe de numérotation wiringPi. C'est un schéma de numérotation simplifié qui fournit une correspondance ("*mapping*" en anglais) de broches virtuelle allant de 0 à 16 vers les vrais numéros de broches du GPIO Broadcom.

Cette fonction à besoin d'être appelée avec le privilège root (administrateur).

wiringPiSetupGpio

```
wiringPiSetupGpio(void);
```

Fonction identique à la précédente, mais permet au programme appelant d'utiliser la numérotation des broches du GPIO Broadcom directement... et donc sans *re-mapping*.

Comme la précédente, cette fonction doit être appelée avec le privilège *root* (administrateur). Faite attention car certaines broches du connecteur GPIO sont différentes entre la révision 1 et la révision 2 des cartes.

wiringPiSetupPhys

```
wiringPiSetupPhys(void);
```

Fonction identique à la précédente. Mais permet au programme appelant d'utiliser les numéros de broche physique sur *le connecteur P1 seulement*.

Comme pour les autres, cette fonction doit être appelée avec les privilèges *root*.

wiringPiSetupSys

```
wiringPiSetupSys(void);
```

Initialise également wiringPi mais utilise l'interface /sys/class/gpio plutôt que de faire des accès matériels direct. Peut-être appelé avec un privilège utilisateur *non-root* ayant exporté les broches du GPIO au préalable avec les programme **gpio**.

Les fonctions principales

Les fonctions du noyaux

pinMode

```
void pinMode(int pin, int mode);
```

Fixe le modde d'une broche.

•**pin**: la broche dont vous voulez modifier le mode.
•**mode**: Le mode désiré parmi les options suivantes:
•**INPUT** pour en faire une entrée.
•**OUTPUT** pour en faire une sortie.
•**PWM_OUTPUT** pour en faire une sortie PWM.
•**GPIO_CLOCK** *(par encore identifié mais son nom laisse à penser qu'il s'agit d'un signal d'horloge*
Limitations:

•Seule la broche 1 de wiringPi (BCM_GPIO 18) supporte la sortie PWM.
•Seule la broche 7 wiringPi (BCM_GPIO 4) supporte le mode CLOCK output modes.
Mode Sys:

Cette fonction n'a aucun effet lorsque l'on est en mode Sys. Si vous avez besoin de changer le mode d'une broche alors vous pouvez le faire avec le programme **gpio** dans un script avant de démarrer le programme.

pullUpDnControl

```
void pullUpDnControl(int pin, int pud);
```

Permet d'activer les résistances pull-up ou pull-down sur une broche donnée, qui doit bien entendu être une broche d'entrée.

A la différence des Arduino, le BCM2835 dispose des deux types de résistances internes. Les résistances pull-up servent à ramener le potentiel à +vcc et les pull-down ramène le potentiel à GND (0v).

•**pin**: Broche d'entrée dont vous coulez modifier la configuration de la résistance pull-up/down.

•**pud**: le mode d'activation de la résistance parmi les valeurs suivantes:
•**PUD_OFF** pas de résistance pull up/down.
•**PUD_DOWN** activer la résistance pull-down (vers la masse).
•**PUD_UP** activer la résistance pull-up (vers 3.3v).
Les résistance pull-up/down internes ont une valeur approximative de 50KΩ (50 000Ω) sur un Raspberry Pi.

Mode sys:
Cette fonction n'a aucun effet sur les broches GPIO du Raspberry Pi lorsque l'on est en mode Sys. Si vous avez besoin d'activer les pull-up/dow, alors vous pouvez le faire à l'aide du programme **gpio** (que vous utilisez dans un script juste avant de démarrer votre programme).

digitalWrite

```
void digitalWrite(int pin, int value);
```

Ecrire un niveau haut ou bas sur une broche préalablement configurée en sortie.

•**pin**: la broche dont vous voulez modifier l'état.
•**value**: Le nouvel état de la broche parmi les valeurs suivantes.
•**HIGH** ou 1 pour placer la broche à l'état haut (soit 3.3v)
•**LOW** ou 0 pour placer la broche à l'état bas (soit la masse/gnd).
WiringPi traite toutes les valeurs différentes de zéro comme **HIGH** (état haut). 0 est la seule représentation de **LOW**.

pwmWrite

```
void pwmWrite(int pin, int value);
```

Ecrit une valeur dans le registre PWM pour une broche donnée. Raspberry Pi dispose d'une broche PWM, la broche 1 (BMC_GPIO 18, broche physique 12) dont la valeur varie entre 0 et 1024. D'autre périphériques PWM pourrait disposer d'autre gamme de valeur PWM.

Cette fonction n'est pas capable de contrôler le broche PWM du Pi lorsque l'on est en mode Sys.

digitalRead

```
int digitalRead(int pin);
```

Cette fonction retourne la valeur lue sur une broche spécifiée.

Paramètre:

•**pin**: broche dont vous voulez lire l'état.
La valeur retournée sera:

•**HIGH** ou 1 si la broches est à un état logique haut (3.3 volts).
•**LOW** ou 0 si la broche est à un état logique bas (0 volt/masse/GND).

analogRead

```
analogRead(int pin);
```

Retourne la valeur lue sur une broche d'entrée analogique. Vous devrez enregistrer un module complémentaire capable de lire des valeurs analogiques pour pouvoir utiliser cette fonction.

Paramètre:

•**pin**: broche analogique que vous désirez lire.
•

analogWrite

```
analogWrite(int pin, int value);
```

Ecrit une valeur analogique sur la broche désignée. Vous devez enregistrer un module analogique complémentaire tel que Gertboard pour pouvoir utiliser cette fonction sur un Pi.

Paramètres:

•**pin**: Broche dont vous voulez modifier la valeur analogique
•**value**: La nouvelle valeur analogique de la broche.