Les GPIO du Raspberry Pi

<https://www.blaess.fr/christophe/2012/11/26/les-gpio-du-raspberry-pi/>

*(Ne pas oublier de consulter les commentaires sur l’URL)*

Le Raspberry Pi offre quelques possibilités d’entrées-sorties directes en utilisant les broches GPIO présentes sur son connecteur P1. Elles ne sont pas très nombreuses (une dizaine) mais cela peut suffire pour des petits projets interactifs nécessitant d’interroger des capteurs tout-ou-rien ou de valider des actionneurs.

Nous pouvons utiliser ces GPIO de différentes façons, depuis l’espace utilisateur ou depuis le noyau. Voyons-en rapidement les principaux aspects…

# Les broches GPIO

Sur [le connecteur P1 du Raspberry Pi](https://www.blaess.fr/christophe/2012/11/02/spi-sur-raspberry-pi-1/), nous pouvons trouver plusieurs broches consacrées aux entrées-sorties GPIO. Celles-ci peuvent être configurées individuellement en entrées ou en sorties numériques. Attention, la tension appliquée sur une borne d’entrée doit rester inférieure à 3.3 V.

Les GPIO directement accessibles sont les suivantes.

| **Broche** | **GPIO** |
| --- | --- |
| 3 | 0 (rev.1) ou 2 (rev.2) |
| 5 | 1 (rev.1) ou 3 (rev.2) |
| 7 | 4 |
| 11 | 17 |
| 12 | 18 |
| 13 | 21 (rev.1) ou 27 (rev.2) |
| 15 | 22 |
| 16 | 23 |
| 18 | 24 |
| 22 | 25 |

On peut remarquer que certaines broches (3, 5 et 13) ont changé d’affectations au gré des versions du Raspberry Pi, aussi évitera-t-on de les employer pour garder un maximum de portabilité aux applications.

# Accès depuis l’espace utilisateur

L’accès simple, depuis le shell – ou tout autre programme de l’espace utilisateur – peut se faire très aisément grâce au système de fichiers /sys.

/ # **cd /sys/class/gpio/**

/sys/class/gpio # **ls**

export gpiochip0 unexport

Demandons l’accès au GPIO 24 (broche 18).

/sys/class/gpio # **echo 24 > export**

/sys/class/gpio # **ls**

export gpio24 gpiochip0 unexport

/sys/class/gpio # **cd gpio24/**

/sys/devices/virtual/gpio/gpio24 # **ls**

active\_low direction edge subsystem uevent value

/sys/devices/virtual/gpio/gpio24 # **cat direction**

in

## Sortie de signal

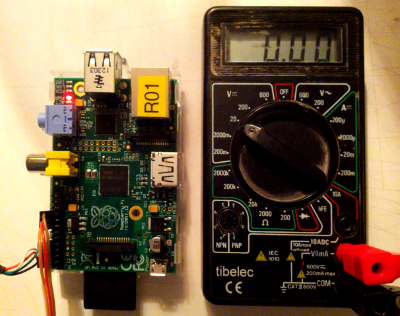
Par défaut, les broches GPIO sont dirigées en entrée. Inversons le sens du 24, puis regardons sa valeur.

/sys/devices/virtual/gpio/gpio24 # **echo out > direction**

/sys/devices/virtual/gpio/gpio24 # **cat value**

0

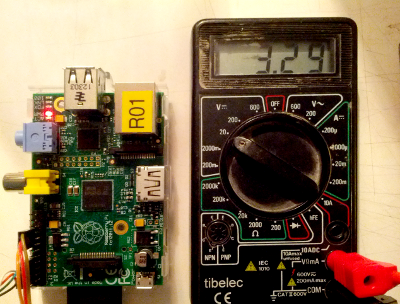
Effectivement, le voltmètre confirme qu’il n’y a pas de tension sur la broche.

[](https://www.blaess.fr/christophe/2012/11/24/les-gpio-du-raspberry-pi/img-01/)

Modifions l’état de la sortie

/sys/devices/virtual/gpio/gpio24 # **echo 1 > value**

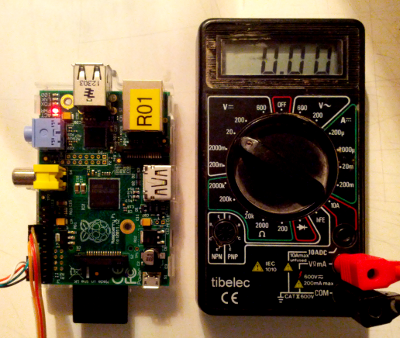
Vérifions la tension.

[](https://www.blaess.fr/christophe/2012/11/24/les-gpio-du-raspberry-pi/img-02/)

Re-basculons la sortie à zéro.

/sys/devices/virtual/gpio/gpio24 # **echo 0 > value**

/sys/devices/virtual/gpio/gpio24 #

[](https://www.blaess.fr/christophe/2012/11/24/les-gpio-du-raspberry-pi/img-03/)

## Lecture d’état

Demandons à présent l’accès à la broche 16 (Gpio 23)

# **cd /sys/class/gpio/**

/sys/class/gpio # **echo 23 > export**

/sys/class/gpio # **ls**

export gpio23 gpio24 gpiochip0 unexport

/sys/class/gpio # **cd gpio23/**

/sys/devices/virtual/gpio/gpio23 # **cat direction**

in

Je relie la broche 16 à la broche 20 (GND), puis je lis la valeur d’entrée.

/sys/devices/virtual/gpio/gpio23 # **cat value**

0

Je relie à présent la broche 16 à la broche 17 (+3.3V)

/sys/devices/virtual/gpio/gpio23 # **cat value**

1

Retour à nouveau sur la broche 20 (GND).

/sys/devices/virtual/gpio/gpio23 # **cat value**

0

/sys/devices/virtual/gpio/gpio23 #

# Accès depuis le kernel

## Lectures et écritures

Nous pouvons écrire un petit module pour accéder en lecture et écriture aux mêmes broches. Dans le module ci-dessous un timer à 8Hz fait clignoter la sortie sur la broche 18 seulement si la broche 16 est mise à 1 (+3.3V).

[**rpi-gpio-1.c**](https://www.blaess.fr/christophe/files/article-2012-11-26/rpi-gpio-1.c)

#include <linux/module.h>

#include <linux/timer.h>

#include <linux/gpio.h>

#include <linux/fs.h>

// Sortie sur broche 18 (GPIO 24)

#define RPI\_GPIO\_OUT 24

// Entree sur broche 16 (GPIO 23)

#define RPI\_GPIO\_IN 23

static struct timer\_list rpi\_gpio\_1\_timer;

static void rpi\_gpio\_1\_function (unsigned long unused)

{

static int value = 1;

value = 1 - value;

if (**gpio\_get\_value(RPI\_GPIO\_IN)** == 0)

value = 0;

**gpio\_set\_value(RPI\_GPIO\_OUT, value)**;

mod\_timer(& rpi\_gpio\_1\_timer, jiffies+ (HZ >> 3));

}

static int \_\_init rpi\_gpio\_1\_init (void)

{

int err;

if ((err = **gpio\_request(RPI\_GPIO\_IN,THIS\_MODULE->name)**) != 0)

return err;

if ((err = **gpio\_request(RPI\_GPIO\_OUT,THIS\_MODULE->name)**) != 0) {

gpio\_free(RPI\_GPIO\_IN);

return err;

}

if ((err = **gpio\_direction\_input(RPI\_GPIO\_IN)**) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

return err;

}

if ((err = **gpio\_direction\_output(RPI\_GPIO\_OUT,1)**) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

return err;

}

init\_timer(& rpi\_gpio\_1\_timer);

rpi\_gpio\_1\_timer.function = rpi\_gpio\_1\_function;

rpi\_gpio\_1\_timer.data = 0; // non utilise

rpi\_gpio\_1\_timer.expires = jiffies + (HZ >> 3);

**add\_timer(& rpi\_gpio\_1\_timer)**;

return 0;

}

static void \_\_exit rpi\_gpio\_1\_exit (void)

{

**del\_timer(& rpi\_gpio\_1\_timer)**;

**gpio\_free(RPI\_GPIO\_OUT)**;

**gpio\_free(RPI\_GPIO\_IN)**;

}

module\_init(rpi\_gpio\_1\_init);

module\_exit(rpi\_gpio\_1\_exit);

MODULE\_LICENSE("GPL");

La compilation se fait avec le fichier Makefile suivant. Les lignes KERNEL\_DIR et CROSS\_COMPILE indiquent respectivement l’emplacement du répertoire de compilation du noyau pour le Raspberry Pi et le préfixe pour la toolchain sur mon système. Il faut les adapter à votre environnement.

[**Makefile**](https://www.blaess.fr/christophe/files/article-2012-11-26/Makefile)

ifneq (${KERNELRELEASE},)

obj-m += rpi-gpio-1.o

else

ARCH ?= arm

**KERNEL\_DIR ?= ~/linux-3.2.27**

**CROSS\_COMPILE ?= /usr/local/cross/rpi/bin/arm-linux-**

MODULE\_DIR := $(shell pwd)

CFLAGS := -Wall

all: modules

modules:

${MAKE} -C ${KERNEL\_DIR} ARCH=${ARCH} CROSS\_COMPILE=${CROSS\_COMPILE} SUBDIRS=${MODULE\_DIR} modules

clean:

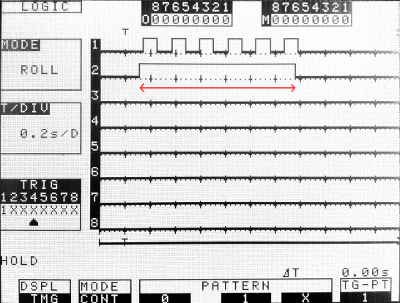
rm -f \*.o .\*.o .\*.o.\* \*.ko .\*.ko \*.mod.\* .\*.mod.\* .\*.cmd

rm -f Module.symvers Module.markers modules.order

rm -rf .tmp\_versions

endif

Après compilation et transfert sur le Raspberry Pi, nous chargeons le module et observons à l’aide d’un analyseur logique les deux broches 16 (canal 2) et 18 (canal 1), tandis qu’un signal de +3.3V est envoyé sur la broche 16.

[](https://www.blaess.fr/christophe/2012/11/24/les-gpio-du-raspberry-pi/img-04/)

Durant la période (mise en évidence par un trait rouge) où la broche 16 se voit appliquer une tension de +3.3V, nous voyons bien une oscillation de la sortie sur la broche 18.

## Interruptions

À présent nous allons traiter les interruptions déclenchées par une entrée GPIO. Le module ci-dessous installe un handler d’interruption pour le GPIO 23 (broche 16). À chaque déclenchement de l’interruption, notre handler basculera l’état de la broche de sortie 18 (GPIO 24).

[**rpi-gpio-2.c**](https://www.blaess.fr/christophe/files/article-2012-11-26/rpi-gpio-2.c)

#include <linux/interrupt.h>

#include <linux/module.h>

#include <linux/gpio.h>

// Sortie sur broche 18 (GPIO 24)

#define RPI\_GPIO\_OUT 24

// Entree sur broche 16 (GPIO 23)

#define RPI\_GPIO\_IN 23

static irqreturn\_t rpi\_gpio\_2\_handler(int irq, void \* ident)

{

static int value = 1;

**gpio\_set\_value(RPI\_GPIO\_OUT, value);**

value = 1 - value;

return IRQ\_HANDLED;

}

static int \_\_init rpi\_gpio\_2\_init (void)

{

int err;

if ((err = gpio\_request(RPI\_GPIO\_OUT, THIS\_MODULE->name)) != 0)

return err;

if ((err = gpio\_request(RPI\_GPIO\_IN, THIS\_MODULE->name)) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

return err;

}

if ((err = gpio\_direction\_output(RPI\_GPIO\_OUT,1)) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

return err;

}

if ((err = gpio\_direction\_input(RPI\_GPIO\_IN)) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

return err;

}

if ((err = **request\_irq(gpio\_to\_irq(RPI\_GPIO\_IN), rpi\_gpio\_2\_handler, IRQF\_SHARED | IRQF\_TRIGGER\_RISING, THIS\_MODULE->name, THIS\_MODULE->name)**) != 0) {

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

return err;

}

return 0;

}

static void \_\_exit rpi\_gpio\_2\_exit (void)

{

**free\_irq(gpio\_to\_irq(RPI\_GPIO\_IN), THIS\_MODULE->name);**

gpio\_free(RPI\_GPIO\_OUT);

gpio\_free(RPI\_GPIO\_IN);

}

module\_init(rpi\_gpio\_2\_init);

module\_exit(rpi\_gpio\_2\_exit);

MODULE\_LICENSE("GPL");

Le second module ayant été ajouté dans le Makefile, nous pouvons le compiler de la même façon que le précédent. Transférons-le sur le Raspberry Pi, puis chargeons-le dans le kernel.

/ # **cat /proc/interrupts**

CPU0

3: 362 ARMCTRL BCM2708 Timer Tick

32: 419 ARMCTRL dwc\_otg, dwc\_otg\_pcd, dwc\_otg\_hcd:usb1

52: 0 ARMCTRL BCM2708 GPIO catchall handler

65: 20 ARMCTRL ARM Mailbox IRQ

66: 1 ARMCTRL VCHIQ doorbell

77: 123 ARMCTRL bcm2708\_sdhci (dma)

83: 18 ARMCTRL uart-pl011

84: 317 ARMCTRL mmc0

FIQ: usb\_fiq

Err: 0

/ # **insmod rpi-gpio-2.ko**

/ # **cat /proc/interrupts**

CPU0

3: 434 ARMCTRL BCM2708 Timer Tick

32: 463 ARMCTRL dwc\_otg, dwc\_otg\_pcd, dwc\_otg\_hcd:usb1

52: 0 ARMCTRL BCM2708 GPIO catchall handler

65: 28 ARMCTRL ARM Mailbox IRQ

66: 1 ARMCTRL VCHIQ doorbell

77: 124 ARMCTRL bcm2708\_sdhci (dma)

83: 119 ARMCTRL uart-pl011

84: 326 ARMCTRL mmc0

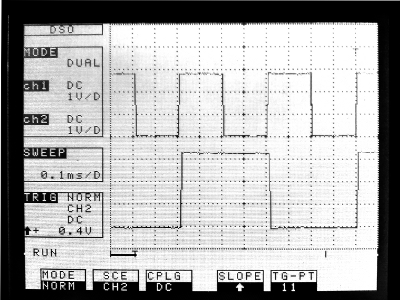
193: 0 GPIO rpi\_gpio\_2

FIQ: usb\_fiq

Err: 0

/ #

Nous voyons que la ligne d’interruption (numéro 193) est apparue dans /proc/interrupts. Pour l’instant aucune interruption ne s’est déclenchée. Connectons sur cette entrée un Générateur-Basse-Fréquence, qui lui envoie un signal carré [0, +3.3V] de 2.5kHz environ.

Aussitôt un signal carré apparaît sur la broche de sortie, avec une fréquence moitié du précédent. Nous pouvons le vérifier avec un oscilloscope.  
[](https://www.blaess.fr/christophe/wp-content/uploads/2012/11/img-05.png)

Nous pouvons également vérifier dans /proc/interrupts que le nombre d’interruptions 193 traitées progresse régulièrement.

/ # **cat /proc/interrupts**

CPU0

3: 646 ARMCTRL BCM2708 Timer Tick

32: 617 ARMCTRL dwc\_otg, dwc\_otg\_pcd, dwc\_otg\_hcd:usb1

52: 29791 ARMCTRL BCM2708 GPIO catchall handler

65: 60 ARMCTRL ARM Mailbox IRQ

66: 1 ARMCTRL VCHIQ doorbell

75: 1 ARMCTRL

77: 124 ARMCTRL bcm2708\_sdhci (dma)

83: 187 ARMCTRL uart-pl011

84: 342 ARMCTRL mmc0

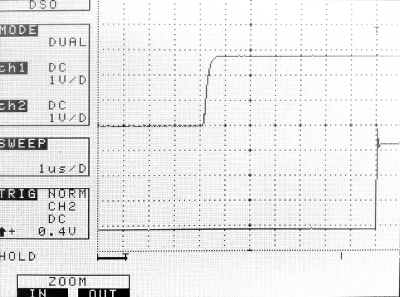
193: 29792 GPIO rpi\_gpio\_2

FIQ: usb\_fiq

Err: 0

/ #

En zoomant sur le point de déclenchement, nous mesurons la durée de prise en compte de l’interruption.

[](https://www.blaess.fr/christophe/wp-content/uploads/2012/11/img-06.png)

La durée entre la montée du signal d’entrée et la réponse du handler est d’environ 7 micro-secondes, ce qui est tout à fait correct pour cette gamme de microprocesseur.

# Conclusion

L’accès aux GPIO du port d’extension du Raspberry Pi est très simple, tant depuis l’espace utilisateur que depuis le noyau. Il existe d’autres broches permettant des entrées-sorties GPIO, mais elles ont une autre fonctionnalité par défaut (RS-232, SPI, etc.).