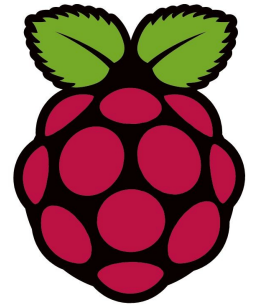




Java dans le Hard

Avec Raspberry Pi



Laurent HUET
SOFTEAM

Qui suis-je ?



@lhuet35

laurent.huet { @softeam.fr (pro) 
@gmail.com (perso) 

✨ **SOFTEAM depuis 2004**

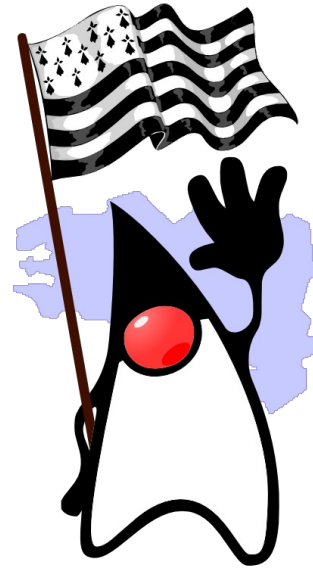
✨ *Consultant / Formateur / Architecte JavaEE*

✨ *Responsable Technique Softeam Ouest depuis fin 2010*

✨ **Sema / SchlumbergerSema / Atos Origin - 2000 à 2004**

✨ **S3EB (filiale Bouygues) – 1996 à 2000**

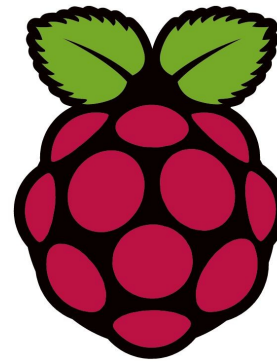
✨ *GTB / GTC (Gestion Technique du Bâtiment / Centralisée)*



Raspberry Pi

✚ C'est quoi ?

✚ Ca sert à quoi ?

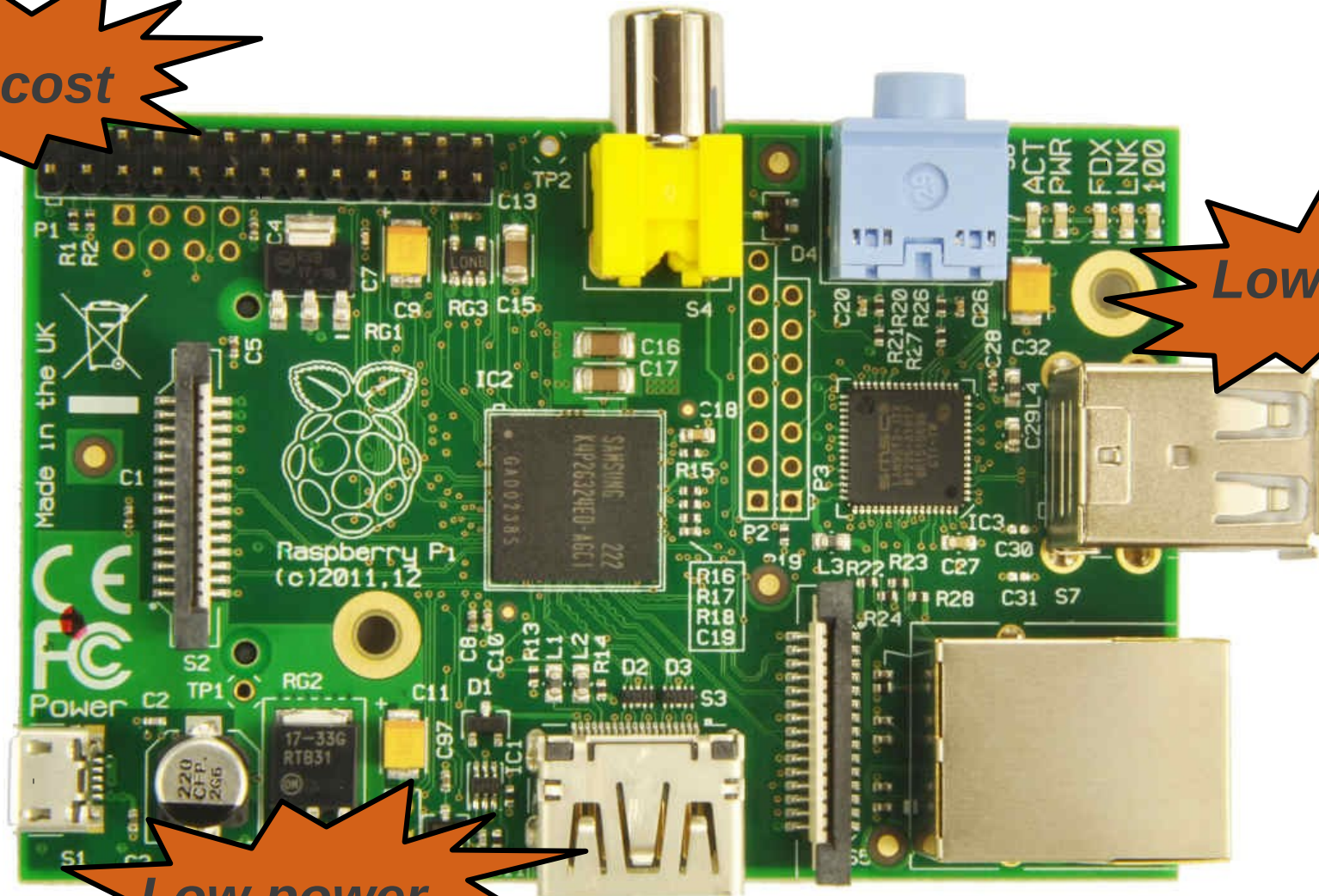


Carte ARM « Open source hardware »

Low cost

Low size

Low power



🌈 Carte « Low cost »

🌈 CPU ARMv6 @700MHz – 256 / 512 Mo RAM

🌈 “Linux inside”

🌈 Connectiques

🌈 SD Card

🌈 HDMI + RCA Vidéo

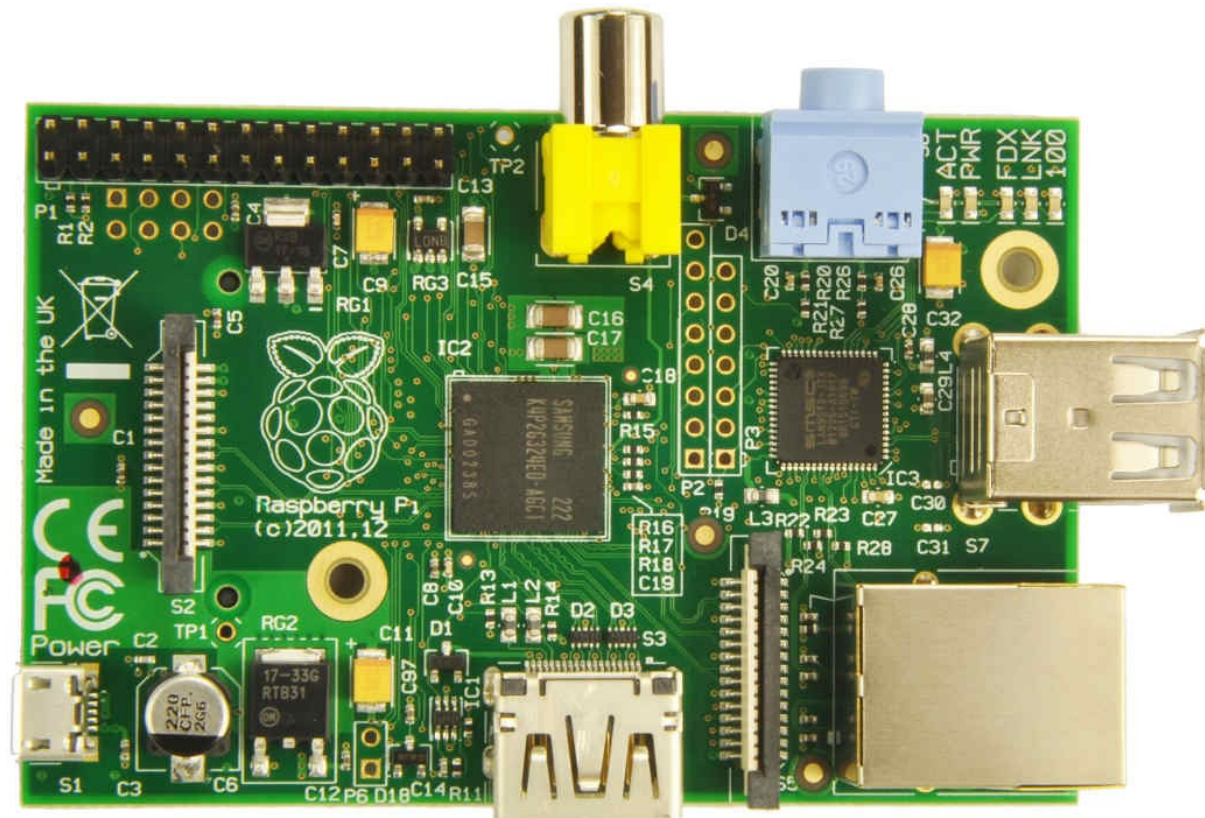
🌈 Son (jack 3.5)

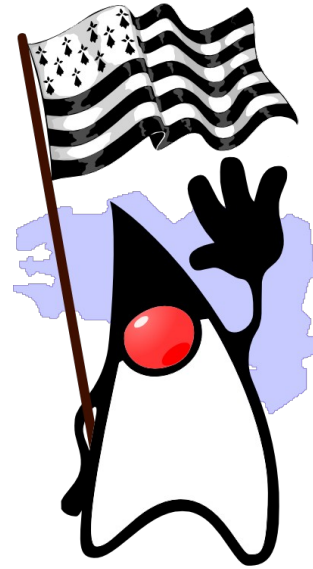
🌈 USB Host

🌈 Lan RJ45

🌈 Connecteur 26-Pins

🌈 GPIO / I2C / SPI / UART





Autres cartes ...

🌈 Microcontrôleurs

- 🌈 Programme limité (4ko à 128ko selon les versions)
- 🌈 Pas d'OS – Programme en “pseudo-C” en mémoire flash
- 🌈 ~16 MHz !



✨ Carte « Low Cost »

✨ CPU ARMv7 @720MHz – 256Mo

✨ OS : Linux / Android

✨ Caractéristiques

✨ Micro SD

✨ USB Host

✨ Ethernet (RJ45)

✨ 2 Connecteurs 46 Pins

✨ 66 GPIO

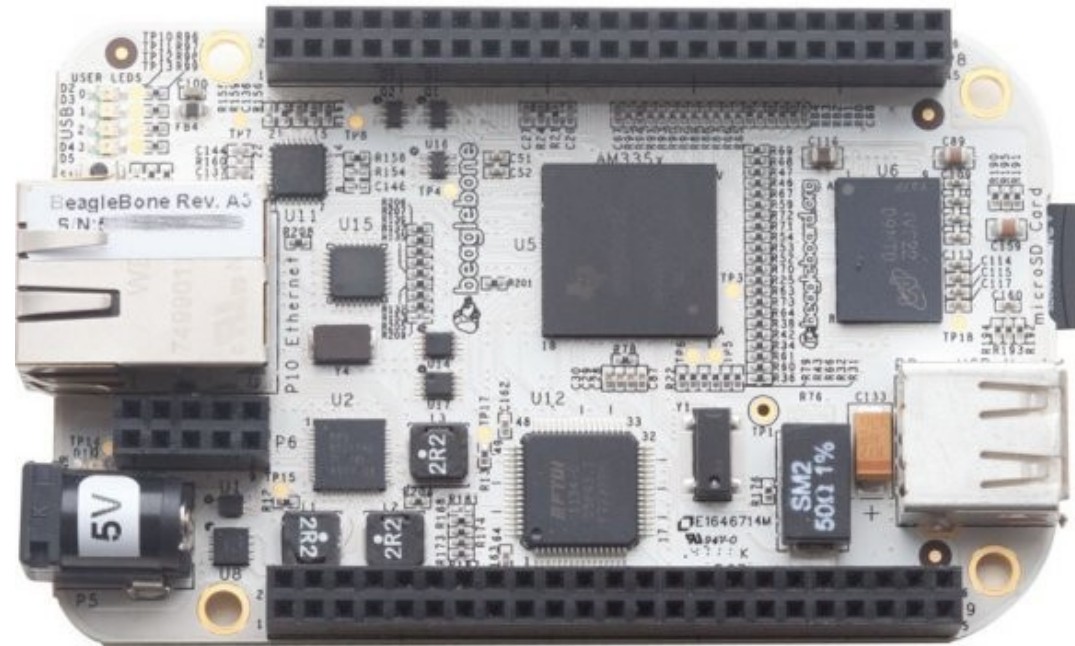
✨ 5 UART

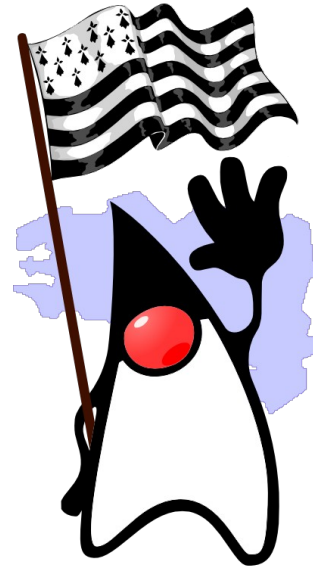
✨ 7 entrées analogiques

✨ 2 I2C

✨ 2 SPI

✨ 2 CAN





Pour quoi faire ?

Quoi faire avec la carte Raspberry Pi ?



- ✚ **Station météo**

- ✚ **Domotique**

- ✚ **Robotique**

- ✚ **OpenCV et OpenNI**

 - ✚ Traitement de l'image en temps réel

 - ✚ Reconnaissance de la voix, mouvement, ...

- ✚ **Media Center**

 - ✚ Distribution OpenELEC par exemple (<http://openelec.tv>)

Quels OS ?



✨ Linux

✨ Raspbian

✨ Distrib. officielle



✨ <http://www.raspbian.org/>

✨ Ubuntu

✨ KO : Support ARMv7 uniquement

✨ Armhf (HFP) vs Armel (SFP)



✨ Android Pi

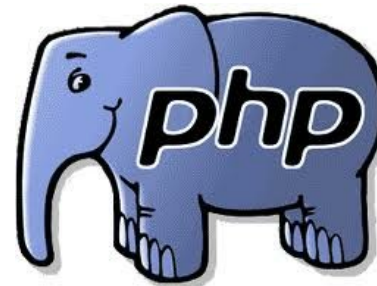
✨ http://androidpi.wikia.com/wiki/Android_Pi_Wiki

Quel langage utiliser ?

✨ Celui qui vous convient !

The C/C++ logo features the text "C/C++" in blue, overlaid on a light blue circular background containing snippets of C and C++ code.

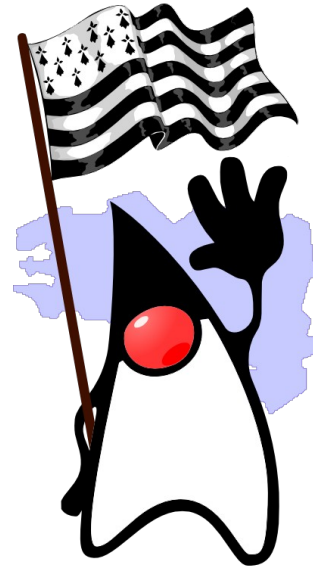
```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```



JavaScript

BASH





Le coté Hard

🚧 Attention aux fils ...

« Disclaimer »

✨ DANGER

- ✨ Toujours vérifier les tensions !
- ✨ Destruction de la carte possible
- ✨ Les pins sont proches !

✨ Tensions à respecter

- ✨ GPIO : 3,3 V
- ✨ Connection directe au SoC

✨ Puissances à respecter

- ✨ Cf wiki : http://elinux.org/RPi_Low-level_peripherals
 - ✨ 50 mA max par pins
- ✨ Attention aux périphériques USB !



Cablage sur Raspberry Pi

✈ Connecteur 26 Pins

✈ 3,3 V / 5,5V / GND

✈ GPIOs

✈ Entrées / sorties numériques

✈ Bus 1-wire

✈ PWM (modulation d'impulsions)

✈ UART (port série)

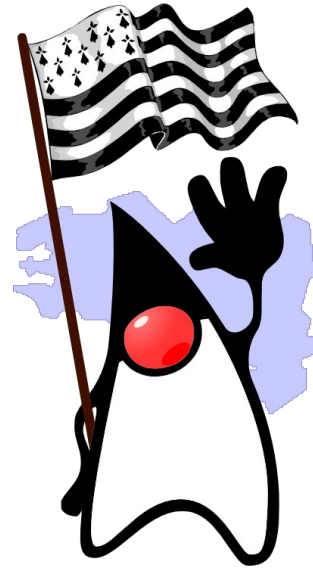
✈ I2C et SPI

✈ Discussion avec d'autres puces

✈ Cartes d'extension

✈ Capteurs "intelligents"





Entrées/Sorties numériques

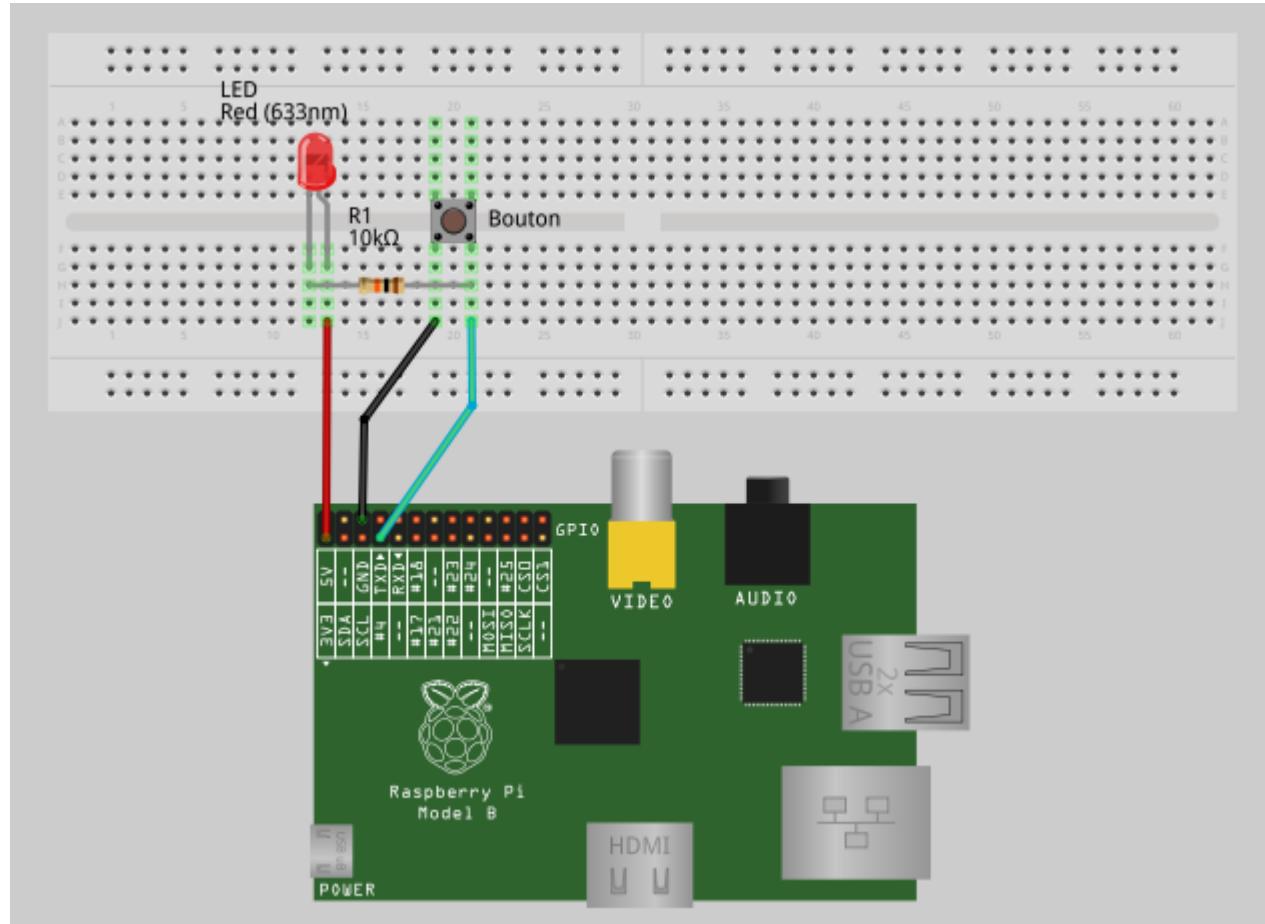
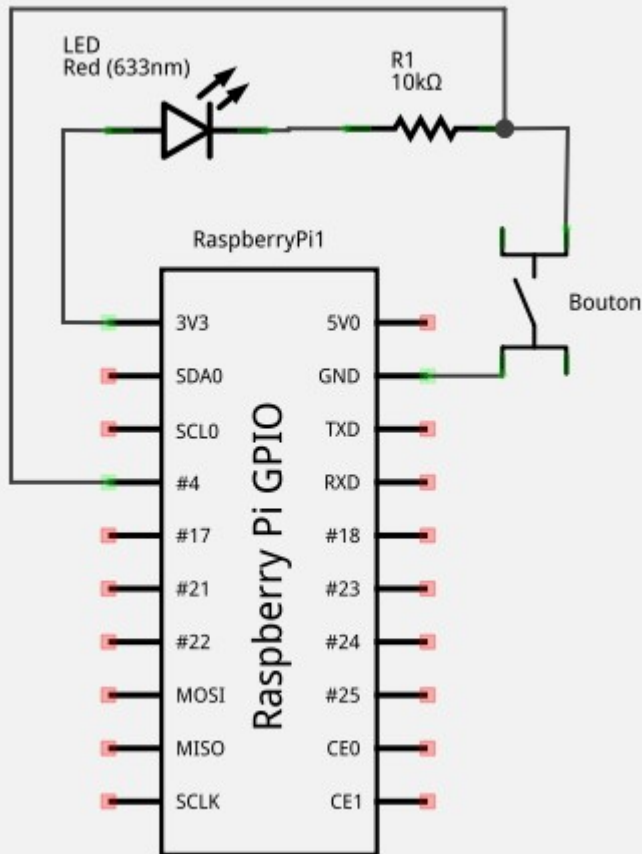
- ✨ GPIO (General Purpose Input/Output)

- ✨ Exemples

 - ✨ Allumer une lampe (led)

 - ✨ Détection d'un bouton

Montage « type »



Comment utiliser les GPIO ?



✈ API C

- ✈ Code C / C++
- ✈ Interface `gpio.h`

✈ API Sysfs

- ✈ Utilisation d'un système de fichier virtuel
 - ✈ API = lecture/écriture de fichiers !
- ✈ Principes
 - ✈ Activer le port GPIO
 - ✈ Configurer le port en input / output
 - ✈ Lire (si input) ou écrire (si output) dans le fichier
 - ✈ Désactiver le port GPIO

API Sysfs : simplicité garantie !



Fichier / répertoire	
<code>/sys/class/gpio</code>	Répertoire de base
<code>/sys/class/gpio/export</code>	Fichier pour initier un port GPIO (entrée/sortie)
<code>/sys/class/gpio/unexport</code>	Fichier pour désactiver un port GPIO (entrée/sortie)
<code>/sys/class/gpio/gpio<no></code>	Répertoire pour le port GPIO <no> une fois initié
<code>/sys/class/gpio/gpio<no>/direction</code>	Ecrire 'in' ou 'out' dans ce fichier pour configurer le port en entrée ou sortie
<code>/sys/class/gpio/gpio<no>/value</code>	Lecture / Ecriture de '0' ou '1' (Entrée / Sortie)
<code>/sys/class/gpio/gpio<no>/edge</code>	Ecriture de 'rising', 'falling' ou 'both' pour détecter un changement d'état

Le <no> correspond au numéro de GPIO sur la description du connecteur et non au numéro de pin ...

Exemple d'utilisation en bash



✨ Export du GPIO4 (pin 7)

```
$ echo 4 > /sys/class/gpio/export
```

✨ Configuration en sortie

```
$ echo "out" > /sys/class/gpio/gpio4/direction
```

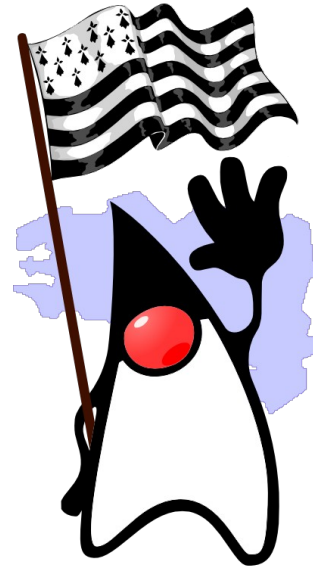
✨ Activation/désactivation de la sortie

```
echo 1 > /sys/class/gpio/gpio4/value
```

```
echo 0 > /sys/class/gpio/gpio4/value
```

✨ « Libération » du port

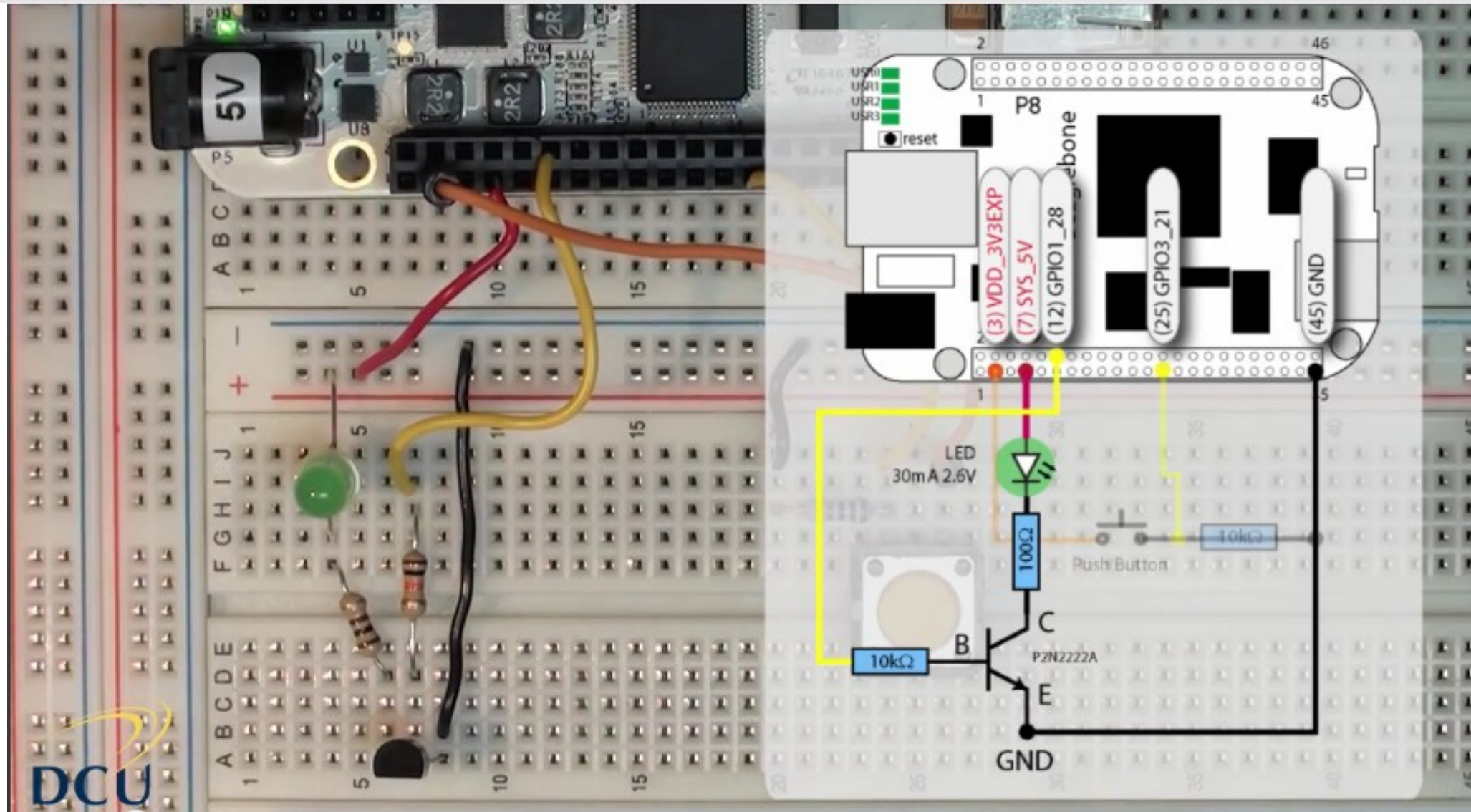
```
echo 4 > /sys/class/gpio/unexport
```

Piloter de la puissance

🌈 Passer d'une led à une lampe halogène !

Utilisation d'un transistor



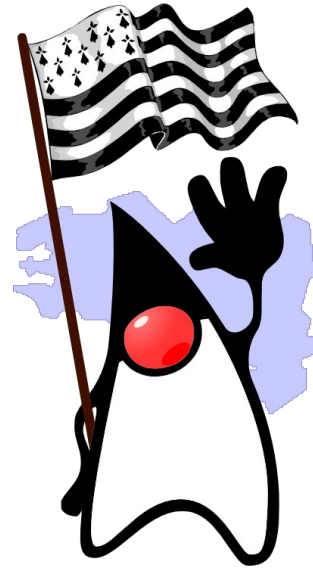
Source : <http://www.youtube.com/user/DerekMolloyDCU>

Utilisation de cartes avec relais

✨ Relais 5V

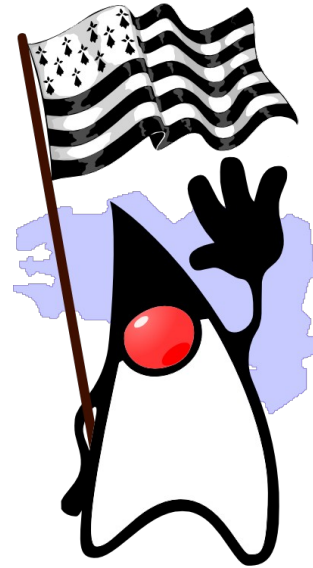
- ✨ GPIO = 3,3 V !
- ✨ Utilisation d'un transistor pour piloter du 5V





Le coté Soft

🚀 Avec Java inside !



Choisir sa JVM ...

🌈 Sur Raspberry PI (ARMv6)

Quelle JVM choisir ?



✈️ Raspbian armhf

- ✈️ OpenJDK 7

- ✈️ ZeroVM, JamVM et Avian

✈️ Raspbian armel

- ✈️ Oracle

- ✈️ JavaSE Embedded (6 et 7) et JavaSE (7 depuis août)

- ✈️ -server uniquement sur ARMv7

- ✈️ OpenJDK 7

- ✈️ ZeroVM, JamVM et Avian

✚ Bench DaCapo 9.12

- ✚ Uniquement avrora, fop, jython, pmd et xalan
- ✚ Suppression des tests sensibles aux accès disques
- ✚ Erreurs sur certains tests
 - ✚ Pas eu le temps d'investiguer
 - ✚ NoClassDefFound sur com.sun. ...

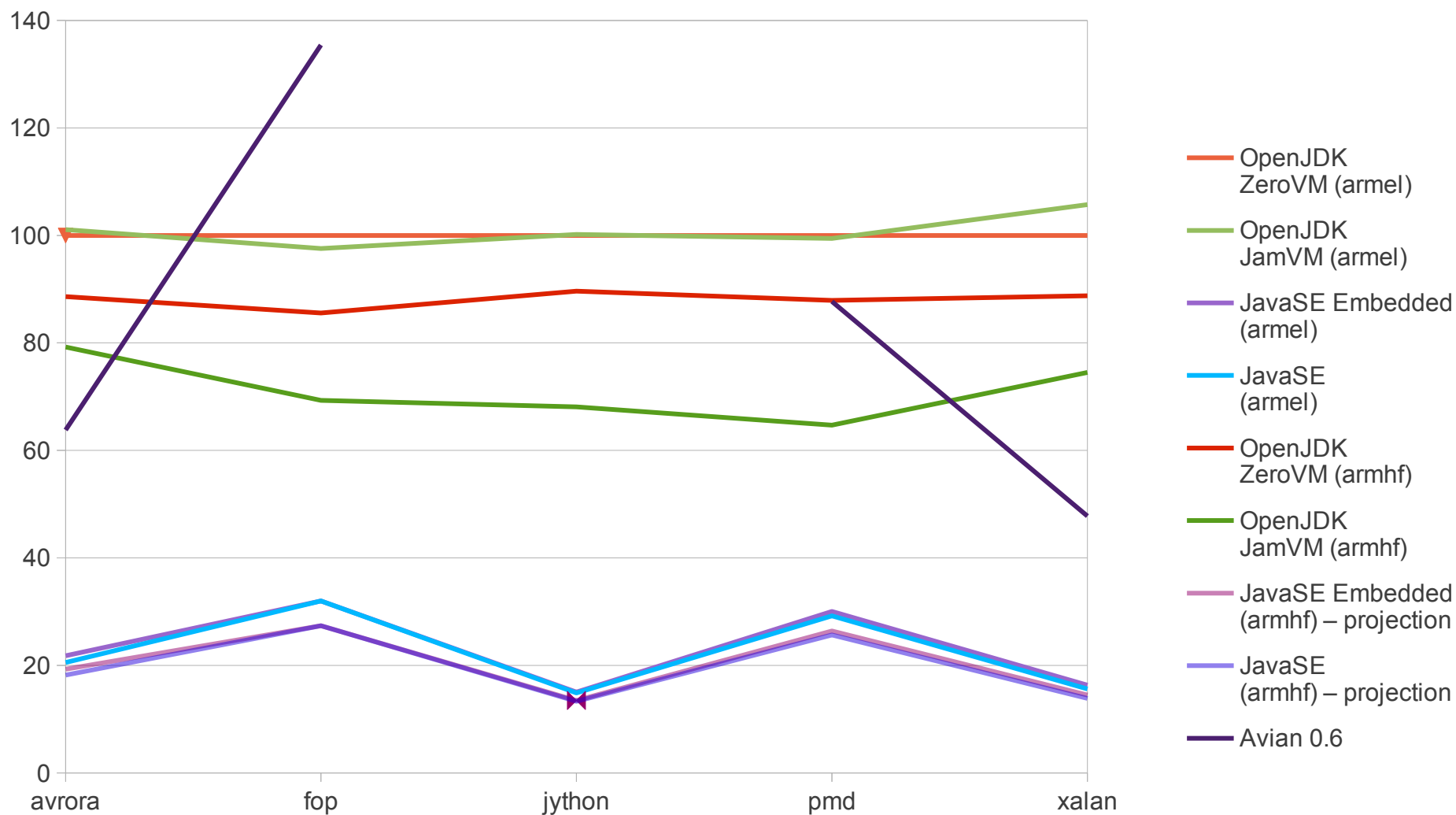
✚ Options JVM

- ✚ -xms128m -xmx128m

✚ Pas de test d'empreinte mémoire

- ✚ Aspect important sur l'embarqué

Tests de performance (Bench DaCapo)



✨ armhf vs armel

✨ > 15 % en perf

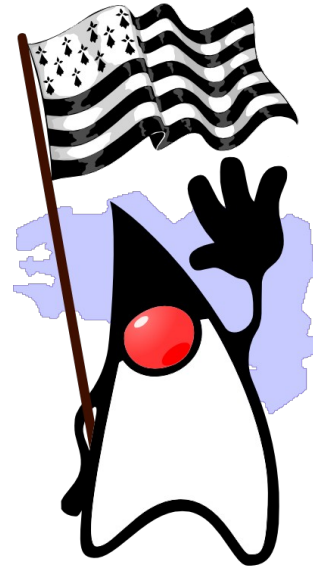
✨ JVM Oracle JavaSE la plus performante

✨ Gain possible avec armhf

✨ TODO

✨ Test web

✨ Comparatif Beaglebone (ARMv7)



Quelle abstraction choisir ?

🌈 On aime bien les frameworks :-)

✈ « Brigde » Java d'une Api native

✈ Librairie C WiringPi

✈ <https://projects.drogon.net/raspberry-pi/wiringpi/>

✈ Fonctionnalités majeures

✈ GPIO : configuration et pilotage (entrée/sortie)

✈ Gestion d'interruption (via callback) avec “trigger”

✈ I2C, SPI et RS232

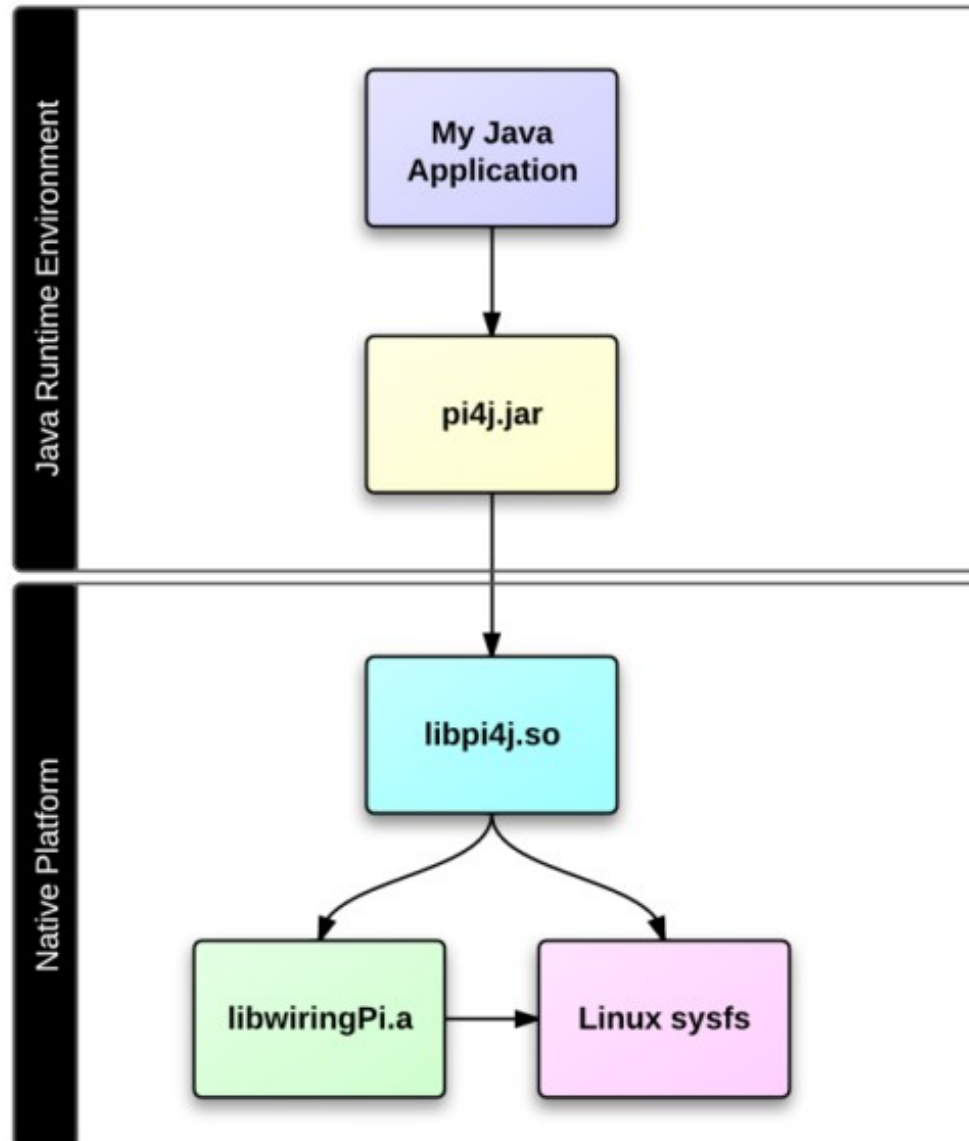
✈ A noter

✈ Site web : <http://pi4j.com/>

✈ Github : <https://github.com/Pi4J/pi4j>

✈ Licence Apache V2

Dépendances



Exemple simple



```
public class ControlGpioExample
{
    public static void main(String[] args) throws InterruptedException
    {
        // create gpio controller
        GpioController gpio = GpioFactory.getInstance();

        // provision gpio pin #01 as an output pin and turn on
        GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,
                                                                    "MyLED", PinState.HIGH);

        Thread.sleep(5000);














        // turn off gpio pin #01
        pin.low();
        Thread.sleep(5000);

        // toggle the current state of gpio pin #01 (should turn on)
        pin.toggle();
        Thread.sleep(5000);

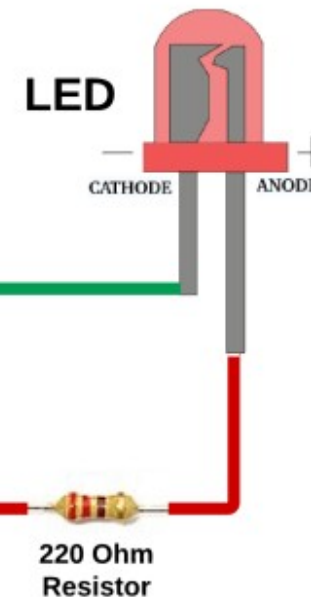
        // toggle the current state of gpio pin #01 (should turn off)
        pin.toggle();
        Thread.sleep(5000);

        // turn on gpio pin #01 for 1 second and then off
        pin.pulse(1000);
    }
}
```

Allumer une lumière

Raspberry Pi P1 Header					
PIN #	NAME			NAME	PIN #
	3.3 VDC Power	1		2	5.0 VDC Power
8	SDA0 (I2C)	3		4	DNC
9	SCL0 (I2C)	5		6	0V (Ground)
7	GPIO 7	7		8	TxD
	DNC	9		10	RxD
0	GPIO 0	11		12	GPIO1
2	GPIO2	13		14	DNC
3	GPIO3	15		16	GPIO4
	DNC	17		18	GPIO5
12	MOSI	19		20	DNC
13	MISO	21		22	GPIO6
14	SCLK	23		24	CE0
	DNC	25		26	CE1

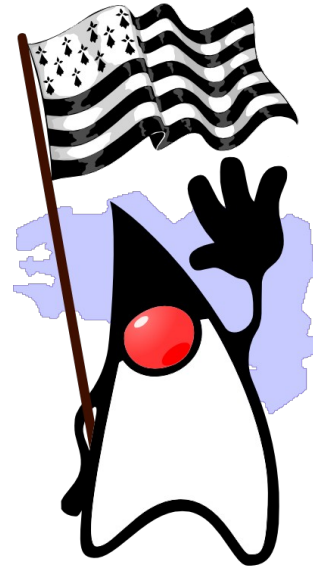
<http://www.pi4j.com>



Code pour allumer la lumière



```
public static void main(String[] args) throws InterruptedException {  
  
    // gpio controller  
    GpioController gpio = GpioFactory.getInstance();  
  
    // Configuration du pin #01 en sortie et à l'état "high"  
    GpioPinDigitalOutput pin =  
        gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,  
                                       "lumiere", PinState.HIGH);  
  
    pin.low();  
    Thread.sleep(5000);  
    pin.toggle();  
    Thread.sleep(5000);  
    pin.toggle();  
    Thread.sleep(5000);  
    pin.pulse(1000);  
}
```



Demo

Interrupteur

Momentary Switch



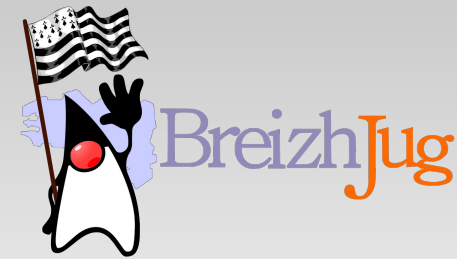
NOTE:

GPIO2 pull-down resistor must be enabled to prevent the pin state from floating.

Raspberry Pi P1 Header					
PIN #	NAME		NAME	PIN #	
		1	3.3 VDC Power	2	5.0 VDC Power
8	SDA0 (I2C)	3		4	DNC
9	SCL0 (I2C)	5		6	0V (Ground)
7	GPIO 7	7		8	TxD 15
	DNC	9		10	RxD 16
0	GPIO 0	11		12	GPIO1 1
2	GPIO2	13		14	DNC
3	GPIO3	15		16	GPIO4 4
	DNC	17		18	GPIO5 5
12	MOSI	19		20	DNC
13	MISO	21		22	GPIO6 6
14	SCLK	23		24	CE0 10
	DNC	25		26	CE1 11

<http://www.pi4j.com>

Ajout de Listener → Interrupteur

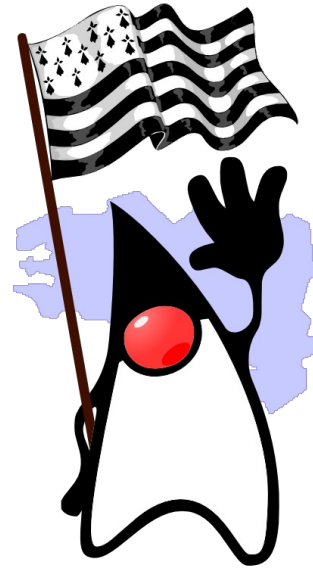


```
public class ListenGpioExample
{
    public static void main(String args[]) throws InterruptedException
    {
        GpioController gpio = GpioFactory.getInstance();
        GpioPinDigitalInput myButton = gpio.provisionDigitalInputPin(RaspiPin.GPIO_02,
                                                                    "MyButton", PinPullResistance.PULL_DOWN);

        myButton.addListener(new GpioExampleListener());

        // keep program running until user aborts (CTRL-C)
        for (;;)
        {
            Thread.sleep(500);
        }
    }
}

class GpioExampleListener implements GpioPinListenerDigital
{
    @Override
    public void handleGpioPinDigitalStateChangeEvent(GpioPinDigitalStateChangeEvent event)
    {
        System.out.println(" --> GPIO PIN STATE CHANGE: " + event.getPin() + " = "
                           + event.getState());
    }
}
```



Demo de Noël !



Questions ?



All text and image content in this document is licensed under the
[Creative Commons Attribution-Share Alike 3.0 License](https://creativecommons.org/licenses/by-sa/3.0/) (unless otherwise specified).