**Lab02: Compiling your first program “LED Blink”**

Le Raspberry Pi offre quelques possibilités d’entrées-sorties directes en utilisant les broches GPIO présentes sur son connecteur. 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. Le but ici est de balayer les principaux aspects.



1. **L’interface sysfs**

Sysfs est un système de fichiers virtuel introduit par le noyau Linux 2.6. Sysfs permet d'exporter depuis l'espace noyau vers l'espace utilisateur des informations sur les périphériques du système et leurs pilotes, et est également utilisé pour configurer certaines fonctionnalités du noyau.

Pour chaque objet ajouté dans l'arbre des modèles de pilote (pilotes, périphériques, classes de périphériques), un répertoire est créé dans sysfs.

* + 1. La relation parent/enfant est représentée sous la forme de sous-répertoires dans **/sys/devices/** (représentant la couche physique).
    2. Le sous-répertoire **/sys/bus/** est peuplé de liens symboliques, représentant la manière dont chaque périphérique appartient aux différents bus.
    3. **/sys/class/** montre les périphériques regroupés en classes, comme les périphériques réseau par exemple.

Pour les périphériques et leurs pilotes, des attributs peuvent être créés. Ce sont de simples fichiers, la seule contrainte est qu'ils ne peuvent contenir chacun qu'une seule valeur et/ou n'autoriser le renseignement que d'une valeur. Ces fichiers sont placés dans le sous-répertoire du pilote correspondant au périphérique.

1. **Lire/Ecrire sur une broche TOR**

**Accès depuis l’espace utilisateur**

L’accès simple, depuis le shell peut se faire très aisément grâce au système de fichiers **/sys**.

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

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

export gpiochip0 unexport

Accédons 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/class/gpio/gpio24 # **ls**

active\_low direction edge subsystem uevent value

**Ecriture**

La broche doit être configurée en sortie pour écrire :

/sys/class/gpio/gpio24 # **echo out > direction**

/sys/class/gpio/gpio24 # **echo 1 > value**

/sys/class/gpio/gpio24 # **echo 0 > value**

**Lecture**

La broche doit être configurée en entrée pour lire :

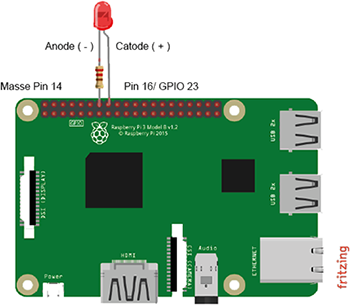
/sys/class/gpio/gpio24 # **echo in > direction**

/sys/class/gpio/gpio24 # **cat value**

0

# Programmation des GPIO

L’utilisation des ports GPIO est liée à l’accès à plusieurs fichiers. Les opérations d’export, de configuration de la direction, de lecture ou d’écriture peuvent être réalisées dans n’importe quel langage de programmation.



# Programme 1 : Clignoter une LED en C

1. Câbler le circuit
2. Copier le code suivant en utilisant la commande
   1. Sudo nano led.c

#include <stdio.h>

#include <wiringPi.h>

// LED Pin - wiringPi pin 0 is BCM\_GPIO 17 n°Pin 11.

#define LED 4

int main (void)

{

printf ("Raspberry Pi blink\n") ;

wiringPiSetup () ;

pinMode (LED, OUTPUT) ;

for (;;)

{

digitalWrite (LED, HIGH) ; // On

delay (500) ; // mS

digitalWrite (LED, LOW) ; // Off

delay (500) ;

}

return 0 ;

}

1. Installer la bibliothèque :
   1. cd /tmp
   2. wget <https://project-downloads.drogon.net/wiringpi-latest.deb>
   3. sudo dpkg -i wiringpi-latest.deb
2. Compilez le code : gcc led.c -o led -lwiringPi
3. Exécutez : ./led

# Programme 02 : Python

$ sudo apt-get install python-rpi.gpio python3-rpi.gpio

import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

from time import sleep # Import the sleep function from the time module

GPIO.setwarnings(False) # Ignore warning for now

GPIO.setmode(GPIO.BOARD) # Use physical pin numbering

GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value to low (off)

while True: # Run forever

GPIO.output(8, GPIO.HIGH) # Turn on

sleep(1) # Sleep for 1 second

GPIO.output(8, GPIO.LOW) # Turn off

sleep(1) # Sleep for 1 second

$ python blinking\_led.py

# Programme 03 : shell

#!/bin/bash

GPIO=23

GPIODIR=/sys/class/gpio/gpio$GPIO

echo "Configuring GPIO $GPIO"

#check if the gpio is already exported

if [ ! -e "$GPIODIR" ]

then

echo "Exporting GPIO"

echo $GPIO > /sys/class/gpio/export

else

echo "GPIO already exported"

fi

echo "Current direction: `cat $GPIODIR/direction`"

echo "Set GPIO as output"

echo out > $GPIODIR/direction

echo "New GPIO direction: `cat $GPIODIR/direction`"

echo "Current value: `cat $GPIODIR/value`"

echo "Set value as high"

echo 1 > $GPIODIR/value

echo "New value: `cat $GPIODIR/value`"

#Endless loop

echo "Start blinking, 1 sec on plus 1 sec off, press CTRL+C to end"

while ( true );

do echo 1 > $GPIODIR/value;

cat $GPIODIR/value;

sleep 1;

echo 0 > $GPIODIR/value;

cat $GPIODIR/value;

sleep 1;

done;

# Programme 04 : C++

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <string>

// Remplacez les ... par le code approprié

#define LED24 "/sys/class/gpio/ ..."

#define BUTTON "/sys/class/gpio/ ... "

using namespace std;

int main(...)

{

ofstream led24; ifstream button;

while(1)

{

button.open(...);

string buttonState;

getline(button,buttonState);

button23.close();

if(buttonState=="1")

{

led24.open(LED24);

//Allumer led24 << "...";

led24.close();

sleep(...);

led24.open(LED24);

//Eteindre led24 << "...";

led24.close();

sleep(...);

}

}

}