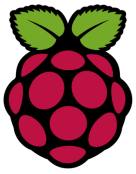
CARNUS
Enseignement supérieur RODEZ



TP Raspberry Pi

C, C++, Python ...

Prof : **Kamal Boudjelaba**

15 septembre 2025

BTS CIEL

LTP Charles Carnus
Avenue de Saint Pierre - Bourran - 12000 RODEZ, France

IR

Table des matières

1 Utilisation des broches GPIO (entrées/sorties) 3
1.1 Programme Python : faire varier l'intensité d'une LED avec PWM . . . 3
1.2 Librairie PIGPIO . . . 3
1.3 Programme C : faire clignoter une LED . . . 3
1.4 Programme Python : faire clignoter une LED . . . 4
2 Travaux Pratiques 5
2.1 TP 1 : Piloter les ports GPIO à partir d'un navigateur . . . 5
2.2 TP 1-V2 : Piloter les ports GPIO à partir d'un navigateur . . . 6
2.3 TP 2 : Allumer une LED avec PiGPIO en C . . . 8

Préambule

Prérequis

— Maîtrise de base du Raspberry Pi et de son système Linux.

— Notions de base en électronique (LED, résistances, signaux PWM).

Objectifs

— Comprendre l'utilisation des broches GPIO pour piloter des composants électroniques.

— Développer des scripts ou programmes capables d'interagir avec le matériel.

— Mettre en œuvre des solutions logicielles accessibles via un navigateur (serveur web embarqué).

Compétences visées

— **C09** : Installer un réseau informatique

— **C10** : Exploiter un réseau informatique

Activités et tâches associées

— **R2 – Installation et qualification**

  — R2-T5 : Réalisation des opérations avec contrôle matériel et logiciel

— **R3 – Exploitation et maintien en condition opérationnelle**

  — R3-T3 : Supervision de l'état du réseau

  — R3-T5 : Configuration matérielle et logicielle des équipements

1. Utilisation des broches GPIO (entrées/sorties)

1.1 Programme Python : faire varier l'intensité d'une LED avec PWM

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Programme : LED_Python.py

import RPi.GPIO as GPIO
import time

led_pin = 18  # GPIO 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

pwm = GPIO.PWM(led_pin, 100)  # fréquence PWM de 100 Hz
pwm.start(0)  # démarrer avec 0 % de rapport cyclique

try:
    while True:
        duty = input("Rapport cyclique désiré (0 à 100) : ")
        pwm.ChangeDutyCycle(float(duty))
except KeyboardInterrupt:
    pass
finally:
    pwm.stop()
    GPIO.cleanup()
```

Exécution :

```
python3 LED_Python.py
```

1.2 Librairie PIGPIO

Installation de la librairie

Sur les dernières versions de Raspberry Pi OS, `pigpio` est préinstallée. Sinon, utilisez :

```
sudo apt-get install pigpio python3-pigpio
```

Ou :

```
sudo apt update
sudo apt install pigpio
```

On peut aussi tester le démon seul :

```
sudo pigpiod # Pour démarrer le démon pigpio
pigs t 4 w  # Tester si on peut écrire sur le GPIO 4
```

Vérifier la version :

```
pigpiod -v
```

1.3 Programme C : faire clignoter une LED

```c
#include <pigpio.h>
#include <unistd.h> // pour sleep()

#define GPIO 14 // GPIO14 = Pin physique 8

int main(int argc, char *argv[]) {
    if (gpioInitialise() < 0) {
        return -1; // Erreur : impossible d'initialiser pigpio
    }

    gpioSetMode(GPIO, PI_OUTPUT);

    for (int i = 0; i < 60; i++) {
        gpioWrite(GPIO, 1);
        sleep(1);
        gpioWrite(GPIO, 0);
```

```
        sleep(1);
    }

    gpioTerminate();
    return 0;
}
```

Compilation :

```
gcc -o progc progc.c -lpigpio -lrt -lpthread
```

Exécution :

```
sudo killall pigpiod  # pour arrêter le démon gpio (et éviter certaines erreurs)
sudo pigpiod          # démarrer le démon
sudo ./progc          # exécuter le programme
```

Test sans code C :

```
sudo killall pigpiod  # Si nécessaire
sudo pigpiod
pigs w 14 1  # GPIO 14 ON
pigs w 14 0  # GPIO 14 OFF
```

1.4 Programme Python : faire clignoter une LED

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Une LED branchée à GPIO 25 clignote toutes les 3 secondes

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

try:
    while True:
        GPIO.output(25, GPIO.HIGH)
        time.sleep(3)
        GPIO.output(25, GPIO.LOW)
        time.sleep(3)
except KeyboardInterrupt:
    print("\nArrêt du programme.")
finally:
    GPIO.cleanup()
```

Sauvegarder ce fichier sous `blink.py` dans le répertoire `/home/pi`, puis exécuter avec :

```
cd /home/pi
python3 blink.py
```

2. Travaux Pratiques

2.1 TP 1 : Piloter les ports GPIO à partir d'un navigateur

Le but de ce TP est de piloter un port GPIO du Raspberry Pi à distance, depuis un navigateur (téléphone, tablette, PC). Pour cela, nous allons utiliser le micro-framework `web.py`, qui contient un serveur web minimaliste.

- Le navigateur se connecte à ton Raspberry Pi via HTTP (par exemple : http ://raspberrypi.local :8080)
- Une interface HTML s'affiche (boutons ON/OFF)
- Chaque bouton envoie une requête au serveur web.py
- Le serveur appelle la bibliothèque pigpio pour changer l'état d'un GPIO

Installation du module web.py

- Installer pip :

```
sudo apt update
sudo apt install python3-pip
```

- Vérifier Python et pip :

```
python3 --version
python3 -m pip --version
```

- Installer le module web :

```
sudo apt install python3-pip pigpio
sudo pip3 install web.py
```

- Démarre le démon pigpio :

```
sudo pigpiod
```

Script Python : `gpio4.py`

```
#!/usr/bin/env python3

import web
import pigpio

# Initialisation du GPIO
GPIO_PIN = 14
pi = pigpio.pi()  # Connexion au daemon pigpiod
pi.set_mode(GPIO_PIN, pigpio.OUTPUT)

# Routes
urls = (
    '/', 'Index',
    '/on', 'TurnOn',
    '/off', 'TurnOff'
)

# HTML simple
html_page = """
<html>
<head><title>Contrôle GPIO</title></head>
<body>
    <h1>Contrôle GPIO 14</h1>
    <form action="/on" method="get">
        <button type="submit">Allumer</button>
    </form>
    <form action="/off" method="get">
        <button type="submit">Éteindre</button>
    </form>
</body>
</html>
"""
```

```
class Index:
    def GET(self):
        return html_page

class TurnOn:
    def GET(self):
        pi.write(GPIO_PIN, 1)
        return html_page

class TurnOff:
    def GET(self):
        pi.write(GPIO_PIN, 0)
        return html_page

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

Lancer le serveur Web

```
sudo python3 gpio4.py
```

Accéder au site depuis un navigateur

Depuis le Raspberry Pi, ou un autre appareil connecté au même réseau :

- Sur le RPi : http://localhost:8080
- Depuis un PC ou téléphone :

http://IP_DU_RPI:8080

(On peut obtenir l'IP avec `hostname -I`)

Remarques :

- Le port par défaut utilisé par web.py est 8080.
- Assurez-vous que le démon `pigpiod` est activé.

2.2 TP 1-V2 : Piloter les ports GPIO à partir d'un navigateur

Utilisation de Flask + pigpio

Étapes du TP

- Créer une page web avec 2 boutons (ON / OFF)
- Quand on clique, une requête est envoyée à Flask
- Flask appelle pigpio pour changer l'état du GPIO

Installer les dépendances

```
sudo apt update
sudo apt install python3-pip pigpio
sudo pip3 install flask
```

Démarrer le démon pigpio (si ce n'est pas déjà fait) :

```
sudo pigpiod
```

script Python `gpio_web.py`

```
from flask import Flask, render_template_string, redirect, url_for
import pigpio

# Initialisation pigpio
GPIO_PIN = 14
pi = pigpio.pi()
pi.set_mode(GPIO_PIN, pigpio.OUTPUT)
```

```
# App Flask
app = Flask(__name__)

# HTML
html_page = """
<!DOCTYPE html>
<html>
<head>
    <title>Contrôle GPIO</title>
</head>
<body>
    <h1>GPIO {{ gpio }} - État : {{ state }}</h1>
    <form action="{{ url_for('on') }}" method="post">
        <button type="submit">Allumer</button>
    </form>
    <form action="{{ url_for('off') }}" method="post">
        <button type="submit">Éteindre</button>
    </form>
</body>
</html>
"""

@app.route('/')
def index():
    state = pi.read(GPIO_PIN)
    return render_template_string(html_page, gpio=GPIO_PIN, state='ON' if state else 'OFF')

@app.route('/on', methods=['POST'])
def on():
    pi.write(GPIO_PIN, 1)
    return redirect(url_for('index'))

@app.route('/off', methods=['POST'])
def off():
    pi.write(GPIO_PIN, 0)
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

Lancer le serveur Flask

```
sudo python3 gpio_web.py
```

Accéder à l'interface depuis un navigateur

Depuis le Raspberry Pi, ou un autre appareil connecté au même réseau :

- Sur le RPi : http://localhost:8080
- Depuis un PC ou téléphone :

http://IP_DU_RPI:8080

(On peut obtenir l'IP avec `hostname -I`)

2.3 TP 2 : Allumer une LED avec PiGPIO en C

Objectif : Écrire un programme en C qui allume une LED pendant 10 secondes, puis l'éteint.

```c
#include <pigpio.h>
#include <unistd.h> // pour sleep()

#define GPIO 17  // Remplacer par le GPIO que vous utilisez

int main()
{
    if (gpioInitialise() < 0) {
        return -1;
    }

    gpioSetMode(GPIO, PI_OUTPUT);

    gpioWrite(GPIO, 1);  // Allume la LED
    sleep(10);           // Attend 10 secondes
    gpioWrite(GPIO, 0);  // Éteint la LED

    gpioTerminate();
    return 0;
}
```

Compilation :

```
gcc -o led10s led10s.c -lpigpio -lrt -lpthread
```

Exécution :

```
sudo pigpiod   # si le démon n'est pas déjà lancé
sudo ./led10s
```

Bilan

- L'étudiant est capable de manipuler les broches GPIO.
- Il est apte à mettre en œuvre des interfaces de contrôle accessibles à distance (web).
- Il comprend les liens entre programmation embarquée, réseau et interaction matérielle.

Vérifier si `web.py` est installé

On peut utiliser cette commande dans le terminal :

```
python3 -m pip show web
```

On obtient une sortie comme :

```
Name: web.py
Version: 0.62
Summary: web.py: makes web apps
...
Location: /usr/local/lib/python3.9/dist-packages
```

Si **aucune sortie**, c'est que `web.py` n'est pas installé.

Test depuis Python

On lance Python et on fait un import :

```
python3
```

Puis dans l'interpréteur Python :

```
import web
print(web.__version__)
```

S'il affiche `ModuleNotFoundError: No module named 'web'`, c'est que le module n'est pas installé.

Lister tous les paquets installés via `pip`

```
python3 -m pip list
```

Il affiche une liste comme :

```
Package      Version
------------ -------
pip          23.2.1
web.py       0.62
Flask        2.2.5
...
```

Rechercher un package dans la liste

Si on a beaucoup de paquets installés :

```
python3 -m pip list | grep web
```

Cela retournera uniquement les lignes contenant "web", par exemple :

```
web.py       0.62
```