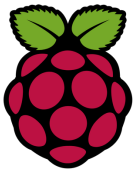
CHARLES CARNUS
LYCÉE TECHNIQUE
ENSEIGNEMENT SUPÉRIEUR



TP Raspberry Pi

C, C++, Python ...

Prof : **Kamal Boudjelaba**

18 septembre 2023

B T S SN

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

IR–EC

Table des matières

1. Introduction : commandes Linux Raspberry Pi

Commande	Signification
`$ raspi-config`	Outil de configuration pour Raspberry Pi
`$ raspistill`	Prendre une photo avec le module camera du Raspberry Pi
`$ apt`	Gestion des paquets sur Raspberry Pi OS
`$ ifconfig`	Afficher la configuration réseau actuelle
`$ nano`	Editeur de fichiers textes par défaut sur Raspberry Pi OS
`$ wget`	Téléchargement d'un fichier sur Raspberry Pi

Commande	Signification & Exemples
`$ apt-get update`	Télécharger la dernière version du dépôt `$ sudo apt-get update`
`$ apt-get upgrade`	Effectuer la mise à jour des paquets si nécessaire `$ sudo apt-get upgrade`

`$ apt-get install <paquet>`	Installer le ou les paquet(s) spécifié(s)
	`$ sudo apt-get install conda`
	`$ sudo apt-get install conda pip`

`$ apt-get remove <paquet>`	Désinstaller un paquet `$ sudo apt-get remove conda`

`$ dpkg -l`	Lister tous les paquets installés sur le système. On peut ajouter grep pour filtrer les résultats
	`$ dpkg -l`
	`$ dpkg -l | grep ...`

`$ apt-get remove <paquet>`	Désinstaller un paquet `$ sudo apt-get remove conda`

Commande	Signification & Exemples
`$ cd <chemin>`	Change de dossier (va au dossier indiqué entre dans <...>) `$ cd /home/pi`

`$ ls`	Lister les fichiers et dossiers présents dans le dossier actuel, ou celui spécifié
	
	`$ ls`
	
	`$ ls /home/pi`

`$ mkdir <dossier>`	Créer un dossier à l'emplacement actuel ou spécifié
	
	`$ mkdir MonDossier`
	
	`$ mkdir /home/pi/MonDossier`

`$ cp <source><destinationr>`	Copier un fichier ou un dossier d'un emplacement à l'autre
	
	`$ cp test.txt /home/pi/Documents/`
	
	`$ cp /home/pi/test.txt /home/pi/Documents/`

`$ mv <source><destinationr>`	Déplacer un fichier ou un dossier
	
	`$ mv /home/pi/test.txt /home/Documents/`
	
	`$ mv /home/pi/MonDossier/ /home/Documents/`

`$ cat <fichier>`	Afficher tout le contenu d'un fichier `$ cat /home/pi/test.txt`

`$ rm <fichier>`	Supprimer un fichier. Pour un dossier, l'option `-rf` devra être ajoutée (pour Forcer une suppression Récursive)
	
	`$ rm test.txt`
	
	`$ rm -rf /home/pi/MonDossier/`

`$ pwd`	Permet de savoir dans quel dossier vous êtes (à ne pas confondre avec `passwd`) `$ pwd`
`$ tree`	Analyser l'emplacement actuel dans l'arborescence des fichiers. Il montrera toute l'arborescence existante dans le dossier actuel ou spécifié `$ tree`

Commande	Signification & Exemples
`$ tar -c`	Utiliser `tar` pour regrouper plusieurs fichiers dans une même archive (généralement avec gzip afin de compresser des fichiers) `$ tar -cvfz archive.tar.gz /home/pi/Documents/MonDossier` `-c` : création d'une archive `-v` : mode verbeux `-f` : on spécifie le nom du fichier juste après `-z` : on utilise gzip pour la compression
`$ tar -x`	Même commande, mais pour extraire les fichiers `$ tar -xvfz archive.tar.gz`

2. Ecrire et exécuter des programmes sur le Raspberry Pi (Linux ARM)

2.1 Introduction : Editeur de texte nano

Nano est un éditeur de texte pour les systèmes d'exploitation Unix et Linux. Nano est doté de nombreuses fonctions puissantes et permet d'éditer et de créer divers fichiers.
Certains OS, comme MacOS et Linux, sont généralement livrés avec l'éditeur de texte Nano préinstallé. Pour vérifier, il suffit d'utiliser la commande suivante :

```
$ nano -version
```

Pour installer Nano sur les machines Debian ou Ubuntu, exécuter la commande suivante :

```
$ sudo apt-get install nano
```

- Démarrer nano : ouverture d'un fichier texte sans nom
  `$ sudo nano`
- Ouvrir un fichier existant :
  `$ sudo nano Nom_Fichier.Extension` Exemple : `$ sudo nano prog1.c`
  - Si le fichier n'est pas dans le répertoire courant, il faut spécifier le chemin complet :
    `$ sudo nano /home/pi/MesProgrammes/prog2.py`

Si on entre un nom de fichier et que ce fichier n'est pas présent dans le répertoire, Nano créera un nouveau fichier.
Si on exécute la commande nano sans spécifier le nom du fichier, Nano créera un fichier vide sans nom et demandera un nom lorsqu'on quitte l'éditeur.

Au bas de la fenêtre Nano, on trouve certains des raccourcis utilisables avec l'éditeur Nano. Le symbole ∧ signifie qu'on doit appuyer sur `CTRL + [Touche]` pour lancer la commande choisie.
Appuyer sur `CTRL + O` pour enregistrer les modifications apportées au fichier et poursuivre l'édition. Pour fermer l'éditeur, appuyer sur `CTRL + X`. S'il y a des modifications, il sera demandé si on souhaite les enregistrer ou non. Saisir Y pour Oui, ou N pour Non, puis on appuie sur Entrée. Mais s'il n'y a pas de changements, on quittera l'éditeur immédiatement.

Commande	Signification
CTRL + A	Aller au début de la ligne.
CTRL + E	Aller à la fin de la ligne.
CTRL + Y	Faire défiler la page vers le bas.
CTRL + V	Faire défiler la page vers le haut.
CTRL + G	Cette commande ouvre une fenêtre d'aide.
CTRL + O	Pour sauvegarder le fichier. Lorsque cette commande est utilisée, on sera invité à modifier ou vérifier le nom de fichier souhaité et après avoir appuyé sur Entrée, elle enregistrera le fichier.
CTRL + W	Elle est utilisée pour rechercher une expression spécifique dans le texte. Elle fonctionne comme la commande CTRL + F sur d'autres applications. Pour rechercher la même expression, appuyer sur ALT + W autant de fois qu'il le faut.
CTRL + K	Coupe toute la ligne sélectionnée dans le "presse-papier".
CTRL + U	Colle le texte du "presse-papier" dans la ligne sélectionnée.
CTRL + X	Quitte l'éditeur de texte Nano. Dans le cas où on a apporté des modifications au fichier, elle nous invite à sauvegarder.

2.2 Programme en C

1. Créer un nouveau dossier nommé `ProgNano` dans l'emplacement `/home/pi`
2. Ouvrir un nouveau fichier (texte) nommé `ProgC.c` en utilisant l'éditeur de texte "nano".
   `$ sudo nano ProgC.c`
   Ce fichier doit avoir comme chemin `/home/pi/ProgNano/ProgC.c`
3. Ecrire le programme suivant puis enregistrer et quitter nano.

```c
#include <stdio.h>

int main(void)
{
    printf("Le programme en C nommé ProgC.c et écrit dans nano a été exécuté.\n");
    return 0;
}
```

4. Compiler le programme `ProgC.c` :
   `$ gcc -o TestC ProgC.c`
5. Exécuter le programme :
   `$ ./TestC`

2.3 Programme en C++

1. Dans le dossier `ProgNano`, créer (ouvrir) un nouveau fichier (texte) nommé `ProgCPP.cpp` en utilisant l'éditeur de texte "nano".
   `$ sudo nano ProgCPP.cpp`
   Ce fichier doit avoir comme chemin `/home/pi/ProgNano/ProgCPP.cpp`
2. Ecrire le programme suivant puis enregistrer et quitter nano.

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    cout << "Le programme en C++ nommé ProgCPP.cpp et écrit dans nano a été exécuté." << endl;
    return 0;
}
```

3. Compiler le programme `ProgCPP.cpp` :
   `$ g++ -o TestCPP ProgCPP.cpp`
4. Exécuter le programme :
   `$ ./TestCPP`

2.4 Programme en Python

1. Dans le dossier `ProgNano`, créer (ouvrir) un nouveau fichier (texte) nommé `ProgPython.py` en utilisant l'éditeur de texte "nano".
   `$ sudo nano ProgPython.py`
   Ce fichier doit avoir comme chemin `/home/pi/ProgNano/ProgPython.py`
2. Ecrire le programme suivant puis enregistrer et quitter nano.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print("Le programme en Python nommé ProgPython.py et écrit dans nano a été exécuté.")
```

3. Exécuter le programme : ce n'est pas utile de compiler le programme car il est exécuté à la volée par l'interpréteur python
   `$ python ProgPython.py`

2.5 Geany : Editeur de texte pour développer sur Raspberry Pi

Geany est un éditeur de texte orienté développement qui est disponible sur toutes les plateformes (Windows, macOS, Linux) et pour les microprocesseurs ARM qui équipent les Raspberry Pi. Geany dispose d'une interface graphique, ce qui est beaucoup plus sympathique à utiliser.
Il apporte tout d'abord la coloration syntaxique qui permet de faciliter la mise au point et la recherche d'erreur. Plusieurs languages sont pris en charge (C++, Python, html, php, ruby, java ...).
Geany est préinstallé sur Raspberry Pi OS avec Desktop. Il se trouve dans le menu principal, sous Programmation, avec les autres outils de développement tels que Thonny Python IDE ...

Exécuter du code depuis Geany

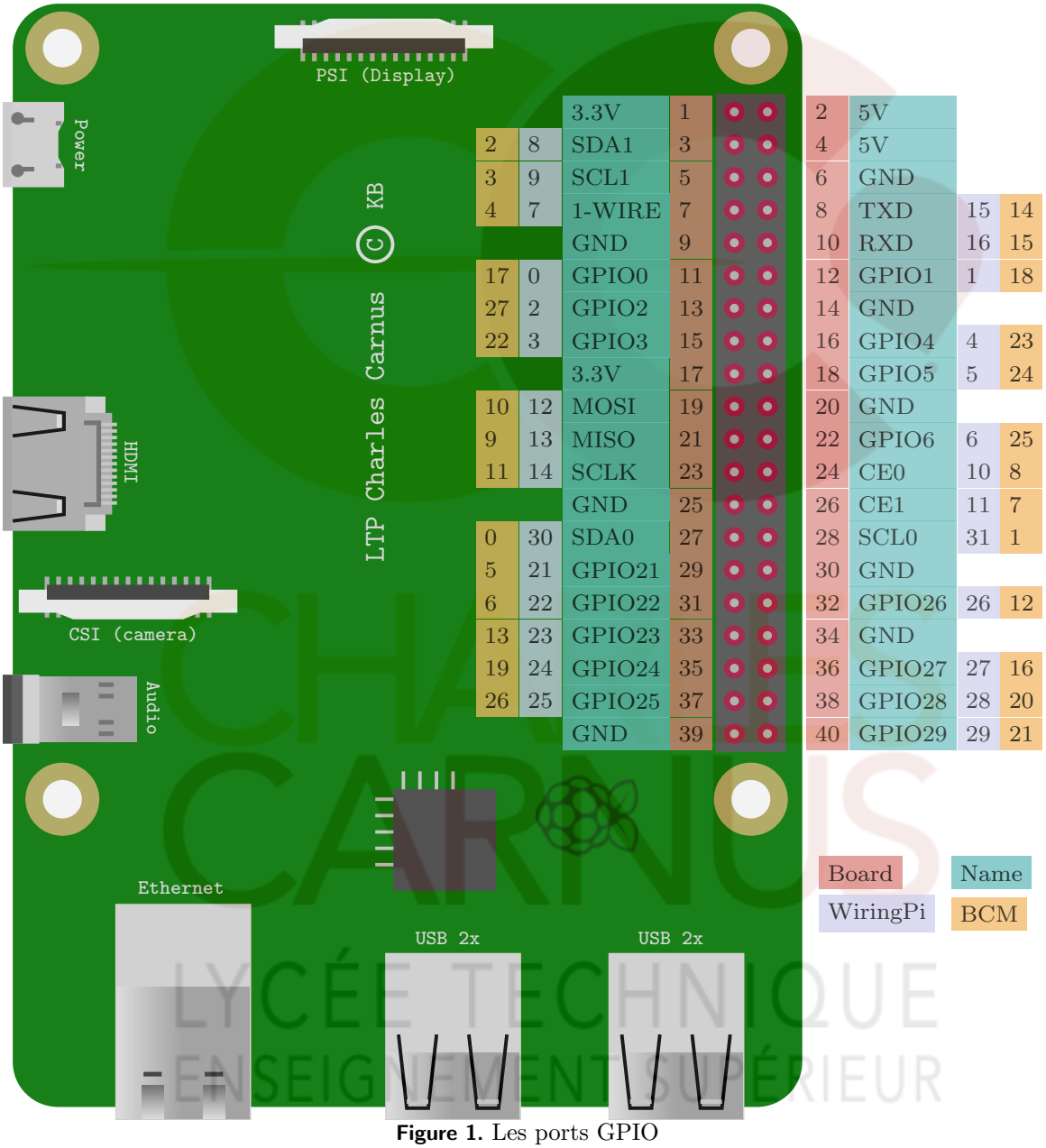
Inutile de quitter Geany pour exécuter votre code.

- Enregistrer le script
- Construire
- Compiler
  Le résultat de la compilation est directement disponible en bas de l'écran dans l'onglet Compilateur
- Exécuter, un Terminal s'ouvre automatiquement et lance le script.

2.6 Thonny Python IDE

Thonny est un IDE (environnement de développement) minimaliste qui permet de programmer en Python. Cet outil intègre son propre interpréteur Python.

3. Utilisation des broches GPIO (entrées/sorties)



		3.3V	1	2	5V		
2	8	SDA1	3	4	5V		
3	9	SCL1	5	6	GND		
4	7	1-WIRE	7	8	TXD	15	14
		GND	9	10	RXD	16	15
17	0	GPIO0	11	12	GPIO1	1	18
27	2	GPIO2	13	14	GND		
22	3	GPIO3	15	16	GPIO4	4	23
		3.3V	17	18	GPIO5	5	24
10	12	MOSI	19	20	GND		
9	13	MISO	21	22	GPIO6	6	25
11	14	SCLK	23	24	CE0	10	8
		GND	25	26	CE1	11	7
0	30	SDA0	27	28	SCL0	31	1
5	21	GPIO21	29	30	GND		
6	22	GPIO22	31	32	GPIO26	26	12
13	23	GPIO23	33	34	GND		
19	24	GPIO24	35	36	GPIO27	27	16
26	25	GPIO25	37	38	GPIO28	28	20
		GND	39	40	GPIO29	29	21

Figure 1. Les ports GPIO

3.1 Programme en Python pour faire varier l'intensité d'une LED

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
# Programme LED_Python.py

servo_pin = 18   # GPIO 18
GPIO.setmode(GPIO.BCM)   # BCM
GPIO.setup(servo_pin, GPIO.OUT)   # pin configurée en sortie
pwm = GPIO.PWM(servo_pin, 50)   # pwm : fréquence = 50 Hz
rapport = 7 # rapport cyclique initial de 7% (rapport cyclique est en pourcentage, de 0 à 100)
pwm.start(rapport)

while True: # Appuyer sur CTRL-C pour arrêter le programme
  print("Rapport cyclique actuel: " , rapport)
  rapport = raw_input ("Rapport cyclique désiré:   ")
  pwm.ChangeDutyCycle(float(rapport))
```

Exécution : `$ python LED_Python.py`

3.2 Librairie PIGPIO

Installer la librairie

Sur les nouvelles versions de Raspbian (Raspberry Pi OS), la librairie "pigpio" est déjà préinstallée.

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

Pour connaître la version installée :

```
pigpiod -v
```

3.3 Programme C pour faire clignoter une LED

```
#include <pigpio.h>
#include <unistd.h>

#define GPIO 14

int main(int argc, char *argv[])
{
        if (gpioInitialise() <0 ) {
                return -1;
        }

        gpioSetMode(GPIO , PI_OUTPUT);

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

Compilation : `gcc -o progc progc.c -lpigpio -lrt -lpthread`
Exécution : `sudo ./progc`

Ou

Compilation : `gcc -Wall -pthread -o progc progc.c -lpigpio -lrt`
Exécution : `sudo ./progc`

3.4 Programme Python pour faire clignoter une LED

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Une LED branchée à la pin 25 clignote
# BCM GPIO25 (numéro 22 physique)
```

```
import RPi.GPIO as GPIO # bibliothèque pour utiliser les GPIO
import time # bibliothèque pour gestion du temps

GPIO.setmode(GPIO.BCM) # mode de numérotation des pins
GPIO.setup(25,GPIO.OUT) # la pin 25 réglée en sortie (output)

while True: # boucle répétée jusqu'à l'interruption du programme
  GPIO.output(25,GPIO.HIGH) # sortie au niveau logique haut (3.3 V)
  time.sleep(3) # on ne change rien pendant 3 seconde
  GPIO.output(25,GPIO.LOW) # sortie au niveau logique bas (0 V)
  time.sleep(3)
```

Sauvegarder le sous le nom `blink.py` dans un fichier sur le répertoire `/home/pi`
On lance ensuite le programme avec les commandes suivantes :

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

4. Travaux Pratiques

4.1 TP 1 : Utiliser les ports GPIO (Python)

Partie 1

On veut réaliser un code en Python pour faire clignoter une LED.

1. allumer la LED,
2. attendre 1 seconde,
3. éteindre la LED,
4. attendre 1 seconde,
5. continuer en 1. (on répète 2 fois la boucle)

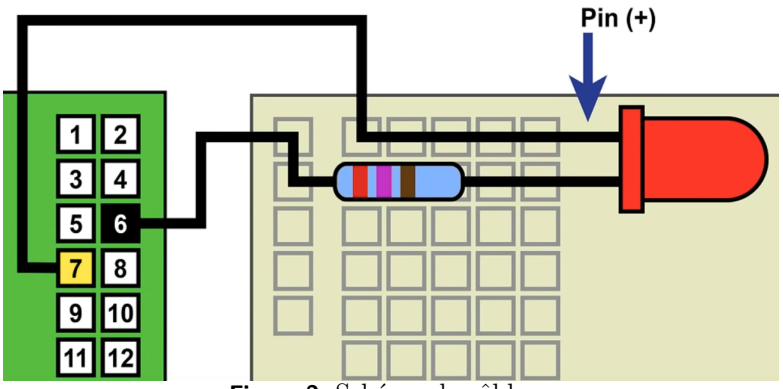


Figure 2. Schéma de câblage

Partie 1 : solution

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)  # Mode de numérotation des pins.
GPIO.setup(7,GPIO.OUT)    # Sortie

for x in range(0,3):
    GPIO.output(7,True)
    time.sleep(1)
    GPIO.output(7,False)
    time.sleep(1)

GPIO.cleanup()
```

Partie 2

On veut maintenant réaliser un code en Python pour faire clignoter 3 LEDs successivement. La figure 3 montre le montage à réaliser.

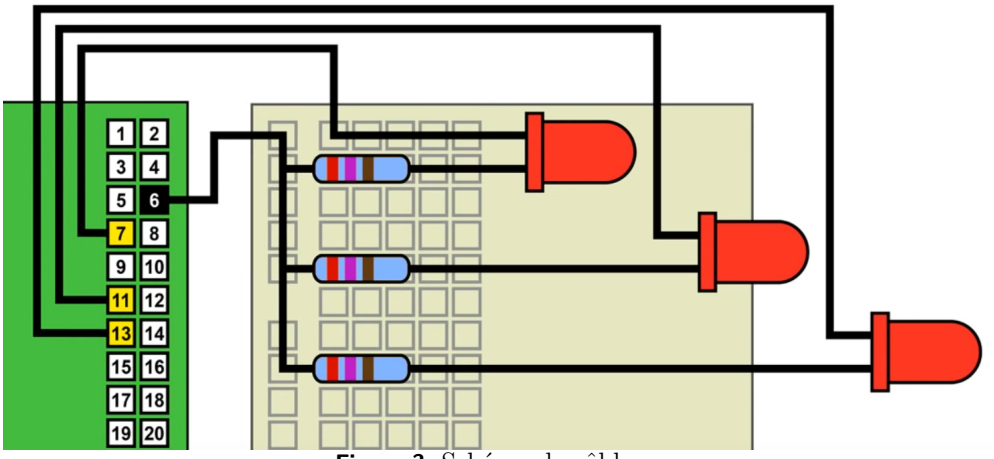


Figure 3. Schéma de câblage

Partie 2 : solution

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)   # Mode de numérotation des pins.
GPIO.setup(7,GPIO.OUT)     # Sortie
GPIO.setup(11,GPIO.OUT)    # Sortie
GPIO.setup(13,GPIO.OUT)    # Sortie

for x in range(0,6):
    GPIO.output(7,True)
    time.sleep(0.5)
    GPIO.output(7,False)
    GPIO.output(11,True)
    time.sleep(0.5)
    GPIO.output(11,False)
    GPIO.output(13,True)
    time.sleep(0.5)
    GPIO.output(13,False)

GPIO.cleanup()
```

Partie 3

On veut réaliser un code en Python pour faire clignoter une LED et commander son fonctionnement avec un bouton poussoire.
La figure 4 montre le montage à réaliser.
Remarque : On peut utiliser un fil (branché ou débranché) pour remplacer le bouton poussoire.

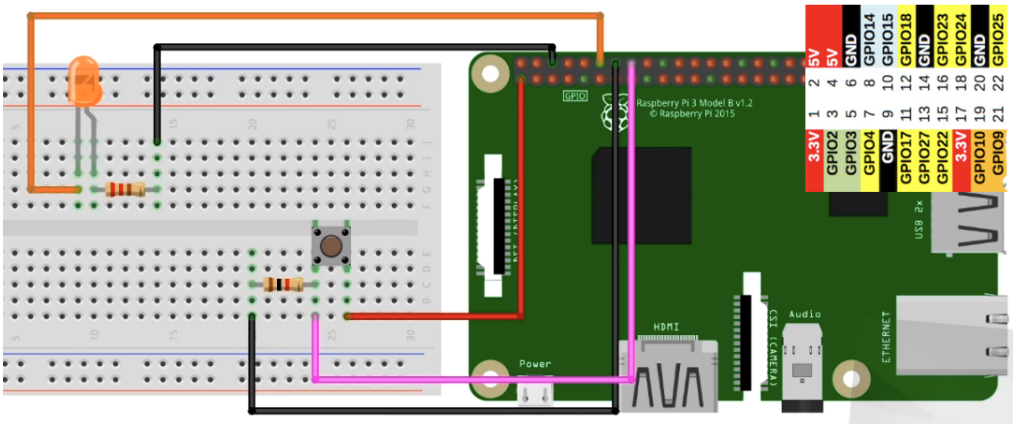


Figure 4. Schéma de câblage

Partie 3 : solution

```
import RPi.GPIO as GPIO
import time

LED = 18 # Pin N  18
BUTTON = 23 # Pin N  23

GPIO.setwarnings(False) # Ne pas afficher les erreurs
GPIO.setmode(GPIO.BCM) """ Mode de numérotation des pins.
          GPIO18 en mode BCM correspond
          à 12 en mode BOARD : GPIO.setmode(GPIO.BOARD) """
GPIO.setup(LED,GPIO.OUT)     # Sortie
GPIO.setup(BUTTON,GPIO.IN)   # Entrée

mode_auto = 0
last_button_statement = 0

try:
    while True:
        if GPIO.input(BUTTON) == 1 and last_button_statement != 1:
        """ Si l'état du bouton est à 1 et
            le dernier état du bouton est différent de 1 """
            if mode_auto == 1:
                mode_auto = 0
                print("Non automatique")
            else:
                mode_auto = 1
                print("Automatique")
            print("Bouton \n")
            last_button_statement = 1
        elif GPIO.input(BUTTON) == 0:
            last_button_statement = 0

        if mode_auto == 1:
            GPIO.output(LED,GPIO.HIGH)
            time.sleep(1)
            GPIO.output(LED,GPIO.LOW)
            time.sleep(1)
except KeyboardInterrupt: # Programme arrêté par le bouton stop
        GPIO.output(LED,GPIO.LOW)
        GPIO.cleanup()
        print("\n Fin du programme")
```

Partie 4

On veut réaliser un code en Python pour récupérer l'état d'un interrupteur (ouvert ou fermé). Á cet effet, on utilise le montage de la figure 5.

Remarque : On peut utiliser un fil (branché ou débranché) pour remplacer l'interrupteur.

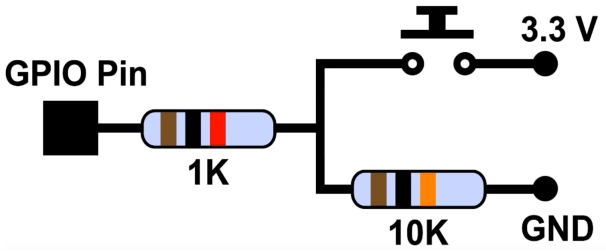


Figure 5. Circuit

Partie 4 : solution

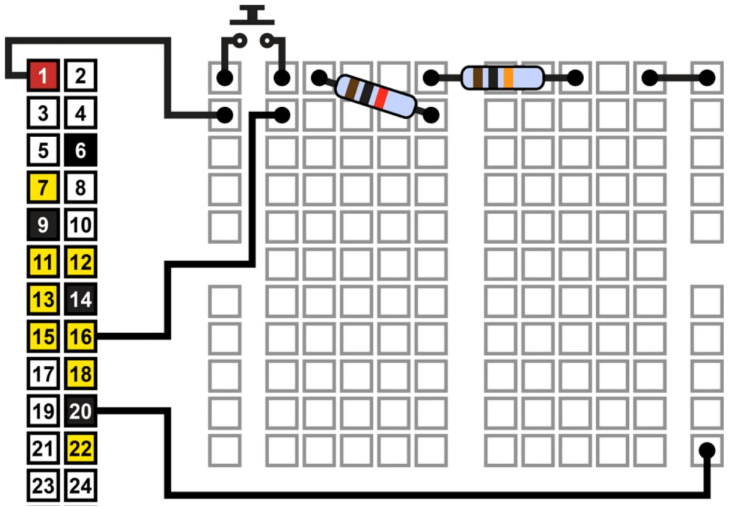


Figure 6. Schéma de câblage

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)  # Mode de numérotation des pins.
GPIO.setup(16,GPIO.IN)    # Entrée

try:
    while True:
        if GPIO.input(16) == 0:
            print("Interrupteur ouvert")
        else:
            print("Interrupteur fermé")

finally:
    GPIO.cleanup()

# CTR C pour arrêter le programme
```

Partie 5

Écrire un programme en Python qui demande à l'utilisateur d'appuyer sur un interrupteur et affiche le message `Merci` dès qu'on actionne l'interrupteur. Á cet effet, il faut réaliser le circuit.
Remarque : On peut utiliser un fil (débranché) pour remplacer l'interrupteur.

Partie 5 : solution

```
import RPi.GPIO as GPIO
# Identique au montage précédent
GPIO.setmode(GPIO.BOARD)  # Mode de numérotation des pins.
GPIO.setup(16,GPIO.IN)    # Entrée

print("Appuyer sur le bouton")

while GPIO.input(16) == 0:
    pass

print("Merci")

GPIO.cleanup()
```

Partie 6 : Enregistrer des données dans un fichier en Python

Le but de cette partie est de pouvoir récupérer des données (à partir d'un capteur de température par exemple) et de les stocker dans un fichier. Cela permettra de pouvoir utiliser ce fichier pour en créer un graphique par exemple.
Pour nous, on va utiliser la récupération de la température du Chipset du Raspberry afin de la stocker dans un fichier.
Le programme doit récupérer toutes les minutes la température du Chipset et doit créer une entrée dans un fichier ; ce fichier contiendra les entrées suivantes : la date, l'heure et la température.

Partie 6 : solution

```
import os
import time

cmd = '/opt/vc/bin/vcgencmd measure_temp'

while True:
    result = os.popen(cmd).readline().strip()
    # le résultat est temp=xx.x'C , donc on traite le résultat
    # pour ne garder xx.x
    temp = result.split('=')[1].split("'")[0]
    f = open('/tmp/temp_log.dat','a')
    print >>f,("%s,%s"%(time.strftime("%d-%m-%Y %H:%M:%S"), temp))
    f.close()
    time.sleep(60)
```

4.2 TP 2 : Piloter les ports GPIO à partir d'un navigateur

Le but de ce TP est de pouvoir piloter un port GPIO du Raspberry à partir d'un téléphone ou d'un iPad. Pour cela, on va utiliser un module `web.py`. Ce module contient un serveur Web.
L'installation de ce module est réalisée par les étapes suivantes :

- Télécharger le module en tapant l'adresse
  http://webpy.org/static/web.py-0.37.tar.gz
- Le décompresser en utilisant les commandes
  ```
  $ cd /home/pi/Téléchargements
  $ tar -xzvf web.py-0.37.tar.gz
  $ cd web.py-0.37
  ```
- Se positionner dans le répertoire et lancer la commande
  ```
  python setup.py install
  ```

Maintenant, il va falloir créer une arborescence :

```
$ cd /
$ mkdir webpyserver
$ cd /webpyserver
$ mkdir templates
$ mkdir static
```

Le dossier `webpyserver` contiendra le fichier Python à exécuter ;
le dossier `webpyserver/templates` contiendra une page HTML ;
et le dossier `webpyserver/static` contiendra une feuille de style au format CSS.
On va créer un programme en Python nommé `gpio4.py` qui permettra d'allumer ou d'éteindre une LED en utilisant le port GPIO4 du Raspberry. Ce fichier sera enregistré dans le dossier `webpyserver`, dont voici le contenu :

```
#!/usr/bin/env python
import web, wiringpi
from web import form

# définit GPIO4 en sortie
io = wiringpi.GPIO(wiringpi.GPIO.WPI_MODE_SYS)
io.pinMode(4,io.OUTPUT)
# définit la page de nom index pour le site web
urls = ('/', 'index')
dossier_web = web.template.render('templates')

app = web.application(urls, globals())

# définit les boutons à afficher
ma_forme = form.Form(
  form.Button("btn", id = "btnon", value = "on", html = "On", class_ = "bouton_on"),
  form.Button("btn", id = "btnoff", value = "off", html = "Off", class_ = "bouton_off")
  )
# définit l'action à effectuer quand la page index est appelée?
class index:
    # utilisé quand la page est demandée
    def GET(self):
        forme = ma_forme()
        return dossier_web.index(forme, "Raspberry Pi control GPIO4")

    # utilisé quand une forme web est soumise
    def POST(self):
        userdata = web.input()
        if userdata.btn == "on":
            io.digitalWrite(4,io.HIGH)
        if userdata.btn == "off":
            io.digitalWrite(4,io.LOW)
        # recharge la page web
        raise web.seeother('/')
# programme
if __name__ == '__main__':
    app.run()
```

On crée un fichier `index.html` qui sera enregistré dans le dossier templates :

```
$def with (form, title)
<!doctype html>
<html>
    <head>
        <title>$title</title>
    </head>

    <body>
        <br />
        <form class="form" method="post">
        $:form.render()
        </form>
    </body>
</html>
```

Il faut lancer la commande `$ python gpio4.py` pour démarrer le serveur Web, puis à partir d'un navigateur taper l'adresse IP du Raspberry en utilisant le port 8080 ou plus simplement : `$ raspberrypi :8080`

4.3 TP 3 : Allumer une LED avec PiGPIO (C)

Ecrire un code en C pour allumer une LED pendant 10s puis l'éteindre.