
Travail pratique 2 – Partie 2

« Initiation aux circuits avec Raspberry Pi et Python»

420-555-AL
Programmation appliquée aux objets connectés

Note: / 100

**À terminer le mardi 7 octobre**

## Objectifs de l’évaluation

Dans ce projet, vous allez utiliser votre Raspberry Pi pour contrôler un circuit à l'aide de Python. Vous vous concentrerez uniquement sur les sorties, sans effectuer de lectures sur les composants. Pour ce faire, vous aurez besoin de LEDs, de résistances, d'avertisseurs sonores et d'un interrupteur. Ce projet se divise en quatre parties :

A. Validation de l’entrée utilisateur : un nombre signé ou une phrase.
B. Encodage d'un nombre en sa représentation binaire signée en complément à deux.
C. Encodage d’un texte en BRAILLE en utilisant une approche sono-lumineuse.
D. Création d’une interface graphique pour contrôler votre circuit.

## Directives importantes

- Ce travail pratique **peut** être réalisé **en équipe de deux.**
- Divisez clairement vos platines en régions distinctes correspondant aux sections A, B et C du travail
- Modalités de remise : le mardi 7 octobre septembre avant la fin du cours, vous devez faire corriger votre platine par l’enseignant. Ensuite, le soir même avant minuit, remettez votre programme Python et votre circuit Fritzing sur Github Classroom dans répo de travaux.

## Section A : Validation de l'entrée (20 points)

Matériel : un avertisseur sonore

Votre programme Python ne doit accepter que l'une OU l'autre des deux chaînes suivantes:

- une suite de caractères minuscules (incluant l'espace)
- un entier signé en format décimal

Tous les trente-sept (37) caractères acceptés sont montrés par le regex suivant :

[a-z0-9\-]

à laquelle on ajoute l'espace, donc trente-huit (38) caractères en tout.

Lorsque l'usager entre un nombre, ce nombre doit être compris entre -128 et +127. Voici des exemples d'entrées valides et invalides :

| **Valides** | **Invalides** |
|---|---|
| gomme a macher | 124.7 ( le point) |
| un deux trois | Un 2 trois (le U et le 2) |
| -24 | 128 (trop grand) |
| 0 | pédant (accent) |

Dans le cas où l'entrée est valide :

- si c'est un nombre : exécute la section B (voir plus loin)
- si c'est une suite de caractères : exécute la section C (voir plus loin)

Dans le cas où l'entrée est invalide :

- le programme affichera un message pertinent à l'écran
- le programme fera sonner un avertisseur sonore pendant une seconde
- le programme attendra une nouvelle entrée de l'usager

Exemple d'interaction dans la console :

> Quelle est votre entrée? **124.7** *Affiche un message et sonne l'avertisseur pour 1 sec.*
> Quelle est votre entrée? **11** *Exécute la section B sur votre platine*
> Quelle est votre entrée? **chat** *Exécute la section C sur votre platine*

## Section B : Encodage d'un nombre décimal en binaire signée

Matériel : huit LEDs et une RGBLED.

Si la donnée entrée est un nombre signé, vous devez commencer par calculer la représentation binaire du nombre décimal reçu. Cette représentation binaire sera ensuite affichée sur huit LED alignées, chaque LED symbolisant un bit de l'octet en complément à deux (pour un nombre négatif).

Un « 1 » sera représenté par une LED allumée, un « 0 » par une LED éteinte. Par exemple, le nombre décimal *11* (onze) sera affiché comme *00001011* dans la console et comme ceci sur votre platine :

Le nombre *-127* sera affiché comme *10000001* sur votre écran et comme ceci sur votre platine :

Ces affichages seront d'une durée de cinq secondes avant de s'éteindre.

De plus, les trois bits de poids forts (ceux à gauche) seront aussi affichés en utilisant une seule LED pleine couleur (RGBLED). Les couleurs utilisées pour ces trois bits sont les suivantes :

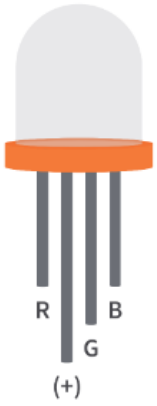
| 3 derniers bits | couleur |
|---|---|
| 111 | éteinte |
| 110 | bleu |
| 101 | vert |
| 100 | cyan |
| 011 | rouge |
| 010 | magenta |
| 001 | jaune |
| 000 | blanc |

Fonctionnement d'une RGBLED :

- Alimentation de 3.3 volts
- C'est un composant à **anode commune** :
  - Les trois broches pour les couleurs ont chacun leur branchement GPIO (n'oubliez pas les résistances)
  - L'anode est branchée sur le 3.3 volts
- Configuration de l'objet gpiozero :
  - Le paramètre *active_high* doit être à false

Connection test ici : attention cette connexion test utilise une RGBLED à *cathode commune*.

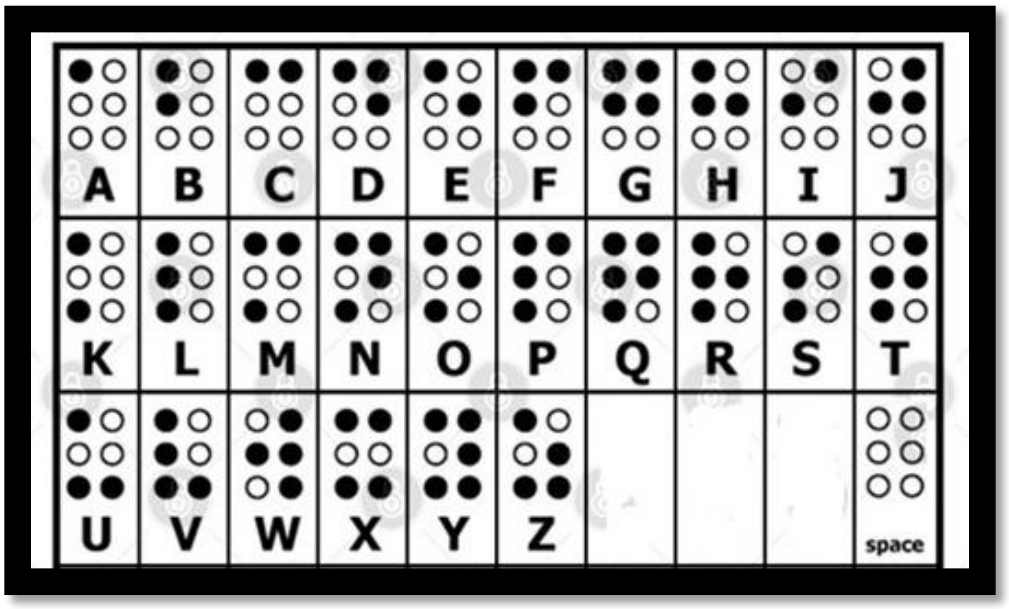

RGB Led



## Section C : Encodage d'une phrase en Braille

Matériel : six LEDs, un avertisseur sonore et un interrupteur.

Dans le cas où la donnée entrée est un mot ou une phrase, vous devez l'encoder en Braille. L'encodage que nous allons utiliser permet de communiquer une phrase sous la forme de signaux lumineux[1] et/ou de sons. Dans ce code, les lettres sont encodées à l'aide de points dans une grille de trois lignes par deux colonnes :

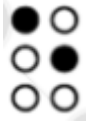
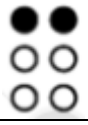
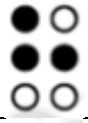
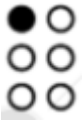



Votre circuit possèdera les caractéristiques suivantes :

- Chaque symbole est représenté par une grille de trois par deux LEDs (couleur à votre choix).
  - Par exemple, la lettre « C » est donc encodée avec les deux LEDs sur la première ligne allumée et les autres éteintes.
- Chaque symbole est aussi encodé avec une trame sonore relié à cette grille :
  - En parcourant la grille de gauche à droite et de haut en bas, un point noir fera bipper l'avertisseur sonore (intervalle de 0.1 seconde) durant une demie seconde alors qu'un point blanc l'activera pendant une demie seconde.
- Pour séparer les lettres, les six LEDs et l'avertisseur seront désactivés (éteints) durant une seconde.
- Pour séparer les mots, autrement dit lorsqu'on rencontre un espace, les six LEDs et l'avertisseur seront désactivés (éteints) durant deux secondes.

[11] Inutile pour une personne avec handicap visuel, mais nous ferons l'hypothèse que chaque signal lumineux s'accompagne d'une protubérance de la LED, palpable au touché.

La phrase « LE CHAT » est donc encodée comme suit:

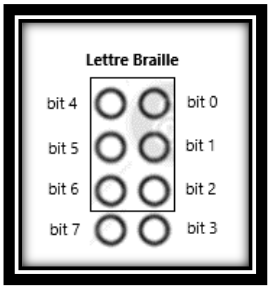
| Symbole | LEDs | Avertisseur | Secondes |
|---|---|---|---|
| L |  | Bip 0.5 sec Actif 0.5 sec<br>Bip 0.5 sec Actif 0.5 sec<br>Bip 0.5 sec Actif 0.5 sec | 0-3 secondes |
| Inter-lettre |  | Éteint | 3-4 secondes |
| E |  | Bip 0.5 sec Actif 0.5 sec<br>Actif 0.5 sec Bip 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec | 4-7 secondes |
| Espace (inter-mot) |  | Éteint | 7-9 secondes |
| C |  | Bip 0.5 sec Bip 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec | 9-12 secondes |
| Inter-lettre |  | Éteint | 12-13 secondes |
| H |  | Bip 0.5 sec Actif 0.5 sec<br>Bip 0.5 sec Bip 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec | 13-16 secondes |
| Inter-lettre |  | Éteint | 16-17 secondes |
| A |  | Bip 0.5 sec Actif 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec<br>Actif 0.5 sec Actif 0.5 sec | 17-20 secondes |
| Inter-lettre |  | Éteint | 20-21 secondes |
| T |  | Actif 0.5 sec Bip 0.5 sec<br>Bip 0.5 sec Bip 0.5 sec<br>Bip 0.5 sec Actif 0.5 sec | 21-24 secondes |

Finalement, vous ajouterez un interrupteur pour activer/désactiver l'avertisseur sonore. Cet interrupteur ne sera pas contrôlé par votre programme Python, il le sera seulement manuellement.

| | | |
|---|---|---|
| Un interrupteur :<br><br>• La seule fonction de cet interrupteur pour ce travail est de désactiver/débrancher l'avertisseur sonore |  |  |

## OPTIONNEL : partage des LEDs pour les circuits B et C

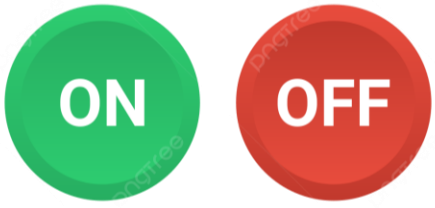
Pour éviter la multiplicité de LEDs sur votre platine, vous pouvez les partager de la façon suivante :




De cette façon, vous n'aurez besoin que de huit LEDS en tout, au lieu de quatorze (huit pour le circuit B + six pour le circuit C).

## Section D : Interface graphique (20 points)

Fournissez une interface graphique de base (Turtle ou autre) avec deux boutons semblables à :




En pressant sur ON : toutes les LEDS et avertisseurs sur votre platine seront allumés.

En pressant sur OFF : toutes les LEDS et avertisseurs sur votre platine seront éteints.

Vous aurez besoin de threads pour exécuter les deux tâches suivantes simultanément :

1. Votre boucle d'interaction avec l'usager dans la console
2. La gestionnaire d'événement pour les clics sur les boutons

Vous pouvez vous inspirer du code Python suivant pour la manipulation des threads. Ce code est fourni avec l'énoncé.

```python
# Gestion d'événements (clics et touches) avec threading: exempleThreading.py
from turtle import *
import threading, keyboard
from time import sleep
tortue = Turtle()
led = True # Simulation d'une led physique
enCours = True # Le thread sans événement
def attrapeClic(x, y): # x et y sont les coordonnées du clic
    global led
    if x >= -50 and x <= 50 and y >= -50 and y <= 50:
        led = not led
# On déclare quelle méthode va attraper les clics gauches
onscreenclick(attrapeClic,1) # 1: clic gauche
# On dessine un rectangle en plein centre
tortue.penup()
tortue.goto(-50, -50)
tortue.pendown()
for _ in range(4):
    tortue.forward(100)
    tortue.left(90)
tortue.hideturtle()
# Définir une fonction sans événement qui tournera dans son propre thread
def autre_tacheSansEvenement():
    global enCours
    while enCours:
        print("La led est " + str(led))
        sleep(1)
# Définir une fonction avec événement qui tournera dans son propre thread
def autre_tacheAvecEvenement():
    global enCours
    while True:
        if keyboard.is_pressed('q'):
            print("La touche 'q' a été appuyée !")
        if keyboard.is_pressed('esc'):
            enCours = False # Arrête le thread sans événement
            bye()  # Fermer la fenêtre turtle proprement, donc arrête onscreenclick()
            print("Programme terminé.")
            break # Arrête le thread avec événement
        sleep(0.1)  # Pour éviter une utilisation excessive de CPU
# Créer les thread pour deux autres tâches
autre_tache_threadSansEvenement = threading.Thread(target = autre_tacheSansEvenement)
autre_tache_threadAvecEvenement = threading.Thread(target = autre_tacheAvecEvenement)
# Démarrer les thread
autre_tache_threadSansEvenement.start()
autre_tache_threadAvecEvenement.start()
mainloop() # pour garder le programme actif
```

| BARÈME | Fonctionnement | Source Python | Circuit Fritzing | Total |
|---|---|---|---|---|
| **Section A : validation** | **/10** | **/5** | **/5** | **/20** |
| **Section B : encodage d'un nombre** | **/14** | **/8** | **/8** | **/30** |
| **Section C : encodage d'une phrase** | **/14** | **/8** | **/8** | **/30** |
| **Section D : interface graphique** | **/12** | **/8** | | **/20** |
| ***Commentaires et note finale*** | | | | |