Chapitre N°5.) Manipulations directes des entrées/sorties (GPIO)

Ce que vous allez apprendre :

- ✓ Piloter les broches d'entrée-sortie du Raspberry Pi
- ✓ Effectuer des montages électroniques à base de leds, BP
- ✓ Python :
  - Utiliser des boucles et range()
  - intercepter les exceptions avec try
  - écrire des fonctions

Sommaire :

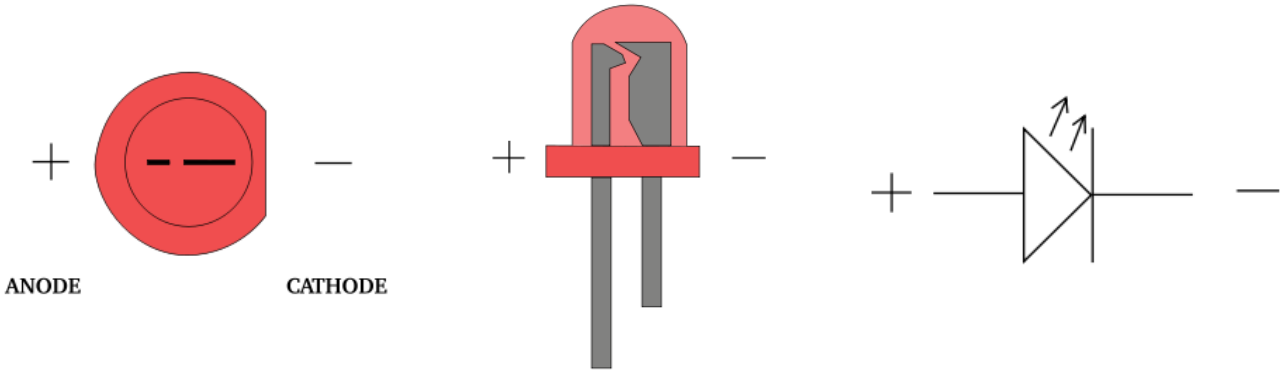
I. Matériels électroniques nécessaires

1. Une led

a. Utilisation d'une LED

Identifier la led

L'anode doit être branchée au potentiel le plus élevé, est la branche la plus longue et la cathode.



Trois « astuces » :

1. Si les branches sont coupées, à regarder de près, à l'intérieur de la led, l'anode est plus petite que la cathode (l'inverse des broches)
2. Pour reconnaître la Cathode, en allemand elle s'appelle Kathode comme le K qu'elle représente 
3. Ou C comme courte pour retrouver la longueur de la patte

Pourquoi mettre une résistance ou pas ?

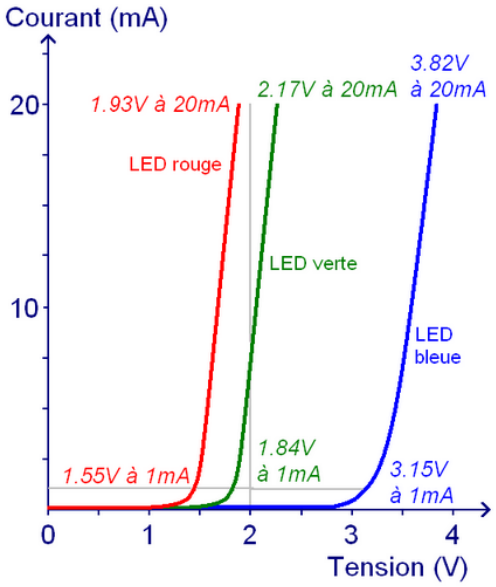
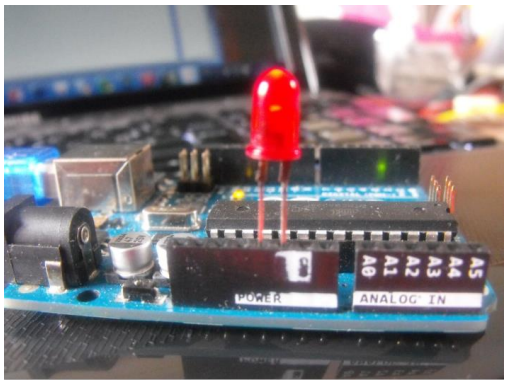


Figure 1 caractéristique d'une led rouge, verte et bleue

Le courant par rapport à la tension dans une led n'est pas linéaire : une petite variation de la tension engendre une importante variation de courant.

Il faudrait donc limiter le courant, non ?

Sur une Arduino par exemple, on peut mettre une diode sans résistance, cette carte débite pas plus de 40mA et une led peut supporter jusque 60 mA. Une seule pas plusieurs, une grosse et pas très longtemps. Dans le cas contraire, la carte ne pourra pas débiter suffisamment de courant et il y en a trop pour une broche …

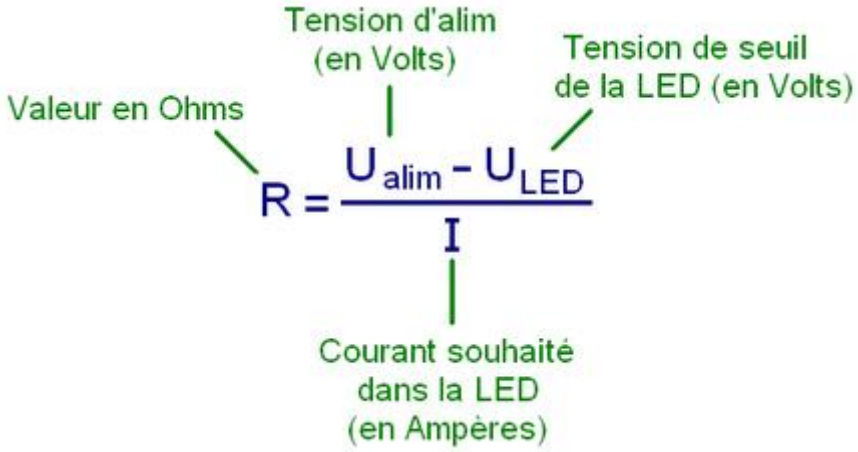


Led sans résistance

Le mieux, pour augmenter la longévité de la carte, est d'en mettre une résistance, assez grosse mais suffisante pour allumer la led.

Calcul de la résistance à mettre

Par habitude on prendra une résistance de 220 Ω en série avec la led, en sachant que l'Arduino sort en 5 Volts et la tension de seuil de d'une led rouge est de 1.8 volt, le courant vaut :



Application numérique pour une rouge V=1.93 et I=20mA d'où R=(5-1.93)/20=153 Ω.

On pourrait dire que la valeur supérieure et la plus commune est 220 Ω.
Mais si on prend 330 Ω : I = (5-1.93)/330 = 9.3 mA, ce qui est encore suffisant pour l'allumer

Conseil : le plus important c'est que la led s'allume, prendre la résistance la plus grande est le meilleur des choses pour ne pas réduire la longévité de la carte et pouvoir utiliser d'autres capteurs/actionneurs

2. bouton poussoir (interrupteur)

On peut s'attendre trouver un BP avec deux broches, en général il y en a quatre :

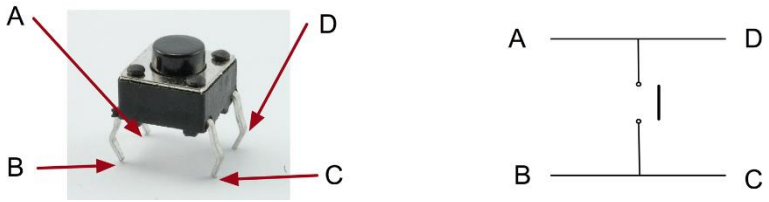
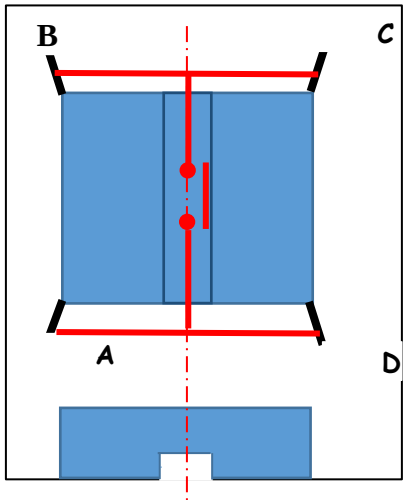


Figure 2 : les broches d'un BP

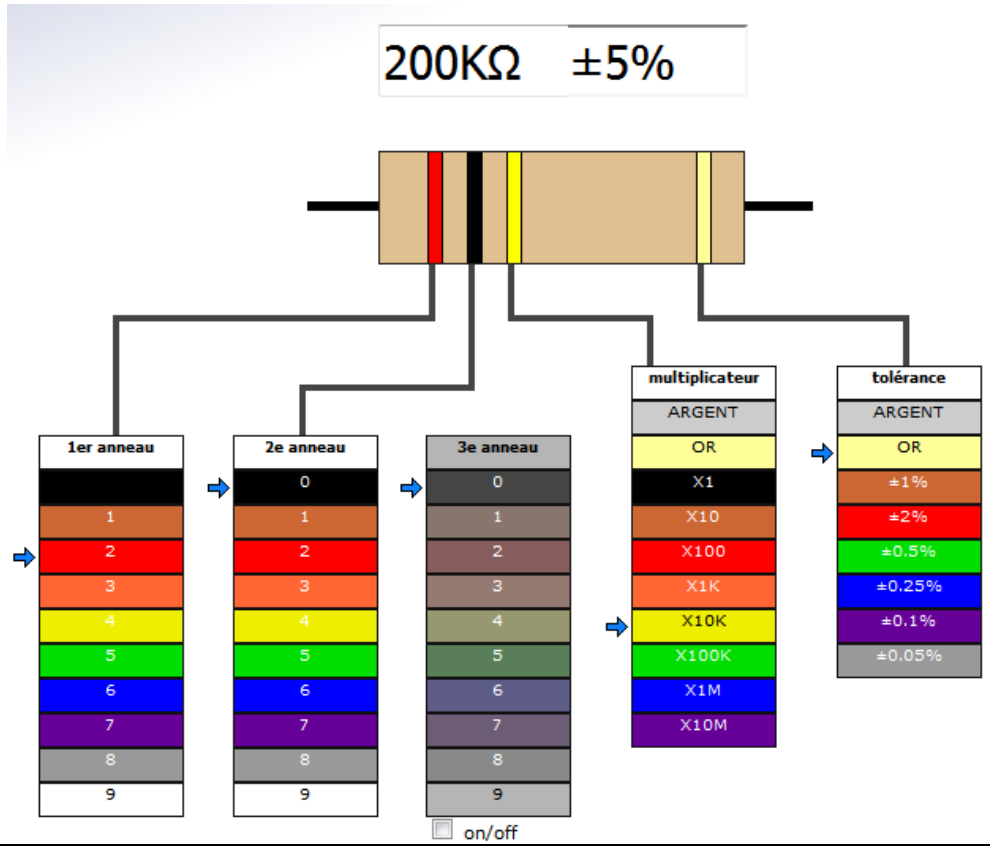


Les broches A et D sont connectées ensemble, de même pour B et C.

Comment les reconnaitre ? Il suffit de le retourner

3. Résistance

Les codes couleurs (cas des résistances 4 bandes) d'après http://www.electronique-radioamateur.fr/elec/composants/resistance-code-couleurs.php
:



Ainsi pour une 220 Ω.

1er anneau : rouge donc 2
2ème anneau rouge donc 2
3ème anneau : marron multiplicateur donc *10
4ème anneau : or, tolérance de 5%
D'où 22*10 = 220 Ω

Ainsi pour une 330 Ω.

1er anneau : orange donc 3
2ème anneau orange donc 3
3ème anneau : marron multiplicateur donc *10
4ème anneau : or, tolérance de 5%
D'où 22*10 = 220 Ω

Marron : 1
Noir : 0 -> 10 k Ω
Orange : *1k

4. BreadBoard : La plaque d'essais sans soudure :

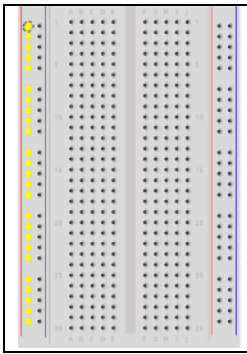
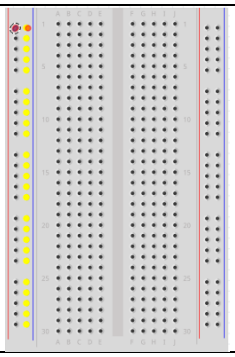
a. Présentation

Une breadboard permet de disposer des composants et de les relier entre eux à l'aide de cavaliers flexibles.
Ces plaques sont utilisées pour effectuer des prototypage rapide.
On y trouve des petits « trous » dans lesquels on va enficher composants et câbles.
Les colonnes et rangées portent des noms :

- Des colonnes A, B, C … à gauche et à droite
- Une séparation
- Une ligne rouge (+) et bleue (-) sur les bords de la plaque
- Des lignes 1,2,3,4 …

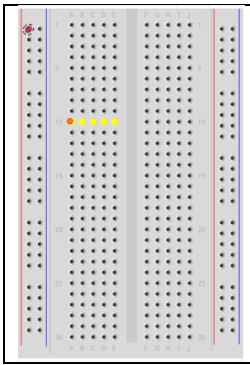
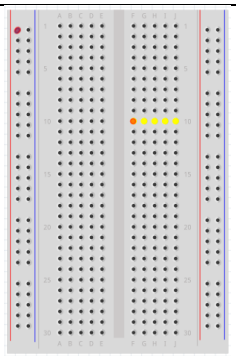
Ils sont reliés entre eux soit en colonne soit en ligne :

b. Les colonnes + et -

La colonne + (Vcc / alimentation)		La colonne – (masse)	
	Elle est repérée par une ligne rouge verticale à sa gauche. Tous les points de cette colonne sont reliés entre eux.		Elle est repérée par une ligne bleue verticale à sa droite. Tous les points de cette colonne sont reliés entre eux.

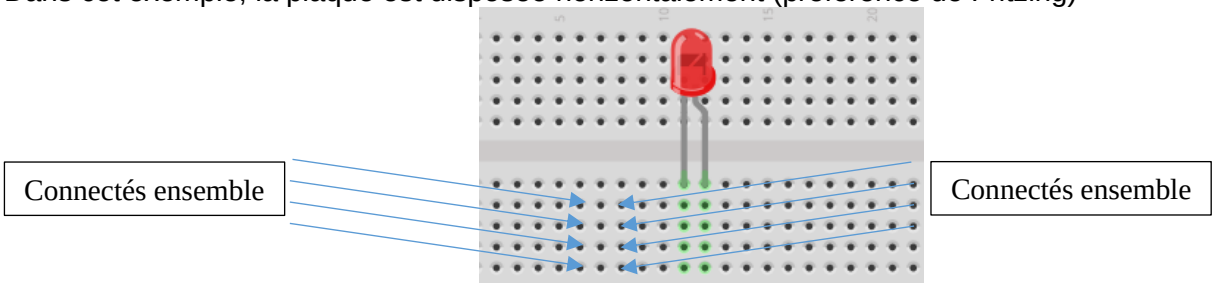
c. Les demi-lignes

De part et d'autre d'une séparation, se trouve une demi-ligne (ici 5 points)

		⚠ Attention, les demi-lignes de chaque côté de la séparation ne sont pas connectées entre-elles.

d. Brancher un composant

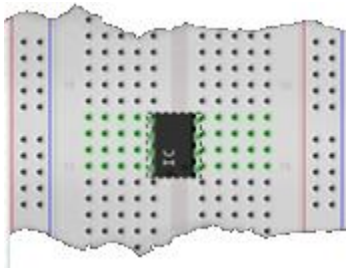
Dans cet exemple, la plaque est disposée horizontalement (préférence de Fritzing)



Chaque patte de la diode est accessible par 4 points différents.

e. Comment brancher un composant avec plusieurs broches

La solution est de brancher le composant sur la séparation entre les deux groupes de colonnes :



Chaque broche dispose ainsi de 4 points à gauche et à droite pour effectuer des liaisons.

5. Les straps

Les straps – fils d'essais- permet de connecter l'Arduino à la breadboard. Les broches du micro-contrôleur et du le carte d'essai étant femelles, les straps seront donc des mâles/mâles :

6. Plaquette d'indentification des entrées/sorties

Un label pour repérer les broches d'entrées-sorties :

II. Du côté logiciel :

1. IDE python

Il existe sur la raspberry trois environnement de développement permettant la programmation Python.
Du plus complet au plus simple d'utilisation :

Geany

Thonny Python IDE

mu

A vous de choisir !!!

2. Fritzing

Fritzing est un logiciel qui vous permettra de dessiner votre câblage. Certes un peu vieux, mais il n'existe aucune autre alternative.

La version utilisée se trouve à l'adresse : . Décompressez-la dans un répertoire de votre choix.

III. Comment rédiger un compte-rendu ?

1. Pour les questions

Bien sûr répondre aux questions mais n'oubliez pas de la rappeler et identifier correctement la partie (**III. Les leds**, 2. Montage avec deux leds …)

2. Pour les exercices et problèmes :

Au tout début, il faudra rappeler le but.

3. Câblage

- Les illustrations des montages devront être réalisées avec fritzing : l'identification exacte des broches et des composants. Mettez les fils de couleurs différentes.
- Le réel montage devra être présenté par une photo.

4. Le code

- Le code doit être commenté.
- Les noms de variables convenablement choisis : pas de x, y ou z, sauf pour celles de boucles (i, j, k…)
- Si le code effectue plusieurs ensemble d'actions (trois ?), il faudra veiller à utiliser des fonctions.
- Le mettez jamais de numéro de broches, de taille de tableaux … directement dans votre code. Utilisez à la place des variables au tout début.
- Même si le langage le permet (Python), initialisez les variables au début de votre programme avec des commentaires.
- Montrez le résultat du programme avec une copie d'écran.
- Utilisez une coloration syntaxique pour mettre votre code dans le C.R. (cf. https://lc.cx/coloration)
- Pas de copie d'écran du code

5. La forme du C.R.

Au tout début : nom et date.
Pour le dépôt, respectez le format demandé.
Si le format est un document .docx, le professeur mettra ses remarques en utilisant la fonction commentaire de word.
Il faut rédiger ! (pas comme cette partie)
Faites un effort sur la forme : couleurs, titres sans oublier la pagination

6. Utilité d'un C.R.

Pour l'enseignant : permettre l'évaluation des compétences acquises par l'élève, ce qu'il sait faire ou non
Pour l'étudiant :

- Développer les compétences rédactionnelles nécessaires pour votre futur métier
- Faire le point sur les acquis
- Avoir une trace écrite de travail réalisé pour pouvoir vous y référer plus tard

Pour finir, pas besoin de recopier celui de votre voisin : vous ne pourrez jamais entièrement le maîtriser et en plus vous n'aurez rien appris. C'est votre travail qui m'intéresse !!!

7. Grille de notation

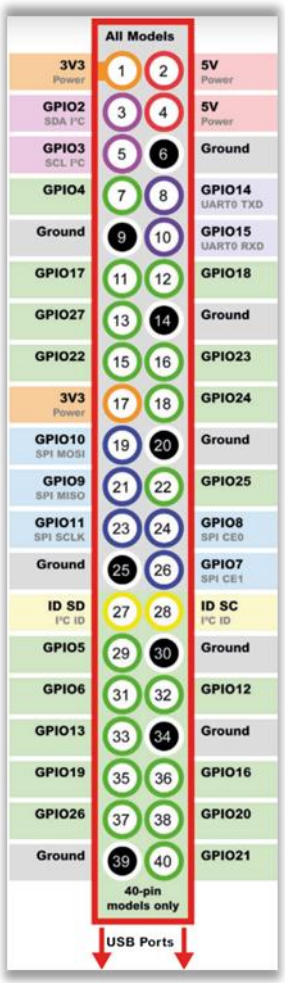
Points (pourcentage)	Critère
10	Forme
20	Réponses aux questions
10	Rédaction
20	qualité de schémas + explications
10	Forme du code : variables, commentaires …
30	Bonnes réponses aux problèmes

IV. Les GIOP

Des GPIO (General Purpose Input/Output) sont disponibles sur le connecteur de la Raspberry Pi. Ce qui permet de l'interfacer avec des circuits électroniques.
Attention : l'Arduino fonctionne en 5 volts, **tandis que** la rasperry est à 3.3 volts.

En mode GPIO ces broches peuvent être configurées soit en entrée (binaire) soit en sortie.

1. Le brochage (pinout) pour le B+/PI2/PI3

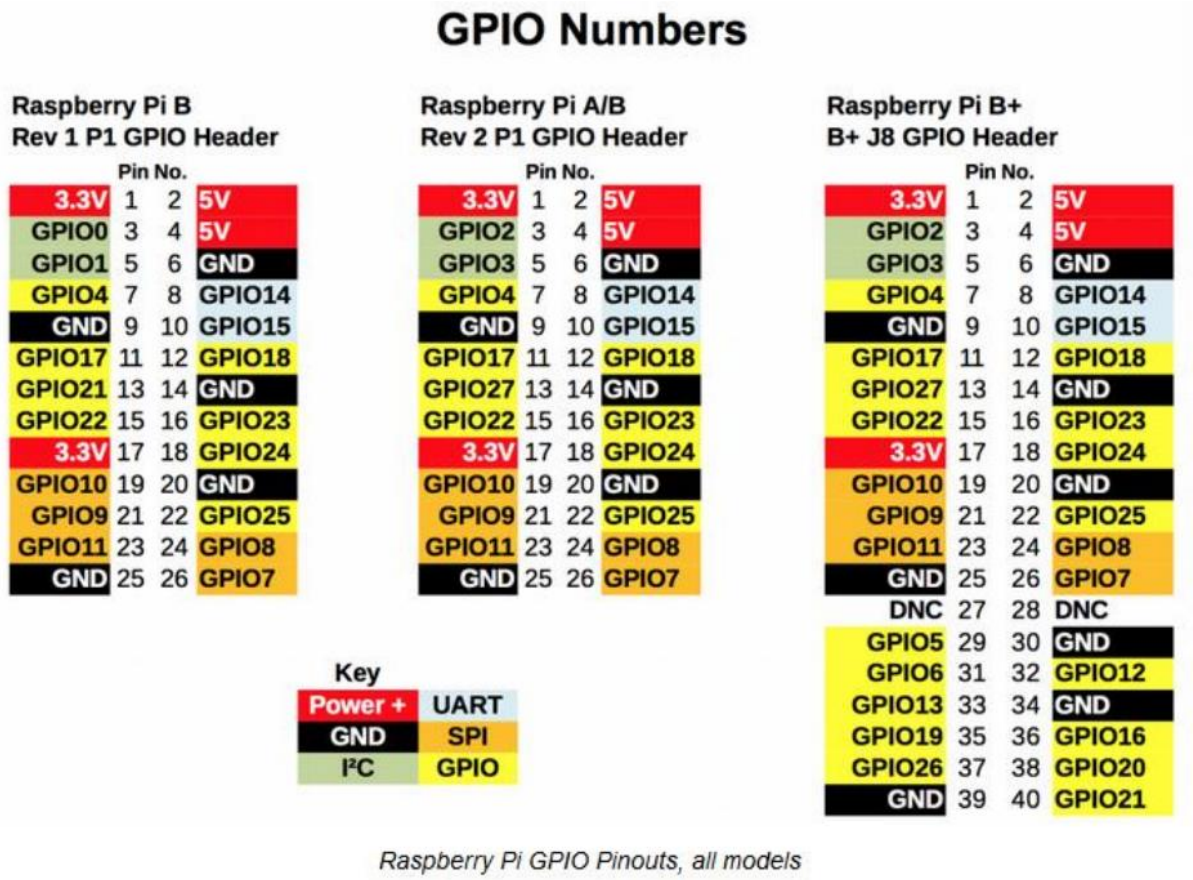


Il existe six types de GPIOs :

1. **5V** ou **3.3V**
2. La **masse**
3. Le port série, **UART**, en bleu. Le **TX** est le GPIO 14, **RX** le 15
4. Le protocole **SPI MOSI GPIO** 19, **MISO** GPIO 21
5. Le protocole **I2C** (27 et 28)
6. Les autres GPIO

GPIO Zero utilise le système de numération **BCM** (Broadcom SOC channel) pour identifier les broches.
Il s'agit des numéros référencés à l'extérieur du schéma, comme 18 pour GPIO18.
Les numéros de la plaque ceux qui se trouvent à l'intérieur). Vous pouvez utiliser toutes les broches marquées GPIO, mais celles en bleu cumulent d'autres fonctionnalités.
C'est pourquoi vous utiliserez uniquement les broches marquées en vert pour ce tutoriel.

2. Tous les modèles



Raspberry Pi GPIO Pinouts, all models

3. Précautions à prendre

Les entrées/sorties sont fragiles et il n'est pas rare de détériorer une broche, voire toute la carte (8 depuis 3 ans).

Voici quelques recommandations pour une meilleure longévité de la carte

- Respecter la tension maximale de 3.3 volts sur une broche configurée en entrée
- 250 mA sur la broche 5V et 50 mA sur celle de 3.3V
- Au total pour toutes les broches, pas plus de 50 mA
- En ce qui concerne, certain conseille de limiter le courant à 3 mA et d'autres à 16mA

Ce dernier point pose problème, il est difficile de trouver la justification : certain préconise 3mA et donc pour allumer une led, une résistance de 470 Ω, d'autres 16mA (R=150Ω) et le magazine officiel effectue les montages avec R=50 Ω et même sans résistance !!!

Le mieux, pour augmenter la longévité de la carte, est d'en mettre une résistance, assez grosse mais suffisante pour allumer la led.

4. Exercice

Un montage qui fonctionne sur la RPI B Rev1 fonctionnera-t-il sur un B+ ? Et inversement ?

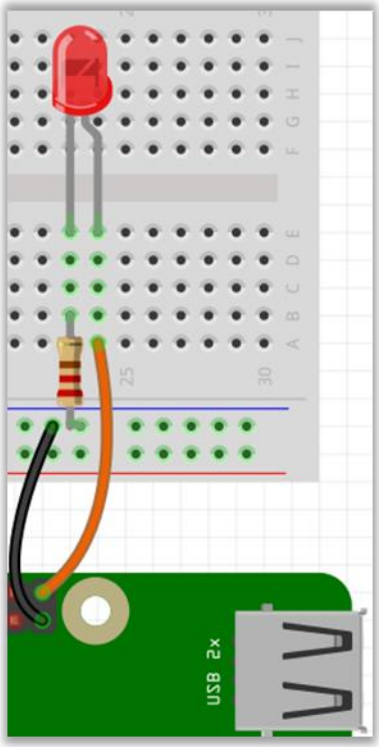
5. Branchement

Afin de mieux repérer les broches, vous allez utiliser une étiquette à mettre sur le Rpi disposé comme ci-dessus :

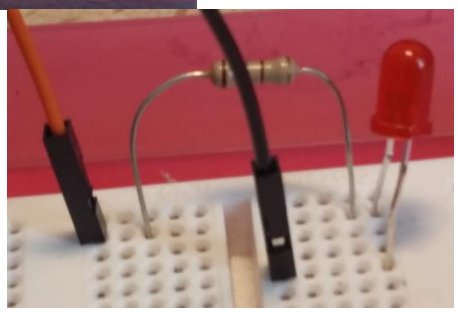
V. Premiers montages avec les Leds

1. Une simple Led

Le montage à réaliser est le suivant :

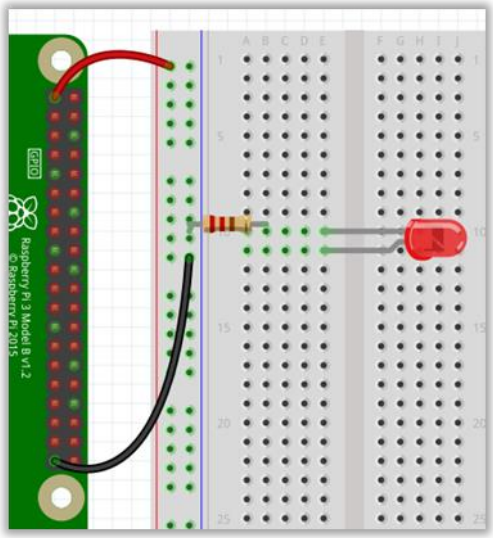


Avec la résistance qui vous avez choisie, quelle est la valeur du courant dans la diode ? La couleur est-elle importante ? Pour rappel, la carte raspberry Pi fonctionne en 3.3 volts.



Effectuez le montage cathode sur la masse et anode sur le GPIO21.

TRUC : Quelques fois pour vérifier le bon sens (cathode/Anode) je réalise ce montage :



En prenant le 3.3 V en haut à droite
Et la masse en bas à droite.

2. Programmation des GPIOs sans gpiozero

a. Exemple

```
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO  # Importer la bibliothèque et donner un synonyme
import time

GPIO.setmode(GPIO.BCM)  # Utiliser le système de numérotation BCM
led = 21  #  N°1

# Configuration des broches
GPIO.setup(led, GPIO.OUT)  # N°2 :

GPIO.output(led, 1)  #  N°3 :
time.sleep(1)  # N°4 :

GPIO.output(led, 0)  # N°5
time.sleep(1)

#Pour terminer « proprement »  remettre à zéro tous les ports
GPIO.cleanup()
```

Note : `# -*- coding: utf-8 -*-` permet d'utiliser les accents dans le code

Sans essayer le programme, que fait-il ? Finissez de le commenter

Code	Commentaire N°	Signification
`led = 21`	1	
GPIO.setup(led, GPIO.OUT)	2	
GPIO.output(led, 1)	3	
time.sleep(1)	4	
GPIO.output(led, 0)	5	

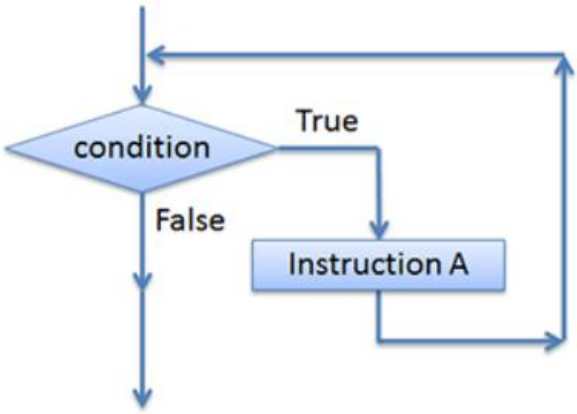
- Créez un répertoire de travail (par exemple /SuperTpsDeMrTomczak
- Utilisez un des ide (mu, Thonny ou geany)
- Testez le code avec le montage précédent

3. Elements algorithmiques en Python

a. Rappel boucle while :

Syntaxe :

```
while condition:
    Instruction A
```



Exemple de programme :

```
x = 1
while x < 10:
print("x a pour valeur", x)
x = x * 2
print("Fin")
```

b. une boucle infinie

Testez le code suivant. Que se passe-t-il si appui sur CTRL + C

```
import time
while True:
    print('coucou')
    time.sleep(1)
    print('pour sortir de la boucle infinie -> CTRL C')
    time.sleep(2)
```

c. Gestion des exceptions

La seule solution pour sortir d'une boucle infinie c'est d'appuyer sur CTRL+C vous avez le message d'erreur *KeyboardInterrupt :*

```
KeyboardInterrupt
```

Définition

Lorsqu'une erreur est détectée, le programme s'arrête et l'interpréteur python affiche un message.
En interceptant cette exception, il est possible d'afficher un message plus personnalisé et terminer proprement le programme.

Mise en œuvre

Les exceptions sont implémentées dans de nombreux langages de programmation.
Le code devra être entouré d'un bloc try pour capturer une éventuelle exception.

La syntaxe pour attraper toutes les exceptions est :

```
try:
    # ... instructions à protéger
except:
    # ... que faire en cas d'erreur
```

Exemples :

```
try:
    f = open(filename, "r")
```

```
except: # toutes les exceptions
    print("Le fichier", filename, "est introuvable")
```

ou

```
try:
    resultat = numerateur / denominateur
except  NameError: # une en particulier
    print("La variable numerateur ou denominateur n'a pas été définie.")
```

Attention aux tabulations.

Modifiez le programme de la tâche précédente afin d'intercepter l'erreur *KeyboardInterrupt* et d'afficher 'Fin du programme' lorsqu'on appuie sur CTRL C

4. Retour sur la led

Écrivez un programme qui fait clignoter la LED (indéfiniment) avec une période de 1 seconde. Il faudra :

- Surveillez le code avec ***try***,
- Interceptez les exceptions et y *nettoyer* les GPIOs avec GPIO.cleanup();

Pour information, le code pour Arduino correspondant est :

Arduino

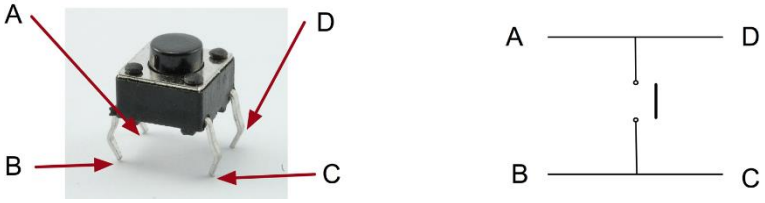
```
int LED = 21;
void setup() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
void loop() {
  // put your main code here, to run
repeatedly:
}
```

VI. Utilisations des boutons poussoirs

1. Rappels sur BP

a. Brochage

On peut s'attendre trouver un BP avec deux broches, en général il y en a quatre :



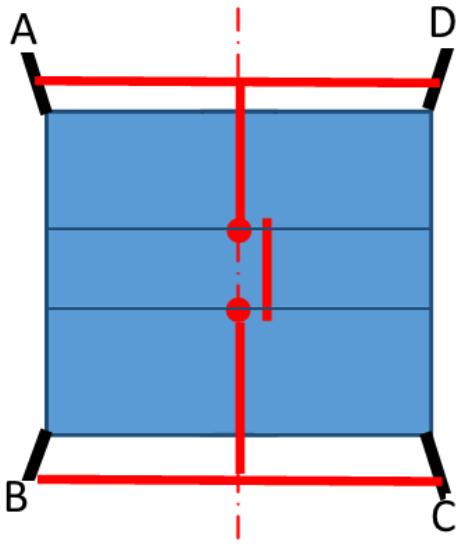


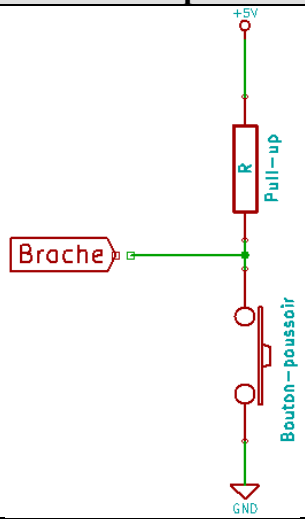
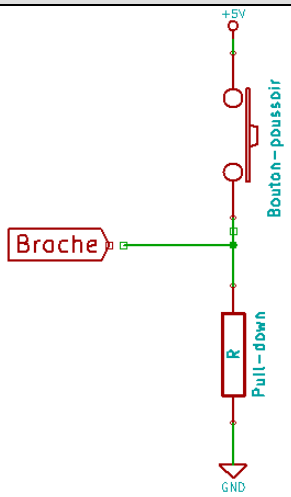
Figure 3 : les broches d'un BP

Les broches A et D sont connectées ensemble, de même pour B et C.

Comment les reconnaitre ? Il suffit de le retourner.

b. Montages pull-up & down

Il existe deux types de montage :

Pull-up	Pull-down
	
Quand le bouton est appuyé, la broche est à la masse	Quand le bouton est appuyé, la broche est au Vcc

Explications

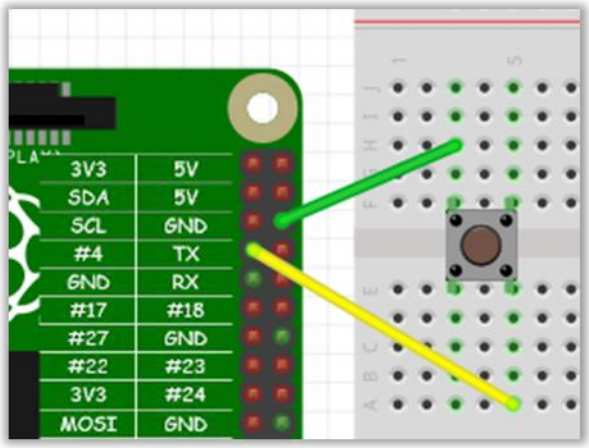
Une résistance de rappel permet de fixer une entrée numérique à un état **Haut (3.3V)** ou **Bas (0 V)**. Elle permet aussi de réduire le bruit, d'éliminer les broches flottantes et surtout, **d'établir deux états électriques clairs et distincts.**

<u>En pull-up</u> , si le BP est pas appuyé, la broche est à l'état Bas sinon c'est l'état Haut (3.3V sur la Rpi)
<u>En pull-down</u>, c'est l'inverse

Comme pour les leds, un état haut ne signifie pas qu'on a appuyé sur le bouton : cela dépend du montage !!!

La résistance est très souvent de 10KOhms.
Il est possible de diminuer cette résistance à 4.7 KOhms mais cela consommera aussi plus de courant.
Il n'est pas conseillé d'utiliser une résistance de pull-up de plus de 10KOhms.... à partir de 100 KOhms, la résistance de pull-up/down interfère avec la circuiterie interne du micro-contrôleur et le résultat (la détection) deviendrait incertain**.**
Pour des raisons liées aux technologies des circuits logiques, les électroniciens préfèrent utiliser une résistance pull up.

2. Mettre en œuvre un bouton-poussoir



Puisque le BP est connecté à la **masse** sans résistance, le montage choisi est donc le **pull-up.**
La broche d'entrée est toujours à 3.3 Volts sauf le bouton est appuyé

Remarquez l'absence de résistance.

Il n'est pas nécessaire de mettre une résistance, le RPI dispose de résistances internes en pull-up ou down aux choix de l'utilisateur.

a. Code Arduino correspondant

Encore une fois, la logique d'utilisation directe des entrées/sorties ressemble à celle de l'Arduino :

```
#define BP 4 // BP sur la broche 4
void setup() {
  /* declare BP en entrée et résistance interne en pull-up */
  pinMode(BP, INPUT_PULLUP);
  Serial.begin(9600);
  Serial.println("Démarrage");
}
void loop() {
  if (digitalRead(BP) == LOW) Serial.println("BP On"); // BPOn si état bas car pull-up
  delay(250);
}
```

Maintenant que vous savez programmer une led, c'est maintenant le tour des boutons :

Code Raspberry Pi

Ecrivez le code python équivalent à celui de l'Arduino

Aide :

- Permettre la configuration de la broche *pin* en sortie et en utilisant la résistance interne pull-down
  `GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)`
- `GPIO.input(pin)` renvoie l'état de la broche d'entrée *pin* soit 0 (=la masse) ou 1 (=3.3V)
- `void loop` se traduit par une boucle infinie
- Ne pas oubliez le `try`

Note 1 : si vous avez «
BP On
BP On
BP On
BP On
BP On
BP On
BP On

Sans avoir appuyé sur le BP, c'est que vous l'avez positionné à l'envers. Alors simplement, faite lui faire un quart de tour (vers la droite ou la gauche).

VII. Exercices-problèmes

1. Qui est le plus rapide ?

Le jeu est simple : deux boutons, deux joueurs et une led. Celui qui appuie dès que la led s'allume a gagné.

2. Eléments du langage python utilisés

a. Boucle pour avec range()

```
for i in range(12):
    print('i=',i)
```

b. Module

Un module est une bibliothèque contenant des fonctions et variables écrite par une autre personne (ou un groupe). Les modules sont des fichiers qui regroupent des ensembles de fonctions

Par exemple l'utilisation du module math peut se faire deux façons différentes :

```
import math
math.sqrt(16)
sqrt(16)

from math import sqrt
math.sqrt(16)
sqrt(16)
```

c. La fonction aléatoire

```
import random
print("nombre aleatoire entre 1 et 10 :" , random.randint(1, 10))
```

d. Fonctions

```
def abs(x):
    if (x>0):
```

```
        return x
    else:
        return -x

y=int(input("Entrer y: "))
print(abs(y))
#affiche par ex 2 si y=-2
```

e. Boucle infinie

Il est souvent utilisé une boucle infinie avec un break qui permet de sortir de la boucle :

```
# -*- coding: utf-8 -*-

print('On y va')
while True:
    print('Les touches A, B ou C permettent de sortir de la boucle' )
    print('Les autres vous renvoient un message')
    print('Quelle est votre choix ? ')
    touche=input()
    if touche=='A' or touche == 'B' or touche =='C':
        print('bye bye')
        break;
    else:
        print('On continue\n')

print('Arrêt du programme')
```

⌨ Essayez de prédire le programme ci-dessus si le joueur a choisit B

Pour information ce programme pourrait se simplifier de cette façon :

```
…
touche=input('Quelle est votre choix ? ')
if touche in  ['A','B','C']:#ou encore in "ABCD"
…
```

3. Règle du jeu

Ce jeu se joue à deux : une led s'allume et le premier qui a appuyé sur son bouton a gagné.

- Phase I : au début la led clignote cinq fois puis reste allumé pendant 2 secondes. Ensuite, le programme attend aléatoirement entre 1 et 3 secondes
- Phase II : la led s'allume et le jeu commence. Le programme attend l'appui sur un des deux boutons.
- Phase III : la led clignotera une fois si c'est le joueur de gauche qui a été le plus rapide ou deux fois si c'est celui de droite.

Quand le jeu est terminé, la led s'éteint.

4. Code du jeu « qui est le plus rapide »

⌨ Effectuez le **montage** virtuel avec fritzing https://fritzing.org/download/. Le faire valider par le professeur

⌨ Implémentez la fonction ***Clignoter*** qui demande un paramètre le nombre de clignotement et fera clignoter la led puis la fonction ***Phase1***. Testez-les.

⌨ Testez le bon montage des boutons-poussoirs avec un mini-programme.

Développons le code :

(Implémentez ce jeu avec une Rpi, une led et deux BP. Utilisez au moins deux **fonctions** pour les deux premières phases

Remarque :
Il est souvent utile de connaitre le nom exact de l'évènement qui a été intercept. Dans le except, ajoutez :

```
except Exception as e:
    print("L'exception est ",str(e))
    print('Fin')
    GPIO.cleanup()
```

5. Correction cachée pour le prof ☺

6. Extensions

a. Extension avec deux leds RGB

Une led RGB par joueur apporteront une information plus précise :

- Phase I : les deux leds Bleues resteront allumés pendant 5 secondes. Ensuite, le programme attend aléatoirement entre 3 et 10 secondes.
- Phase II : Si le joueur a gagné, sa led clignotera au Vert (sinon c'est Rouge)

En veillant à bien sauvegarder votre travail, implémentez cette nouvelle version

b. Redémarrage

Une nouvelle phase permet de redémarrer le jeu.

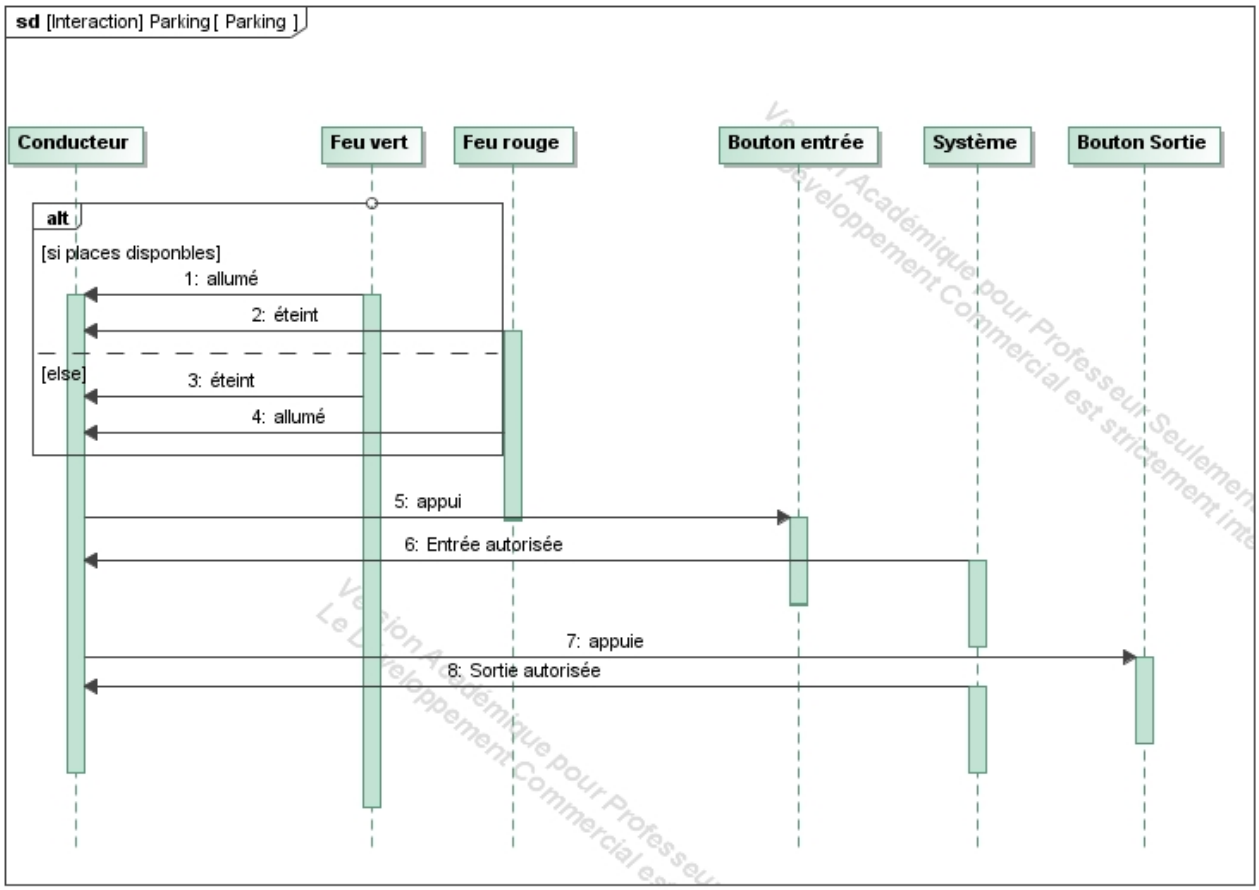
- Phase III : pour redémarrez le jeu, il faudra appuyer simultanément sur les deux boutons.

En veillant à bien sauvegarder votre travail, implémentez cette nouvelle version

c. Exercice : Places de parking

Le gérant d'un parking voudrait mettre en place un système permettant de prévenir les usagers lorsque le parking est complet : le feu vert indique des places libres et le feu rouge est allumé dans le cas contraire.

Un conducteur, désirant utiliser ce parking, arrive à l'entrée, les feux lui indiquent si c'est complet ou non. Ensuite pour entrer, il doit appuyer sur un bouton. De même pour sortir, il doit appuyer sur le bouton « sortie »



Configuration :

- Utiliser deux BP en **pull_up**
- Une led RGB

Aide pour la programmation :

- Pour faciliter les tests, le nombre maximum de voitures est fixé à 5.
- Définir une variable globale NombreVoitureParking représentant le nombre de voiture dans le parking