Avant de commencer...

Cet environnement de travail s'appelle un **notebook Jupyter** : à l'intérieur d'un navigateur Web, alternent

- des cellules de média (comme celle-ci) avec du texte, des images, des liens;
- des cellules de code (destinées à contenir du Python).

Pour exécuter une cellule de code, il faut :

- cliquer sur la cellule pour qu'elle soit encadrée en vert (selectionnée);
- le curseur de texte clignote à l'intérieur : on peut modifier le contenu de la cellule;
- pour exécuter la cellule, il y a le bouton « Exécuter » du menu du haut, ou mieux, le raccourci `MAJ + Entrée` .

Quand une cellule **n'a pas encore exécutée**, dans sa marge gauche on peut lire `IN [ ]` (ou `Entrée [ ]`).

Une fois exécutée, on pourra lire `IN [n]` , où `n` représente le nombre de fois qu'on a exécuté n'importe laquelle des cellules de code.

En théorie

Le notebook est écrit pour que les cellules soient exécutées les unes après les autres, dans l'ordre de la lecture : en effet, toutes les exécutions sont « gardées en mémoire ».

À FAIRE : Exécuter les deux cellules suivantes.

In [4]:
```
a = 3


b= 14
```

In [5]:
```
print(b)
```

```
14
```

À FAIRE : Modifier la première cellule : affecter une valeur autre que 3, puis exécuter de nouveau les deux cellules et constater le résultat.

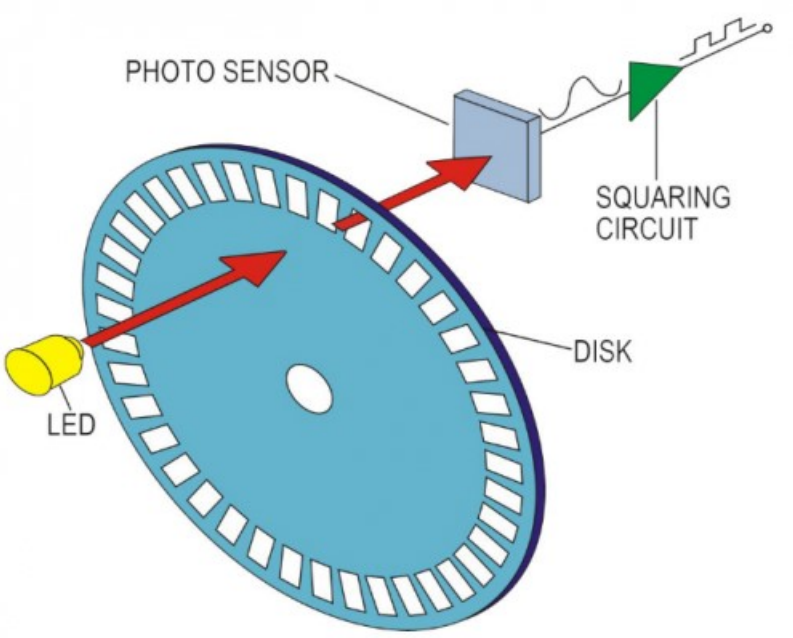
In [3]:
```
## Bibliotèques nécessaires pour la suite du TP

import matplotlib.pyplot as plt
import numpy as np
```

Mesures à partir de codeurs

Utilisation d'un codeur incrémental

Un codeur incrémental est un capteur permettant de mesurer un angle ou une vitesse de rotation. Bien souvent, les codeurs s'appuient sur des technologies optiques. La méthode utilisée pour la mesure peut, bien évidemment, être étendue à d'autres technologies.

- Ce capteur est lié mécaniquement à un arbre qui l'entraîne. Son axe fait tourner un disque qui lui est solidaire. Le disque comporte une succession de parties opaques et transparentes.
- Une lumière est émise par une ou des Diode(s) Electroluminescente(s) (DEL), la luminère traverse les fentes de ce disque créant sur la ou les photodiode(s) réceptrice(s) un signal analogique.
- Le signal est ensuite amplifié puis converti en signal carré, qui est alors transmis à un système de traitement



À FAIRE : Pour voir, en réalité, à quoi ce type de capteur ressemble, taper sur un moteur de recherche "codeur incrémental" et observer rapidement les images retournées.

Mesure de l'angle et de la vitesse angulaire avec une voie de mesure et détection des fronts montants

Un montage tel que celui présenté sur le premier schéma permet de relever le signal en sortie du montage de redressement. Ce relevé, réalisé à l'oscilloscope, permet d'obtenir un fichier texte dans lequel on retrouve l'évolution du signal mesuré $signal$ (en Volts) en fonction du temps t (en s).

Ce relevé a été obtenu pour un **codeur incrémental ayant 64 fentes** équiréparties sur le pourtour du disque. Il y a **équirépartition** dans le sens où la zone laissant passer la lumière a la même dimension que la zone ne la laissant pas passer.

Le signal est donc une succession de valeurs de tension :

- ***à l'état haut*** (ici environ 5 V) : la lumière arrive sur le récepteur ;
- ***à l'état bas*** (ici environ 0 V) : la lumière est bloquée.

Ce fichier est appelé `brut_1voie.txt` . Pour effectuer le tracé, on peut utiliser la fonction `lecture_fichier` définie ci-dessous (il suffit d'exécuter la cellule, tout est déjà fait !).

In [ ]:
```
def lecture_fichier(nomfichier,affichage_graphique):
    ##########################################################################
    ##########################  LECTURE DES DONNEES  #########################
    ##########################  issues d'un fichier  #########################
    ##########################################################################
    source = open(nomfichier, "r")

    donnees_totales = []
    ## Lecture donnees ligne par ligne
    for ligne in source:
        # Extraction des données d'une ligne
        donnees = ligne.rstrip('\n\r').split("\t") ## split " " pour un espa
        ndonnees = len(donnees)
        donnees_totales.append(donnees)

    n = len(donnees_totales)
    ## On range les donnees dans deux listes distinctes
    temps = [float(donnees_totales[i][0]) for i in range(0,n)]

    signaux = []
    for j in range(1,len(donnees_totales[0])):
        signalj = [float(donnees_totales[i][j]) for i in range(0,n)]
        signaux.append(signalj)

    if affichage_graphique:
        figure = plt.figure(figsize = (12, 10))
        for j in range(1,len(donnees_totales[0])):
            plt.plot(temps,signaux[j-1],label='signal'+str(j))
        plt.xlabel('Temps (s)')
        plt.ylabel('Signal (V)')
        plt.legend()
        plt.grid(True)
        plt.show()

    return (temps,signaux)
```

In [ ]:
```
temps,signaux=lecture_fichier("brut_1voie.txt",True)
signal = signaux[0]
```

Question 1. Dessiner le disque utilisé avec ses 64 fentes (ne pas tout dessiner et ne pas respecter l'échelle !). Tracer ensuite l'évolution théorique du signal lorsque $\theta \in [0; 2\pi]$ (où θ représente la position angulaire du disque). On supposera que le signal est à l'état haut pour $\theta = 0$.

Question 2. a. Commenter la courbe affichée précédemment puis tracer approximativement **et en mesurant les valeurs nécessaires** :

- l'angle θ en fonction du temps ;
- la vitesse de rotation (que l'on notera ω) en fonction du temps.

Question 2. b. Est-il possible, avec ce capteur, de connaitre l'angle θ à l'instant initial ? Est-il possible de connaitre le sens de rotation du disque ?

Une étape préliminaire au traitement des données pour obtenir numériquement l'évolution de θ et ω en fonction du temps est de modifier le signal tracé précédemment (et stocké dans la variable `signal`).

Question 3. Modifier les instructions données ci-dessous afin d'obtenir la variable `signal_modif` qui correspond au signal `signal` et qui vaut `True` lorsque le signal est à l'état haut et `False` sinon.

In [ ]:
```
def modif(signal):
    n = len(signal)
    s_modif = np.zeros(n) # création d'un tableau rempli de 0.
    for i in range(0,n):
        if signal[i]>4.5:
            s_modif[i] = ####  ..................................... à comp
        else:
            s_modif[i] = ####  ..................................... à comp
    return s_modif

signal_modif = modif(signal)
figure = plt.figure(figsize = (12, 10))
plt.plot(temps,signal_modif) # Python peut "tracer des booléens", il affect
plt.xlabel('Temps (s)')
plt.ylabel('Signal modifié (0/1)')
plt.grid(True)
plt.show()
```

Une notion classiquement utilisée pour ce type de problématique est la détection des fronts montants (et descendants). **Un front montant est l'évènement qui correspond au passage du signal de l'état bas vers l'état haut**.

Question 4. Modifier les instructions données ci-dessous afin d'obtenir le tableau (numpy) `Tm` qui contiendra l'ensemble des temps associés aux fronts montants du signal modifié. Vérifier le résultat obtenu.

In [ ]:
```
def Tmontant(s_modif,temps):
    Tm = []
    n = len(s_modif)
    for i in range(0,n-1):
        if ####  ..................................... à compléter :
            Tm.append(temps[i])
    return np.array(Tm) # pour obtenir un tableau numpy

Tm = Tmontant(signal_modif,temps)
print('Tm =',Tm)
```

Question 5. Quel est l'angle $\Delta\theta$ parcouru entre deux fronts montants si le disque contient n_f fentes ?

A RETENIR : Le plus petit angle que l'on peut mesurer avec un capteur s'appelle la **résolution** du capteur. La résolution d'un codeur incrémental de n_f fentes et pour lequel on ne détecte que les fronts montants est donc le résultat de la question précédente.

Question 6. Compléter le code suivant afin de tracer l'évolution de θ en fonction du temps (en utilisant uniquement les fronts montants).

In [ ]:
```
nf = 64 # nombre de fentes
Dtheta = 2*np.pi/nf # angle entre deux fronts montants

def calcul_theta_fm(Tm,Dtheta): # T correspond aux temps des fronts montant
    Theta = []
    nT = len(Tm)
    for i in range(0,nT):
        Theta.append(####  ..................................... à complété
    return np.array(Theta) # pour obtenir un tableau numpy

theta = calcul_theta_fm(Tm,Dtheta)

print('theta =',theta)

figure = plt.figure(figsize = (12, 10))
plt.plot(Tm,theta)
plt.xlabel('Temps (s)')
plt.ylabel('Angle $\\theta$ (radians)')
plt.grid(True)
plt.show()
```

Question 7. Modifier le code suivant afin de tracer la vitesse de rotation ω en fonction du temps.

In [ ]:
```
def deriv(Tm,theta): # T correspond aux temps des fronts montants
    nT = len(Tm)
    omega = []
    for i in range(1,nT): # on perd une valeur
        omega.append(####  ..................................... à complété
    return np.array(omega) # pour obtenir un tableau numpy

omega = deriv(Tm,theta)

print('omega =',omega)

Tmbis = Tm[:-1]

figure = plt.figure(figsize = (12, 10))
plt.plot(Tmbis,omega)
plt.xlabel('Temps (s)')
plt.ylabel('Vitesse angulaire $\\omega$ (radians/sec)')
plt.grid(True)
plt.show()
```

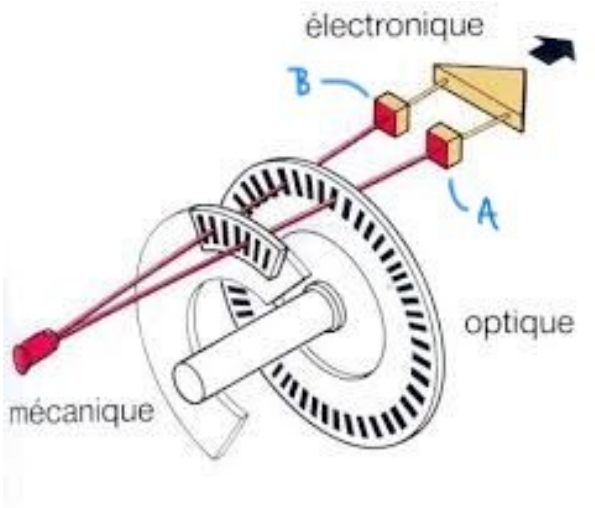
Question 8 - BONUS. Ecrire les instructions permettant de tracer l'angle θ et la vitesse de rotation ω en prenant en compte **les fronts montants et les fronts descendants**.

In [ ]:
```
##
## Code (long) à écrire
##
```

Question 9. Quelle sera la résolution (en radians) d'un codeur incrémental de n_f fentes, muni d'une voie de mesure, et pour lequel ou détecte les fronts montants et descendants ?

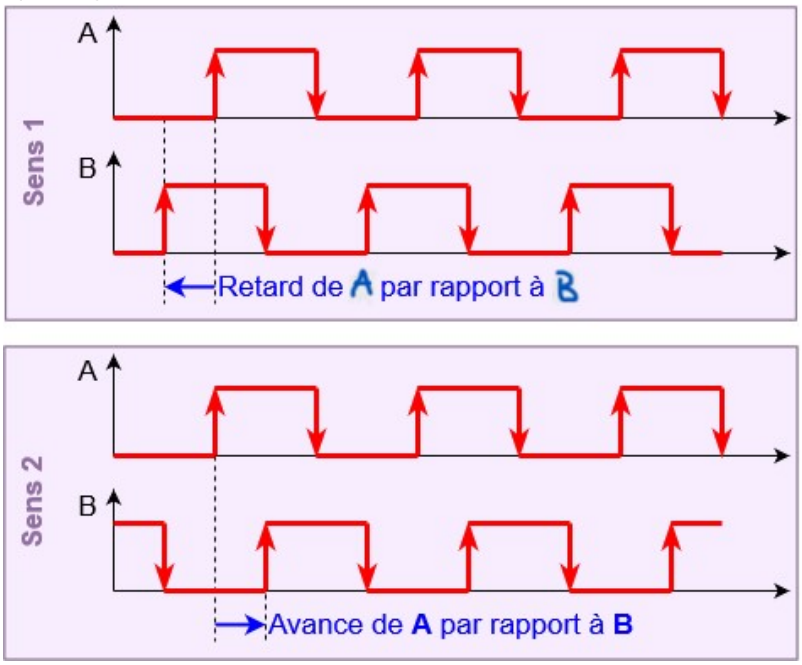
Mesure de l'angle avec deux voies de mesure

L'avantage d'utiliser deux voies de mesures est essentiellement de pouvoir connaitre le sens de rotation du disque. D'une manière simplifiée, un codeur avec deux voies de mesures se schématise de la manière suivante :



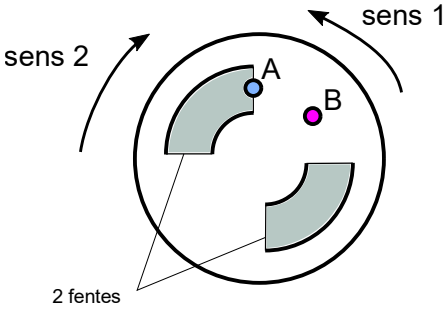
Les signaux A et B seront, bien souvent, décalés d'un quart de période. Dans un sens de

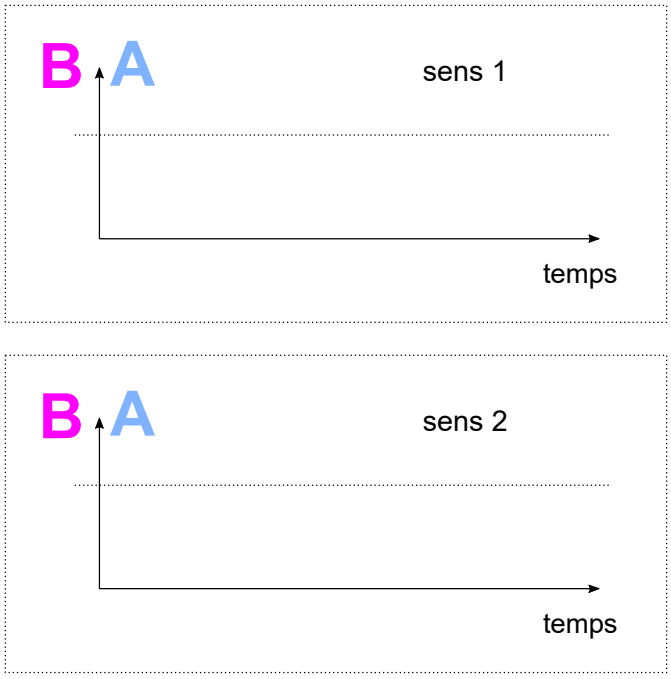
rotation (sens 1), le signal B sera en avance sur le signal A. Dans l'autre sens, le signal B sera en retard (voir figure ci-dessous qui représente A et B en fonction du temps).



Question 10. Quel sera le plus petit angle (en radians) mesurable avec un codeur incrémental de n_f fentes, **muni de 2 voies de mesure**, et pour lequel on détecte les fronts montants et descendants ?

Question 11. Retrouver les résultats de la figure ci-dessous en traçant l'évolution de A et B dans les sens 1 et 2 pour le montage ci-dessous. On supposera une vitesse de rotation constante.





On donne, dans le fichier `brut_2voies.txt` , les signaux brutes A et B obtenus avec un codeur à deux voies de mesures. Ce fichier contient trois colonnes qui correspondent, dans

l'ordre, au temps, au signal A et au signal B.

Question 12. Tracer les signaux A et B en fonction du temps (en utilisant la fonction `lecture_fichier`).

In [ ]:
```
temps,signaux = lecture_fichier(####  ..................................
signalA,signalB = signaux
```

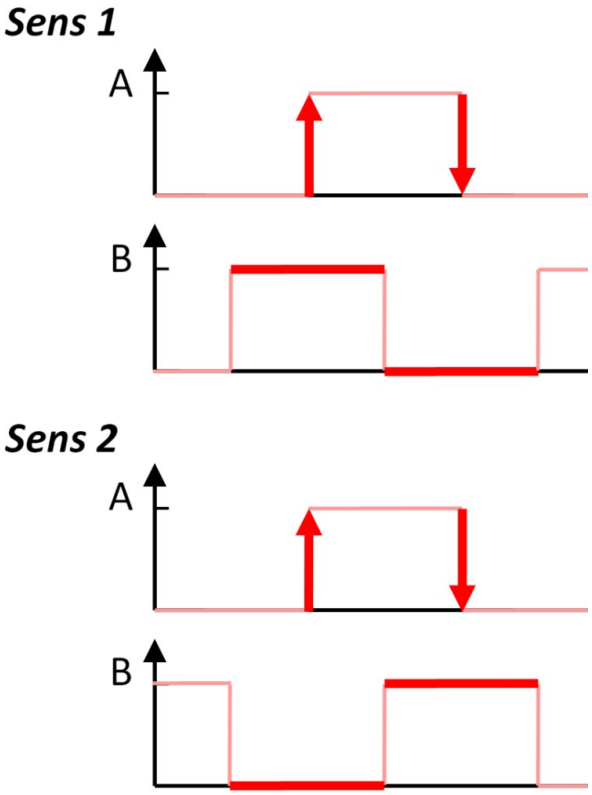
On peut également utiliser la fonction `modif` afin de convertir les signaux A et B.

In [ ]:
```
signalA_modif = modif(signalA)
signalB_modif = modif(signalB)
figure = plt.figure(figsize = (12, 10))
plt.plot(temps,signalA_modif,label='signal A') # Python peut "tracer des bo
plt.plot(temps,signalB_modif,label='signal B') # Python peut "tracer des bo
plt.xlabel('Temps (s)')
plt.ylabel('Signaux modifiés (0/1)')
plt.legend()
plt.grid(True)
plt.show()
```

Une manière de détecter le sens de rotation est, par exemple, d'analyser l'état du signal B lors d'un front montant ou descendant du signal B.

On remarquera alors que :

- lors d'un front montant de A : si B est à l'état haut alors le disque tourne dans le sens 1 et dans le sens 2 sinon ;
- lors d'un front descendant de A : si B est à l'état haut alors le disque tourne dans le sens 2 et dans le sens 1 sinon.



Question 13. On se limite d'abord au cas des fronts montants pour un codeur de 64 fentes. Compléter le code ci-dessous afin de renvoyer un tableau, de taille (nombre de fronts montants) x 2, contenant :

- les temps correspondants aux fronts montants de A (dans la première colonne) ;
- +1 ou -1 si on tourne respectivement dans le sens 1 ou le sens 2 (dans la deuxième colonne).

In [ ]:
```
def Tmontant_sens(sA_modif,sB_modif,temps):
    Tm = []
    Sens = []
    n = len(sA_modif) # sA_modif et sB_modif ont la même taille
    for i in range(0,n-1):
        if sA_modif[i+1] == True and sA_modif[i] == False: ## Détection d'u
            if ####  ..................................... à compléter
                Tm.append(temps[i])
                Sens.append(####  ..................................... à c
            else:
                Tm.append(temps[i])
                Sens.append(####  ..................................... à c
    return np.array([Tm,Sens]) # pour obtenir un tableau numpy

TmSens = Tmontant_sens(signalA_modif,signalB_modif,temps)
print('TmSens =',TmSens)
```

Question 14. Compléter le code suivant afin de tracer l'évolution de θ en fonction du temps (en utilisant uniquement les fronts montants et en supposant que le disque possède 64 fentes).

In [ ]:
```
nf = 64 # nombre de fentes
Dtheta = 2*np.pi/nf # angle entre deux fronts montants

def calcul_theta_fm(TmSens,Dtheta): # T correspond aux temps des fronts mon
    Theta = []
    nT = len(TmSens[0])
    t = 0 # t est l'angle theta
    for i in range(0,nT):
        Theta.append(t)
        t = t + ####  ..................................... à compléter
    return np.array(Theta) # pour obtenir un tableau numpy

theta = calcul_theta_fm(TmSens,Dtheta)

print('theta =',theta)

figure = plt.figure(figsize = (12, 10))
plt.plot(TmSens[0],theta) # Python peut "tracer des booléens", il affecte 1
plt.xlabel('Temps (s)')
plt.ylabel('Angle $\\theta$ (radians)')
plt.grid(True)
plt.show()
```

Question 15 - BONUS. En déduire, en écrivant les instructions adaptées, l'évolution de la vitesse de rotation ω en fonction du temps.

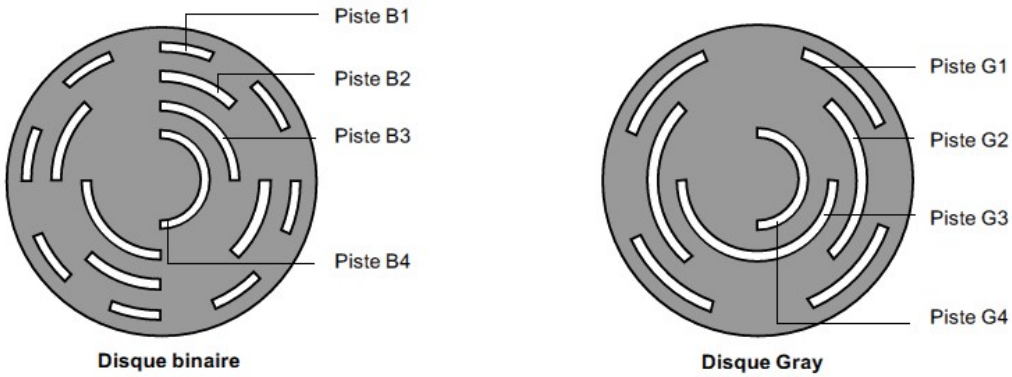
In [ ]:
```
##
## Code (long) à écrire
##
```

Question 16 - BONUS. Ecrire les instructions permettant de tracer l'angle θ et la vitesse de rotation ω en prenant en compte **les fronts montants et les fronts descendants**.

In [ ]:
```
##
## Code (long) à écrire
##
```

Mesure de l'angle avec un codeur absolu

L'avantage d'utiliser un codeur absolu est de pouvoir connaitre, sans initialisation, la position angulaire du disque. Cela nécessite d'utiliser un disque avec plusieurs fentes. Sur la figure ci-dessous, il y a 4 pistes codées selon deux méthodes différentes (codage binaire ou codage Gray).



Question 17. Quelle est la résolution d'un codeur absolu avec N pistes ?

A RETENIR : Le résultat de la question 17 est à connaitre !

On note dans la suite `Lbn = [1,0,0,0,0,1 ...]` une liste qui représente le code mesuré en **binaire**. Ce code est de taille N où N est le nombre de pistes utilisées. Un 1 dans la liste signifie que la lumière peut passer par la fente et 0 signifie que la lumière est bloquée.

Le dernier terme de la liste `Lbn` est le terme associé à la piste `Bn` (voir schéma ci-dessus).

Question 18. Définir la fonction `bin2dec` dont la trame est fournie ci-dessous qui permet de convertir la liste `Lbn = [1,0,0,0,0,1 ...]` en un entier compris entre 0 et $2^N - 1$ (le dernier terme de `Lbn` correspond au bit de poids le plus fort).

In [ ]:
```
def bin2dec(Lbn):
    n = len(Lbn)
    val = 0
    for i in range(0,n):
        val = val + ####  ..................................... à compléter
    return val

Lbn = [1,1,0] # pour tester la fonction
val = bin2dec(Lbn)
print(val)
```

On donne, dans le fichier `brut_absolu.txt` , le relevé des signaux issus d'un codeur absolu à **7 pistes**. Ce fichier contient huit colonnes qui correspondent, dans l'ordre :

- au temps ;
- au signal issu de la piste qui est associée au bit de poids le plus faible ;
- ...
- au signal issu de la piste qui est associée au bit de poids le plus fort.

On simplifiera en considérant que pour `Lbn = [1,1,1,1,1,1,1]` , on a $\theta = 2.\pi$ et pour `Lbn = [0,0,0,0,0,0,0]` , on a $\theta = 0$.

Question 19. Ecrire l'ensemble des instructions nécessaires afin d'afficher l'évolution de θ en fonction du temps.

In [ ]:
```
##
## Code (long) à écrire
##
```

Utilisation du code gray (ou code binaire réfléchi)

Vous pouvez vous rendre ici pour avoir des images : [https://si.blaisepascal.fr/1t-code-binaire-reflechi/ (https://si.blaisepascal.fr/1t-code-binaire-reflechi/)](https://si.blaisepascal.fr/1t-code-binaire-reflechi/)

Les codeurs absolus utilisent en souvent un codage appelé code binaire réfléchi ou code

Gray. Il permet d'éviter les erreurs de lecture lors d'un changement d'état puisque, dans ce nouveau codage, un seul bit change de valeur lors d'une incrémentation ou décrémentation du nombre associé en décimal.

Règle d'incrémentation

« Pour passer d'une ligne à la suivante, on inverse le bit le plus à droite possible conduisant à un nombre nouveau. »

Addition décalée

Pour passer du binaire naturel au code binaire réfléchi, on peut également utiliser la méthode de l'addition décalée sans retenue :

- On pose l'addition du nombre en binaire naturel avec le même nombre décalé d'un bit vers la gauche,
- On additionne sans les retenues,
- On enlève le bit de poids faible du résultat.

Par exemple, on veut représenter 15_{10} et 10_{10} en code de Gray

15_{10} s'écrit 1111_2 en binaire naturel et donc en code Gray : $1111 + 11110 = 10001$

10_{10} s'écrit 1010_2 en binaire naturel et donc en code Gray : $1010 + 10100 = 11110$

Donc 15_{10} en binaire réfléchi s'écrit 1000 et 10_{10} s'écrit 1111.

Cette méthode revient en fait à faire une opération logique : un « ou exclusif » entre le nombre binaire naturel et le même nombre binaire pour lequel ont a décalé les bits de 1 place vers la gauche.

On maintenant `Lg = [1,0,0,0,0,1 ...]` une liste qui représente le code mesuré en **Gray**. Ce code est de taille N où N est le nombre de pistes utilisées. Un 1 dans la liste signifie que la lumière peut passer par la fente et 0 signifie que la lumière est bloquée.

Question 20. Définir la fonction `dec2gray` qui permet de convertir un entier compris entre 0 et $2^N - 1$ en une liste `Lg = [1,0,0,0,0,1 ...]` (le dernier terme de `Lg` correspond au bit de poids le plus fort).

Question 21. Afficher le résultat de la fonction précédente pour les entiers allant de 0 à 15. Dessiner alors le disque associé et vérifier que lors de sa rotation un seul bit change d'état pour chaque entier.

Question 22. Définir la fonction `gray2dec` qui permet de convertir la liste `Lg = [1,0,0,0,0,1 ...]` en un entier compris entre 0 et $2^N - 1$ (le dernier terme de `Lg` correspond au bit de poids le plus fort).