td	td 4.1	TSI1
	Codage binaire des entiers	Période 5
		1h

Le codeur incrémental étudié permet de déterminer la position angulaire.
Les deux pistes A et B permettent de savoir dans quel sens tourne la roue.
Une marque permet de définir la position 0, lors de l'initialisation du système.

Lorsque le disque tourne dans le sens positif, on incrémente de 1 la valeur en sortie du capteur. Pour une rotation dans le sens négatif, le compteur est décrémenté de -1.
La commande du système, conduit à ce que le compteur ne soit jamais négatif.

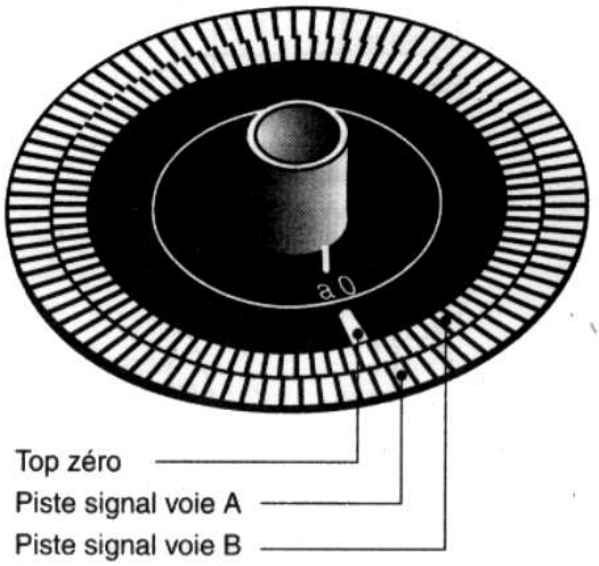

Figure 1 : Disque à 256 positions par tour

Conversion manuelle d'un nombre binaire
La position indiquée en sortie de capteur est
pos1 = 0b1000 0000.

1) Donner la valeur correspondante en base 10. Ecrire ce nombre en python.

2) Donner la valeur correspondante en base 16. Ecrire ce nombre en python.

3) Sachant que le codeur permet de coder 256 positions différentes, de combien de tour(s) est décalée la roue qui atteint la position **pos1** depuis sa position initiale à **0**.

Conversion manuelle d'un nombre décimal en nombre binaire
La position mesurée est incrémentée ou décrémentée dans une variable **pos2**.
Cette variable est stockée dans un registre (mémoire binaire) codé sur 16 bits (8bits sont utilisés pour la position dans le tour et 8 bits pour le nombre de tours).

4) Combien de tours de roue pourra-t-on coder avec ce registre ?

5) Si le codeur fait 300 tr, que se passerait-il ? Quelle conséquence pour le système.

On souhaite que le codeur fasse 10 tours à partir de la position initiale 0.

6) Quel code binaire sera stocké dans le registre ? Traduire ce nombre en hexadécimal. Quelle instruction en python permet d'obtenir cette conversion.

On souhaite que le codeur fasse 10 tours à partir de la position 10.

7) Quel code binaire sera stocké dans le registre ? Traduire ce nombre en hexadécimal.

Algorithme de conversion en base 10.

8) Définir une fonction **dec()** permettant de convertir le code binaire **pos** en code décimal **N10**.
Le code binaire **pos** est fourni sous forme d'une chaine de caractères (exemple **'0b1100'**)

Exemple d'utilisation :

```
>>> pos2 = bin(128)
>>> pos2
'0b10000000'
>>> dec( pos2 )
128
```

9) Comment plus simplement obtenir cette conversion de **pos2** avec python ?

[In 1]:

[Out1]:

Indication du nombre de tour

10) Ecrire une fonction **tour()** qui permet de :

- convertir le code binaire **pos** (chaine de caractères, exemple **'0b1110'**) en nombre de tours
- affiche et renvoie le nombre de tours **tours** avec une précision de 1/100.

Ecrire l'instruction qui permet d'appliquer cette fonction à la position **pos2**.

11) Ecrire une fonction **increment()** qui admet en arguments 3 booléens **Aold**, **A** et **B** et qui renvoie en sortie la valeur **inc** de l'incrément détecté :

- sur front montant de **A** (passage de 0 à 1 ; **0** est équivalent à **False** et **1** est équivalent à **True** donc on pourra utiliser pour cette question indifféremment ces notations) :
  - si **B=1** alors **inc** doit valoir **1**,
  - si **B=0** alors **inc** doit valoir **-1**,
- si **A** n'évolue pas alors **inc** doit valoir **0**.

12) Proposer un jeu de test à écrire en console pour vérifier que la fonction **increment()** renvoie les valeurs attendues. Indiquer dans un commentaire la valeur renvoyée qui est attendue.

On peut améliorer la résolution du capteur en détectant les fronts descendants de A (avec des conditions inverses pour B).

13) Quelles seront alors des adaptations à prévoir dans ce qui précède, notamment si on souhaite conserver le même système de codage sur un registre commun de 16 bits ?

Enfin, si on traite aussi les fronts de la piste B, on peut encore multiplier par 2 le nombre de mesures obtenues à la question précédente (on divise encore par 2 la précision).

14) En déduire la valeur décimale du plus grand code de la position dans le tour et la valeur du plus grand nombre de tours si on utilise un registre de 16bits communs à ces 2 informations.