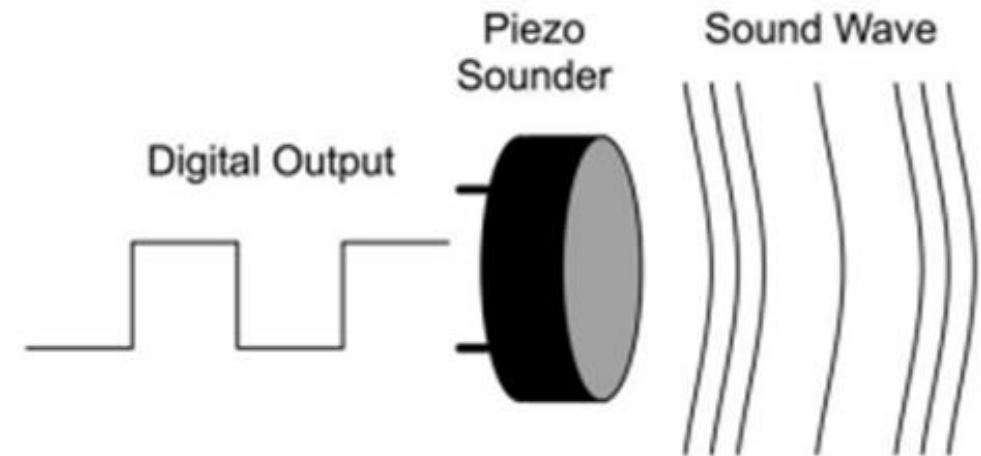


A3Exx Le Buzzer

Un matériau piezo électrique est une substance qui produit un courant électrique lorsqu'il est déformé. Et inversement, lorsqu'une tension électrique est placée sur la substance, une déformation a lieu.

Cet effet est causé par des polarisations de molécules. En effet toute molécule est chargée, donc un bout est chargé plus négativement que l'autre. On appelle ceci un dipôle. On peut imaginer alors une orientation des atomes définie par des vecteurs. Dans un monocristal, tous ces vecteurs sont dans le même sens et direction. Au contraire, dans un polycristal, ces vecteurs vont dans tous les sens et directions.

Avec un signal carré en entrée, le une déformation suivie d'un retour à l'état normale va engendrer une oscillation :



A3Exx Le son

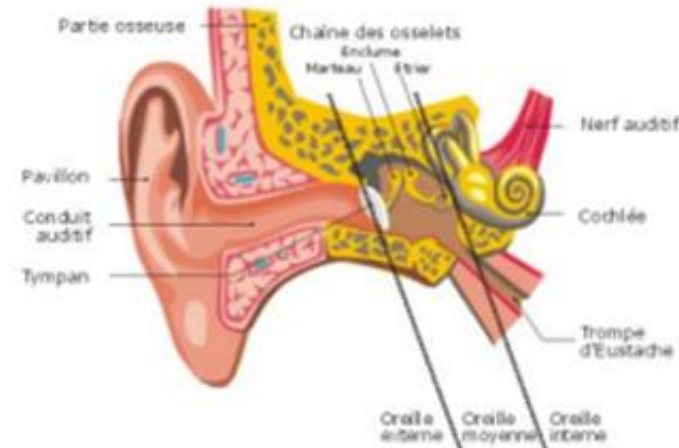
D'un point de vue physique, un son est une énergie qui se propage sous forme de vibrations dans un milieu compressible (dans l'eau, dans l'air, dans les matériaux solides, mais pas dans le vide!).

Lorsqu'on jette une pierre dans l'eau, on peut facilement observer le phénomène de propagation des ondes à la surface:



Lors de la diffusion d'un son dans un concert, c'est l'air qui permet sa transmission jusqu'à nos oreilles. De même que l'exemple de l'eau illustré ci-dessus, les molécules d'air transmettent l'énergie et sont donc un support pour le son.

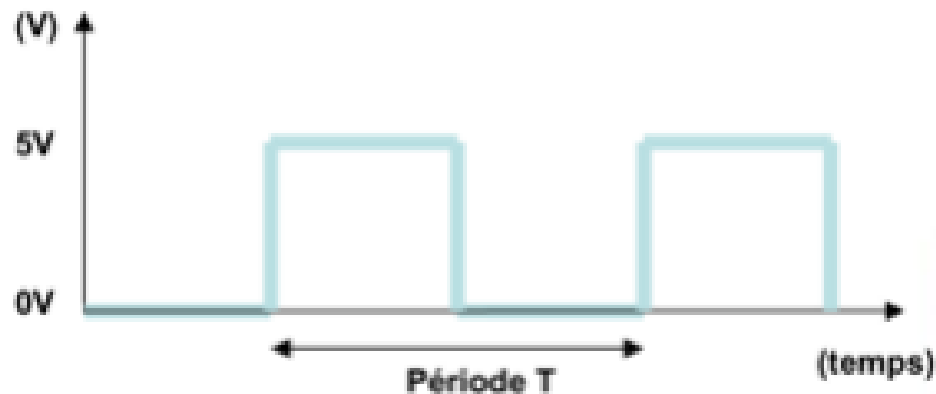
Pour être perçue, il doit y avoir un récepteur sensible. Chez l'homme, l'oreille possède une membrane (le tympan) capable de transmettre les informations de vibration en signaux nerveux jusqu'au cerveau, grâce au nerf auditif. De même, le microphone possède également une membrane permettant de transformer les déplacements de l'air en signaux électriques.



A3Exx Jouer une note

Le buzzer se câble sur une sortie numérique en PWM (ou MLI en français) et le microcontrôleur lui envoie alors un signal périodique dont on fait varier la fréquence en fonction de la note que l'on désire jouer.

$$f = 1/T ; T = 1/f ; f : \text{fréquence}; T : \text{période}$$

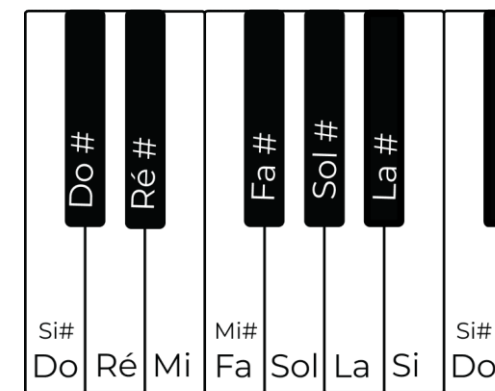


Exemple : le LA est un signal d'une fréquence f de 440 Hertz soit un signal qui varie 440 fois par seconde. La fréquence du DO est 262 Hz(octave 3) etc ..

A3Exx Les notes et leurs fréquences

Fréquences des hauteurs (en Hertz)







Note\octave	0	1	2	3	4	5	6	7
Do	32,70	65,41	130,81	261,63	523,25	1046,50	2093,00	4186,01
Do#	34,65	69,30	138,59	277,18	554,37	1108,73	2217,46	4434,92
Ré	36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	4698,64
Ré#	38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	4978,03
Mi	41,20	82,41	164,81	329,63	659,26	1318,51	2637,02	5274,04
Fa	43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	5587,65
Fa#	46,25	92,50	185,00	369,99	739,99	1479,98	2959,96	5919,91
Sol	49,00	98,00	196,00	392,00	783,99	1567,98	3135,96	6271,93
Sol#	51,91	103,83	207,65	415,30	830,61	1661,22	3322,44	6644,88
La	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	7040,00
La#	58,27	116,54	233,08	466,16	932,33	1864,66	3729,31	7458,62
Si	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	7902,13



Remarques :

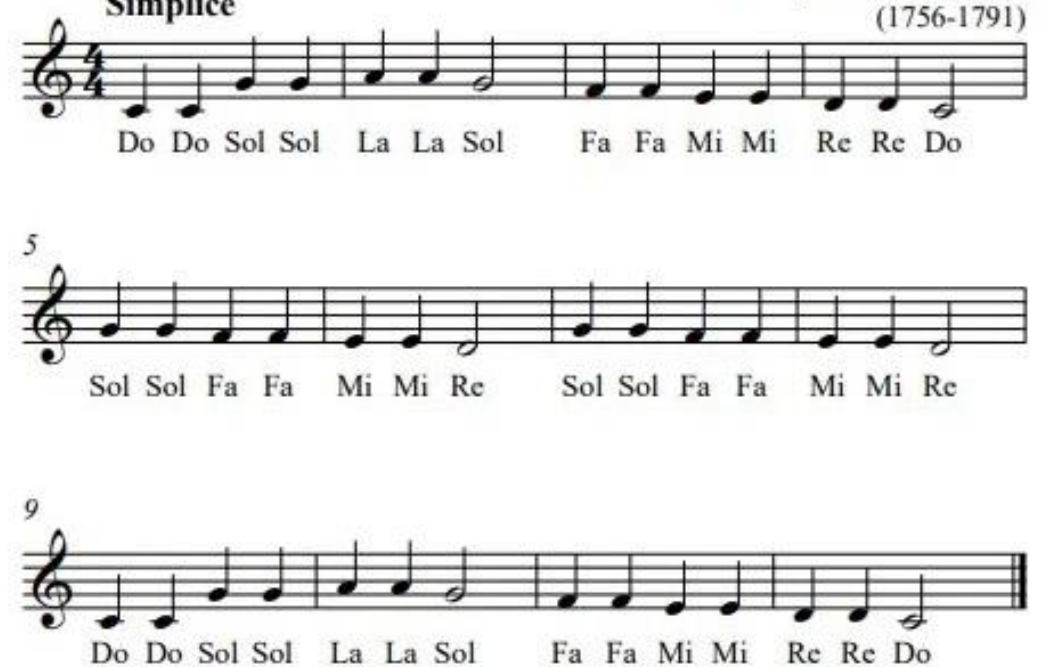
- Ne lisez pas hastag mais dièse à côté des notes !
- Le dièse augmente la fréquence tandis que le bémol la diminue. Ainsi un Do# peut s'écrire également Ré bémol (Réb)
- Lorsqu'on dit que la fréquence du La est de 440 Hz c'est vrai mais à l'octave 3.
- Pour passer d'un octave à un autre, pour une même note il suffit de multiplier ou diviser par deux (ou un multiple)

A3Exx Durée d'une note

<i>Ronde</i>	<i>blanches</i>	<i>noires</i>	<i>croches</i>	<i>doubles- croches</i>	<i>triples- croches</i>	<i>quadruples- croches</i>
la 	vaut 2	ou 4	ou 8	ou 16	ou 32	ou 64
la 	vaut 2	ou 4	ou 8	ou 16	ou 32	
la 	vaut 2	ou 4	ou 8	ou 16		
la 	vaut 2	ou 4	ou 8			
la 	vaut 2	ou 4				
la 	vaut 2					

Twinkle Twinkle Little Star

Simplice Wolfgang Amadeus Mozart (1756-1791)



Do Do Sol Sol La La Sol Fa Fa Mi Mi Re Re Do

5 Sol Sol Fa Fa Mi Mi Re Sol Sol Fa Fa Mi Mi Re

9 Do Do Sol Sol La La Sol Fa Fa Mi Mi Re Re Do

A3Exx Partitions simples (décodées)

VIVE LE VENT !

(avec les doigtés pour la main droite)



3 3 3 3 3 3
 MI MI MI, MI MI MI
 3 5 1 2 3
 MI SOL DO RÉ MI
 4 4 4 3 3 3
 FA FA FA, MI MI MI
 2 2 2 3 2 5
 RÉ RÉ RÉ MI RÉ SOL !
 3 3 3 3 3 3
 MI MI MI, MI MI MI
 3 5 1 2 3
 MI SOL DO RÉ MI
 4 4 4 3 3 3
 FA FA FA, MI MI MI
 3 5 5 4 2 1
 MI SOL SOL FA RÉ DO.

FRÈRE JACQUES



d'en bas pour
la dernière ligne

DO RÉ MI DO DO RÉ MI DO
Frè-re Jac-ques, Frè-re Jac-ques,
 MI FA SOL MI FA SOL
Dor-mez vous? Dor-mez vous?
 SOL LA SOL FA MI DO
Son-nez les ma-ti-nes
 SOL LA SOL FA MI DO
Son-nez les ma-ti-nes
 DO ↘ SOL ↗ DO DO ↘ SOL ↗ DO
Ding daing dong, Ding daing dong

Au Clair De La Lune

Moderato ♩ = 80

Traditional



A3Exx Partitions à décoder

Comment lire les notes de piano sur une portée?

Voici comment les notes sont positionnées sur la portée de la partition:



Le symbole situé tout à gauche des partitions, s'appelle la **clé de sol**. La clé indique en son centre l'étage de la note sol (comme son nom l'indique). Vérifiez vous-même : le centre de la clé de sol correspond bien à la ligne sur laquelle est située la note sol (observez la flèche rouge).

Ok?



Le reste des notes s'articule autour du sol (avec la série qu'on a apprise à la première leçon: do, ré, mi, fa, sol, la et si).
Remarquez que les notes se positionnent non seulement sur les lignes qui composent la portée mais également entre celles-ci!

Super Mario Bros Thème

- 1 -

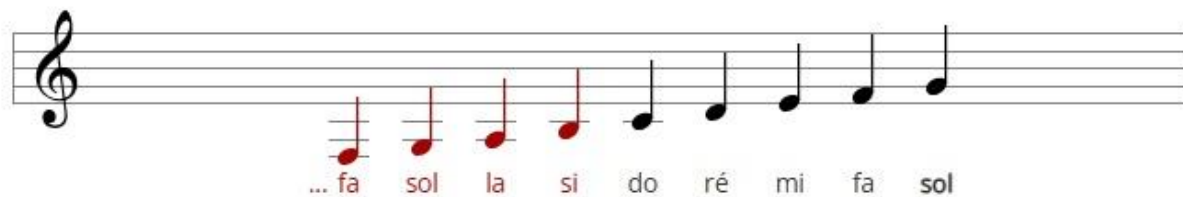


A3Exx Partitions à décoder

Voici comment les notes continuent vers les aigus:



Et voici comment les notes continuent vers les graves:



Tetris Type A

- 1 -

Piano

The piano accompaniment for 'Tetris Type A' consists of four systems, each with a treble and bass staff. The music is in common time (C) and features a steady, rhythmic accompaniment with various note values and rests. A large, faint watermark 'L' is visible across the piano part.

A3Exx Utilisation de la librairie Buzzer.

```
class Buzzer:
    def __init__(self, i_io):
        self._io_pwm = PWM(Pin(i_io, Pin.OUT))
        self.stop()
        self.set_volume(10)

    def start(self):
        self._io_pwm.duty_u16(self._volume)

    def stop(self):
        self._io_pwm.duty_u16(65535)

    def set_volume(self, i_volume):
        self._volume = int(((100-i_volume)*65535)/100)

    def set_freq(self, i_freq):
        self._io_pwm.freq(i_freq)
```

Fonctions utilisées:

buzzer = Buzzer(i_pin_num) -> on va utiliser la classe Buzzer avec un paramètre: le numéro de pin sur laquelle est branché le buzzer.

buzzer.set_freq(i_frequency): -> fréquence que l'on veut envoyer au buzzer.

buzzer.start(): -> on démarre le buzzer

buzzer.stop(): -> on arrête le buzzer

A3Exx Programme à écrire

Importer les modules nécessaires:

- Machine pour avoir accès aux IOs (Pin)
- Time pour pouvoir compter le temps
- Buzzer

Définitions des fréquences des notes à utiliser sous forme de liste de paires note/freq:

freq_notes = {"do":1046, "re":1175,...}

Déclaration du buzzer (Pin 2)

On éteint le buzzer

On crée une fonction `joue_note` qui prends comme paramètres une note et le temps qu'elle doit être jouée.

Le programme principal parcourt la liste (*for note in ...*) et appelle la fonction à chaque note...

A3Exx Correction

```
from machine import Pin
from buzzer import Buzzer
from time import sleep

# frequences des notes
freq_notes = {"do":1046,"do_":1109,
              "re":1175,"re_":1245,
              "mi":1318,
              "fa":1397,"fa_":1480,
              "so":1568,"so_":1661,
              "la":1760,"la_":1864,
              "si":1967}

# declaration du buzzer (Pin2, alimentation en 3.3V)
buz = Buzzer(2)
buz.stop()

# fonction qui va jouer une note
def joue_note(val_note,time):
    global buz
    buz.set_freq(freq_notes[val_note])
    buz.start()
    sleep(time)
    buz.stop()
```

```
#Jingle Bells
notes = [("mi",0.25),("mi",0.25),("mi",0.5),
         ("mi",0.25),("mi",0.25),("mi",0.5),
         ("mi",0.25),("so",0.25),("do",0.25),("re",0.25),
         ("mi",1),
         ("fa",0.25),("fa",0.25),("fa",0.5),
         ("mi",0.25),("mi",0.25),("mi",0.5),
         ("re",0.25),("re",0.25),("re",0.25),("mi",0.25),
         ("re",0.5),("so",0.5),
         ("mi",0.25),("mi",0.25),("mi",0.5),
         ("mi",0.25),("mi",0.25),("mi",0.5),
         ("mi",0.25),("so",0.25),("do",0.25),("re",0.25),
         ("mi",1),
         ("fa",0.25),("fa",0.25),("fa",0.5),
         ("mi",0.25),("mi",0.25),("mi",0.5),
         ("so",0.25),("fa",0.25),("mi",0.25),("re",0.25),
         ("do",1)]

while True:
    for note in notes:
        joue_note(note[0],note[1])
        sleep(0.01)
```