TP Faisons un peu de bruit

Objectif : prise en main du buzzer et gestion de l'affichage ainsi que d'un bouton

Rappel :
Sauvegarder le programme sous le nom ***main.py****.*
Copier le sur la mémoire flash de votre carte (PYBFLASH).
Attendez que la led rouge 1 s'éteigne.
Appuyer sur le bouton RST de la carte.

Le Buzzer (BEEP sur la carte) est un élément piézoélectrique qi va produire un son à une certaine fréquence en fonction du courant qu'il reçoit. On en trouve dans les montres ou dans les réveils par exemple.
On pourra paramétrer le son voulu soit par la fréquence à obtenir soit par la tension envoyé au buzzer (codé en bits)

Pour cette expérience, nous utiliserons DAC (pour le buzzer) et Switch (pour les boutons) du module Pyb, Pin et I2C du module machine et SSD1306_I2C du module SSD1306 pour l'affichage.

Module DAC :
DAC(port,bits=8) :
【port】: mettre à 1;
【bits】: Soit 8 soit 12.

`dac.noise(freq)` :Génère un son à une fréquence donnée.

`dac.triangle(freq)` :Génère un signal triangulaire à une fréquence donnée.

`dac.write(value)` :
Précise le courant reçu par le buzzer entre 0 et 3,3V.
【value】Plage: 0 à 2^bits-1 (par exemple pour 8bits, la plage est de 0-255)

`write_timed(data,freq,mode)`
【data】: Tableau de bytes pour la valeur de la tension
【freq】Fréquence de sortie;
【mode】peut être paramétré comme suit:
DAC.NORMAL: Une seule répétition ;
DAC.CIRCULAR: En mode boucle.

Si nous voulons faire un signal carré, nous pouvons très bien utiliser `write(0)` puis `write(255)`. En utilisant la fonction delay entre les commandes, on obtient un signal carré.
Mais on pourra aussi utiliser la fonction `write_timed(data,freq,mode)` comme dans le prochain code.

Le code ici est plus dense. Prenez le temps de l'examiner pour le comprendre et **chercher le rythme** pour trouver la mélodie…
Conseil : vérifiez bien que les décalages (indentations) soient les mêmes : une indentation = 1 TAB = 4 espaces.

```
# importation des modules
from pyb import DAC, Switch
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
# Initialisation des modules
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)
sw = Switch() # bouton usr -->sw
dac = DAC(1) # DAC1 --> X5
# 10 fréquences en Hertz différentes stockées dans une liste :
freq = [1,262, 294,330,294,262,330,294,262,1]

# Definition de la valeur des vagues en 8-bit
# 0 et 255 correspondent au sortie 0V and 3.3V respectivement.
# Nous devons les définir dans un tableau de byte (byte array).
buf = bytearray(2)
buf[0] = 0
buf[1] = 255
```

```
key_node = 0 # initialisation bouton
i = 0 # variable pour le choix de la fréquence parmi les 4
##############################################
# gestion du bouton
##############################################
def key():
    global key_node
    key_node = 1
sw.callback(key) # qd le bouton est pressé la fonction key est exécutée
##############################################
# Affichage de l'écran
##############################################
oled.fill(0) # écran vierge de départ
oled.text('Club Progr LAB', 0, 0)
oled.text('Buzzer', 0, 15)
oled.text('Pls pressez USER', 0, 40)
oled.show()
while True:
    if key_node == 1: # bouton pressé
        i = i+1
        if i == 10:  #nombre de notes ds la liste freq
            i = 0
        key_node = 0
 # Sortie du buzzer à la fréquence choisie
        dac.write_timed(buf, freq[i]*len(buf), mode=DAC.CIRCULAR)
 # Affichage de la fréquence sur l'écran
        oled.fill(0)
        oled.text('Club Prog LAB', 0, 0)
        oled.text('Buzzer', 0, 15)
        oled.text(str(freq[i]) + 'Hz', 0, 40)
        oled.show()
```

Modifier les fréquences pour faire une autre mélodie…
Vous pourrez aller chercher sur internet les fréquences des différentes notes qui vous intéressent.