Wprowadzenie

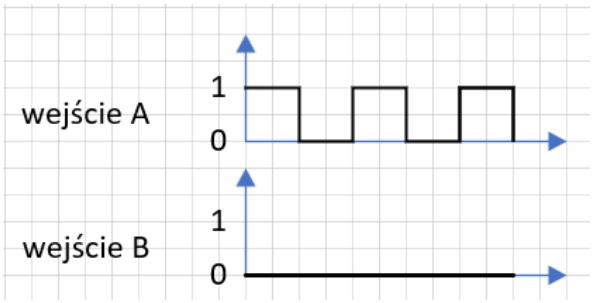
Płytka mikrokontrolera zestawu sprzętowego została wyposażona, między innymi, w piezoelektryczny przetwornik dźwiękowy. Pozwala on generować dźwiek za pośrednictwem wyjśc GPIO mikrokontrolera.

Wymagane wiadomości:

- wyprowadzenia GPIO mikrokontrolera
  - ustawianie funkcji
  - ustawianie stanu
  - zmiana stano na przeciwny (funkcja `toggle`)
- piezoelektryczny przetwornik dźwiękowy
  - zasada działania
  - sterowanie jedno i dwubiegunowe
- funkcja `sleep_us`

Zadanie 1

1. Napisać program który:
   - na jednym z wejść przetwornika ustawi znan niski
   - na drugim będzie generował przebieg prostokatny o częstotliwości 1kHz



   Do generowania używać, w tym i w następnych zadaniach funkcji `sleep_us`.

   Numery pinów mikrokontrolera użytych do sterowania przetworenikiem ustalić we własnym zakresie na podstawie oglądu płytki.

2. Wstawić **cały** kod programu

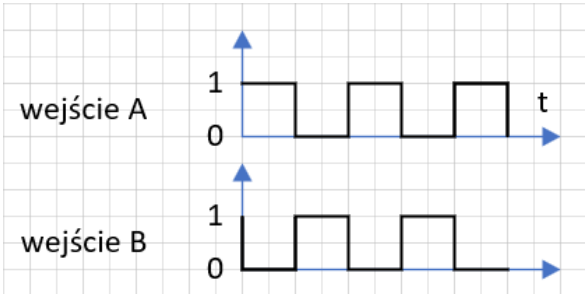
```
from machine import Pin
from time import sleep_us

buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
while(True):
    buzzer_b.value(0)
    sleep_us(500)
    buzzer_b.value(1)
    sleep_us(500)
```

Zadanie 2

1. Zmodyfikować poprzedni program realizował sterowanie dwubiegunowe, jak na poniższym rysunku



2. Wstawić kod bez importów.

```
buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

while(True):
    buzzer_a.value(0)
    buzzer_b.value(1)
    sleep_us(500)
    buzzer_a.value(1)
    buzzer_b.value(0)
    sleep_us(500)
```

Zadanie 3

1. Wykonać zadanie 1 ale z użyciem funkcji `toggle`
   Funkcja `toggle` może pojwaić się tylko raz w kodzie programu

2. Wstawić kod bez importów.

```
buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
while(True):
    buzzer_b.toggle()
    sleep_us(500)
```

Zadanie 4

1. Wykonać zadanie 2 ale z użyciem funkcji `toggle`
   Funkcja `toggle` może pojwaić się tylko dwa razy w kodzie programu

2. Wstawić kod bez importów.

```
buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
buzzer_b.value(1)

while(True):
    buzzer_a.toggle()
    buzzer_b.toggle()
    sleep_us(500)
```

Zadanie 5

1. Napisać program generujący dźwięk, którego czestotliowść będzie zależeć od naciśniętego przycisku, w sposób podany poniżej.
2. Użyć maks. 2x funckji `toggle` oraz 1x funkcji `ReadKeyboard`

Naciśnięt przycisk	Częstotliwośc dźwięku
Żaden	Cisza
0	250 Hz
1	500 Hz
2	1 kHz
3	2 kHz

3. Wstawić kod **z** importami.

```
from machine import Pin
from my_peri import ReadKeyboard
from time import sleep_us

buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
buzzer_b.value(1)

while(True):
    key = ReadKeyboard()

    if key!= None:
        buzzer_a.toggle()
        buzzer_b.toggle()

    if key==0:
        sleep_us(2000)
    elif key==1:
        sleep_us(1000)
    elif key==2:
        sleep_us(500)
    elif key==3:
        sleep_us(250)
```

Zadanie 6

1. Zdefiniować w programie funkcję **`beep(f_Hz,t_s)`**, która będzie generowała dźwięk o częstotliwości **`f_Hz`**, przez czas **`t_s`**.

2. Zmodyfikować pętlę głowną tak aby z użyciem funkcji `beep` generowała na przemian dźwięk o częstotliwości 500 Hz przez 0.5 sekundę i 2000 Hz przez 0.2 sekundy
3. Wstawić kod bez importów.

```
buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
buzzer_b.value(1)

def beep(f_Hz,t_s):
    T_us = 1_000_000 * (1/f_Hz)
    for i in range(2*t_s*f_Hz):
        buzzer_a.toggle()
        buzzer_b.toggle()
        sleep_us(int(T_us/2))

while(True):
    beep(500,0.5)
    beep(2000,0.2)
```

Zadanie 7

1. Napisać program, który, z użyciem pęli for iterującej po liście wygeneruje poniższą sekwencję częstotliwości. Dźwięki oraz odsepy pomiędzy xwiękami powinny trwać 0.1s

   1319,1175,740 ,831 ,1109,988 ,587 ,659 ,988 ,880 ,554 ,659 ,880

2. Wstawić kod pętli głownej.

```
for f in
1319,1175,740 ,831 ,1109,988 ,587 ,659 ,988 ,880 ,554 ,659 ,
880 :
    beep(f,0.1)
    sleep_us(100_000)
```

Zadanie 8

Zmodyfikować program tak aby współpracował z listą, kórej każdy element zawiera 3 elementową listę, której elementy, idąc od lewej reprezentują:

- wysokośc dźwięku w Hz
- czas trwania dźwięku wyrażony w milisekundach
- czas przerwy międz dźwiękami wyrażony w milisekundach

  (1319, 66, 71), (1175, 66, 71), (740, 133, 138), (831, 133, 138), (1109, 66, 71), (988, 66, 71), (587, 133, 138), (659, 133, 138), (988, 66, 71), (880, 66, 71), (554, 133, 138), (659, 133, 138), (880, 266, 271)

Wstawić kod programu bez importów.

```
buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
buzzer_b.value(1)

def beep(f_Hz,t_s):
    T_us = 1_000_000 * (1/f_Hz)
    for i in range(2*t_s*f_Hz):
        buzzer_a.toggle()
        buzzer_b.toggle()
        sleep_us(int(T_us/2))

l = (1319, 66, 71), (1175, 66, 71), (740, 133, 138), (831,
133, 138), (1109, 66, 71), (988, 66, 71), (587, 133, 138),
(659, 133, 138), (988, 66, 71), (880, 66, 71), (554, 133,
138), (659, 133, 138), (880, 266, 271)

for f,d,s in l:
    beep(f,d/1000)
    sleep_us(1000*s)
```

Zadanie 9

Przenieść funkcję beep wraz z kodem koniecznym do jej działania do modułu my_peri

Poprawność przeniesienia przetestować programem z poprzedniego zadania

Wstawić kod modułu my_peri

```
from peripherals import LedSet, LedClr, ButRead
from machine import Pin
from time import sleep_us

def LedPoint(pos):
    for i in range(4):
        LedClr(i)
    if pos != None:
        LedSet(pos)

def LedBar(led_nr):
    for i in range(4):
        LedClr(i)
    for i in range(led_nr):
        LedSet(i)

def ReadKeyboard():
    for i in 0,1,2,3:
        if ButRead(i):
            return i
    return None

buzzer_a = Pin(14, Pin.OUT)
buzzer_b = Pin(22, Pin.OUT)

buzzer_a.value(0)
buzzer_b.value(1)

def beep(f_Hz,t_s):
    T_us = 1_000_000 * (1/f_Hz)
    for i in range(2*t_s*f_Hz):
        buzzer_a.toggle()
        buzzer_b.toggle()
        sleep_us(int(T_us/2))
```

Wstawić kod programu

```
from my_peri import beep
from time import sleep_us

l =(1319, 66, 71), (1175, 66, 71), (740, 133, 138), (831,
133, 138), (1109, 66, 71), (988, 66, 71), (587, 133, 138),
(659, 133, 138), (988, 66, 71), (880, 66, 71), (554, 133,
138), (659, 133, 138), (880, 266, 271)

for f,d,s in l:
    beep(f,d/1000)
    sleep_us(1000*s)
```