Taller personalizado de Programación y Robótica en CMM BML

24_25 – CL13

[PyR] **List comprehension** en Python

[PyR] **Diccionarios** en Python básico

[R] **Sonido en Pico** con PWM 2da :

- NPN+SP + cap -> 1ros programas de test
- Frecuencias de notas: algo de teoría musical
- Canción Frere Jacques: con diccionario simpe -> con listas anidadas sobre diccionario

7 junio 2023

Voluntario : J.C. Santamaria

Clase 12 – Índice
(90 minutos)

1. **[R] Sonido básico 1 en Pico con PWM**

- Sonido solo buzzer: Activo / pasivo

- Sonido : NPN+Altavoz & 1ros programas test -5’

2. **[PyR] List comprehension y Diccionarios** en Python – 15’ + 15’

3. **[R] Sonido básico 2**

Frecuencias de notas del piano -10’ + 15’

Canción Frère Jacques modo simple y con listas anidadas– 10’ + 20’

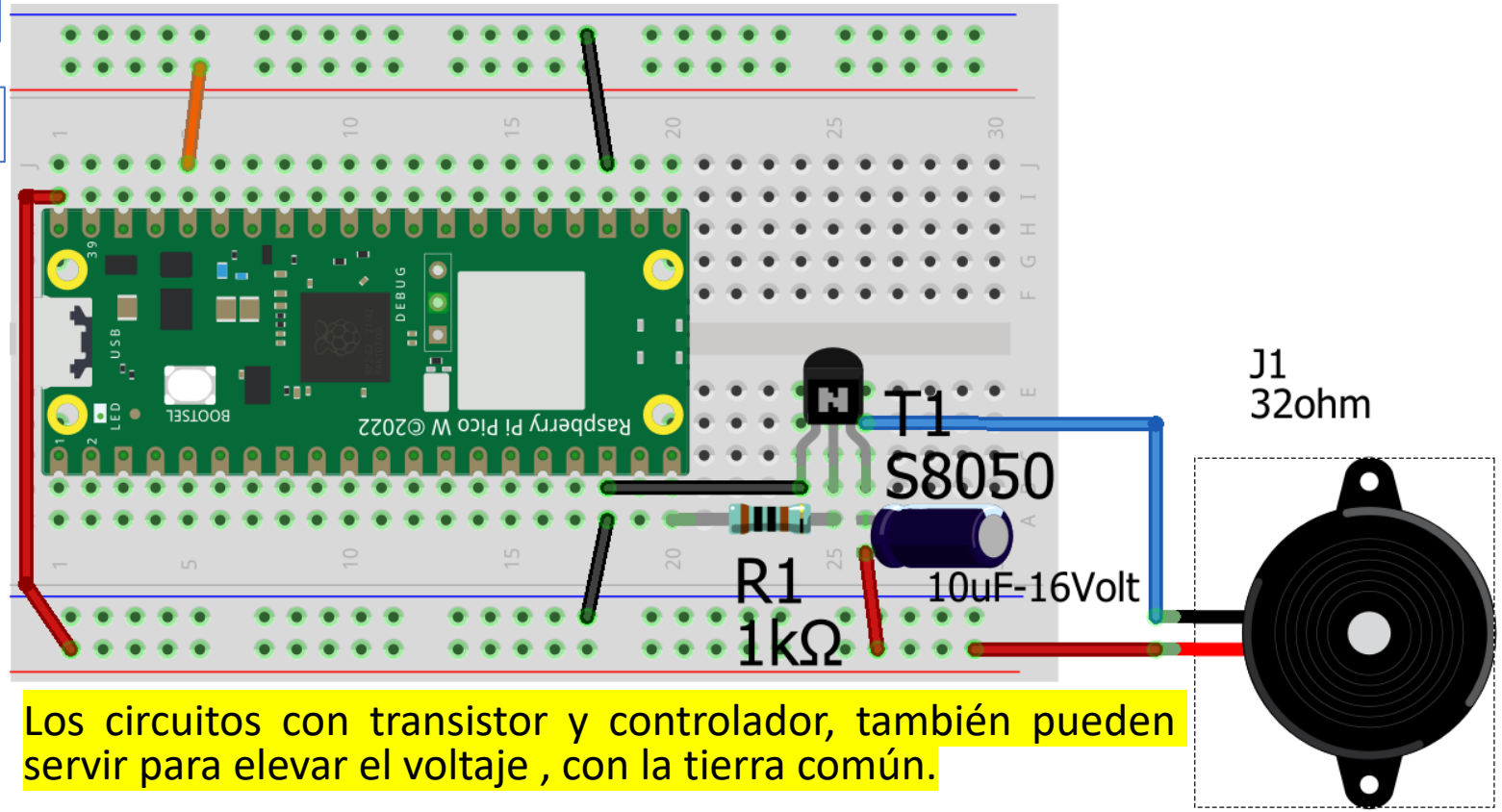
Clase 14.2.1[R] – **Sonido en Pico** con PWM 2da : refresco y 1ros programas de test

BMMR_CL14_speaker_3notas_1_0.py

BMMR_CL14_speaker_in_freq_1_1.py

```
def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty_u16(40000)
    utime.sleep_ms(duration)
    pin.duty_u16(0)
```

Se puede reusar el programa de la CL3 con un led externo por PWM, pero ahora lo que se cambia es la frecuencia de la onda PWM. Se usa una función '**tone**' para cambiar la frecuencia



Los circuitos con transistor y controlador, también pueden servir para elevar el voltaje , con la tierra común.

En este caso el altavoz se alimentará con 4,8volt (VBUS) , controlados por el NPN a 3,3volt

[PyR] Clase 14.1.1 **List comprehension** en Python

Ref listas compresión : https://recursospython.com/guias-y-manuales/comprension-de-listas-y-otras-colecciones/
Ref 2 https://realpython.com/list-comprehension-python/

En muchas ocasiones necesitamos **generar una Lista con algun tipo de operación repetida**, **o algo más complicado con una operación repetida y un filtrado de resultados**. 'List comprehension' lo hace de forma más compacta y aun asi muy facil de leer. Veamos ejemplos SIN y CON . **También disponible para Diccionarios y Conjuntos (Set)**

BMMP_CL14_list_compr_ex1.py

 # Ex1: cuadrados de los números del 1 al 10

```
cuadradosNumeros = [] # creamos una lista vacía
for x in range(1, 11):
    cuadradosNumeros.append(x * x)    # vamos añadiendo elementos
print(cuadradosNumeros)
```

```
>>> %Run BMMP_CL14_list_compr_ex1.py
  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

BMMP_CL14_list_compr_ex2.py

 # Ex2: cuadrados de los numero del 1 al 10, pares

```
cuadradosNumerosPar = [] # creamos una lista vacía
for x in range(1, 11):
    if x % 2 == 0:
        cuadradosNumerosPar.append(x * x) # vamos añadiendo cuadrados solo si par
print(cuadradosNumerosPar)
```

```
>>> %Run BMMP_CL14_list_compr_ex2.py
  [4, 16, 36, 64, 100]
```

 # Ex1.1: cuadrados de los numero del 1 al 10 con List comprehension

```
cuadradosNumerosLC = [x*x for x in range(1,11)]
print(cuadradosNumerosLC)
```

BMMP_CL14_list_compr_ex1_1.py

```
>>> %Run BMMP_CL14_list_compr_ex1_1.py
  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

 # Ex2.1: cuadrados de los numero del 1 al 10, pares con List comprehension

```
cuadradosNumerosParesLC = [x*x for x in range(1,11) if x % 2 == 0]
print(cuadradosNumerosParesLC)
```

BMMP_CL14_list_compr_ex2_1.py

```
>>> %Run BMMP_CL14_list_compr_ex2_1.py
  [4, 16, 36, 64, 100]
```

[PyR] Clase 14.1.2 **Diccionarios** en Python

Ref Basica: https://docs.python.org/es/3/tutorial/datastructures.html#dictionaries

Ref Detallada : https://realpython.com/python-dicts/

Listas y **cadenas** tienen propiedades en común, como el indexado y las operaciones de rebanado. Son dos ejemplos de datos de tipo ***secuencia*** (ver Tipos secuencia — list, tuple, range).

Tuplas las hemos visto **por encima** con los neopixels en la definición de colores ejemplo (rojo, verde, azul). Las tuplas son immutables y normalmente contienen una secuencia **heterogénea** de elementos vs las **listas** son mutable, y sus elementos son normalmente **homogéneos**

Python también incluye un tipo de dato para **Conjuntos** (no se verá en esta clase, ni seguramente en el taller)

Diccionarios a veces llamados «memorias asociativas» o «arreglos asociativos», son conjunto de pares clave : valor con el requerimiento de que las claves sean únicas (dentro de un diccionario). Un par de llaves crean un diccionario vacío: {}. Los diccionarios se indexan con *claves*, que pueden ser cualquier tipo inmutable

Creación

```
ExtensionTel = {'jack': 4098, 'sape': 4139}
# crea un diccionario directamente desde secuencias de pares clave-valor
dict([('sape', 4139), ('guido', 4127), ('jack', 4098)]) :
```

Iteración

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
        print(k, v)
gallahad the pure
robin the brave
```

En COMUN Listas y Diccionarios	DIFERENTE Listas y diccionarios
• Ambos son MUTABLES • Ambos son DINAMICOS, pueden crecer y decrecer • Ambos se pueden ANIDAR	• Diccionarios se **accede** a los elementos por CLAVES • Listas , elementos se **acceden** por su POSICION, indices.

Clase 14.2.2 [PyR] – **Sonido en Pico** con PWM 2da : Frecuencias de notas: algo de teoría musical

En música, la **altura** es una de las cuatro cualidades esenciales del sonido, junto con la duración, la intensidad y el timbre.

La Altura depende de la frecuencia en que se repita la onda sonora. Las notas musicales están determinadas por un número de frecuencia.

- La octava: desde un punto de vista físico es el rango de frecuencias entre dos notas que están separadas por una relación 2:1.
- LA4 (cuarta octava) es la referencia= 440 Hz.
- LA5 = 880 Hz y el LA3 = 220hz

Dentro de una octava hay 12 notas (con semitonos), luego la frecuencia se incrementará o decrementará

en múltiplos de $2^{1/12}$ $f_i = 440 \times 2^{i/12}$

¿ Como codificamos esta fórmula en Python para las teclas de un piano?

- Exponente = **
- LA4 (nota inglesa) es la tecla 49,

  Tecla 1(blanca todo izquierda) => LA0 =
  440 x 2**(1-49)/12 = 440x 2**(-4) = 27,5 Hz
- 2(negra) => LA#0 => 29,14 Hz
- 3(blanca) => SI0 => 30,87 Hz

  .....

Freq_tecla n => `2**((n-49)/12) * 440`

BMMR_CL14_speaker_3notas_1_0.py

Clase 14.2.2 [R] Sonido de notas en **Pico _ y W – Diccionario**

Frecuencias de notas del piano – ejemplo de uso de diccionarios

Ref : https://pybonacci.org/2020/04/01/haciendo-musica-con-python/ Author : Katie He katieshiqihe

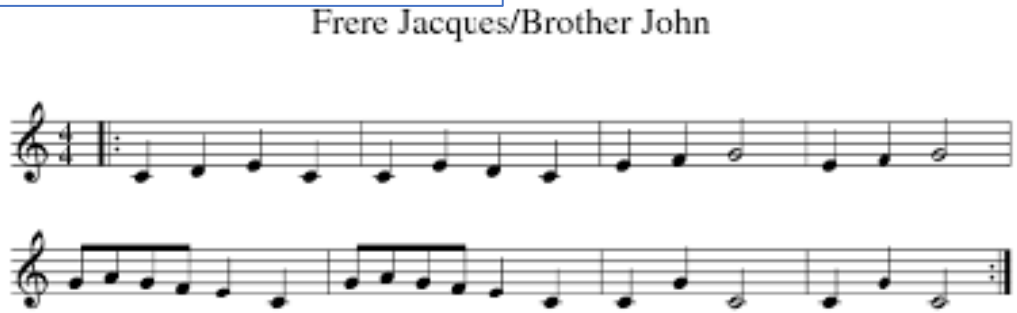
BMMR_CL14_speaker_in_note_2_0.py

```
def get_piano_notes():

    octave = ['C', 'c', 'D', 'd', 'E', 'F', 'f', 'G', 'g', 'A', 'a', 'B']
    base_freq = 440 #Frequencia base de La4 o Nota A4
    ks = [x+str(y) for y in range(0,9) for x in octave] # compression de listas DOBLE


    keys = ks[ks.index('A0'):ks.index('C8')+1]


    # genera un diccionario con las claves del nombre de letra y los valores la frecuencia
    # zip empaqueta listas en tuplas , dict genera un diccionario con tuplas de 2
    note_freqs = dict(zip(keys, [2**((n+1-49)/12)*base_freq for n in range(len(keys))]))
    return note_freqs
```

Clase 14.2.3.1 [R] Sonido de notas en **Pico _ y W – Canción Frére Jacques - simple**

BMMR_CL14_speaker_FJac_3_0.py

```
def silence(pin, duration):
    pin.duty_u16(0)
    utime.sleep_ms(duration)
BPM = 98 # Beats per minute para Frere JaqueS
BPMEAS = 4 #      4 /4 Ritmo de Frere Jacques
NEGRA = 4 * 60000 // (BPM * BPMEAS)
silCompas = NEGRA // 32 # arbitrario
# Cancion Frere Jacques son 8 compases
tone(buzzer,round(notesFreq['C4']),NEGRA) # COMPAS 1
tone(buzzer,round(notesFreq['D4']),NEGRA)
tone(buzzer,round(notesFreq['E4']),NEGRA)
tone(buzzer,round(notesFreq['C4']),NEGRA)
silence(buzzer, silCompas)
```




```
tone(buzzer,round(notesFreq['C4']),NEGRA) # COMPAS 2
tone(buzzer,round(notesFreq['D4']),NEGRA)
tone(buzzer,round(notesFreq['E4']),NEGRA)
tone(buzzer,round(notesFreq['C4']),NEGRA)
silence(buzzer, silCompas)
tone(buzzer,round(notesFreq['E4']),NEGRA) # COMPAS 3
tone(buzzer,round(notesFreq['F4']),NEGRA)
tone(buzzer,round(notesFreq['G4']),2*NEGRA)
silence(buzzer, silCompas)
……
```

Clase 14.2.3.1 [R] Sonido de notas en **Pico _ y W – Canción Frére Jacques - con listas anidadas de notas y duración en Tuplas**

```
BPM = 98 # Beats per minute para Frere JaqueS
BPMEAS = 4 # 4 /4 Ritmo de Frere Jacques
N = 4 * 60000 // (BPM * BPMEAS)
B = N * 2
C = N // 2
silCompas = NEGRA // 32 # arbitrario .  cancion Frere Jacques 8 compases
compasesNotaDur = [[('C4',N), ('D4',N), ('E4',N), ('C4',N)], # COMPAS 1
                   [('C4',N), ('D4',N), ('E4',N), ('C4',N)], # COMPAS 2
                   [('E4',N), ('F4',N), ('G4',B)], # COMPAS 3
                   [('E4',N), ('F4',N), ('G4',B)], # COMPAS 4
                   [('G4',C), ('A4',C), ('G4',C), ('F4',C), ('E4',N), ('C4',N)], # COMPAS 5
                   [('G4',C), ('A4',C), ('G4',C), ('F4',C), ('E4',N), ('C4',N)], # COMPAS 6
                   [('C4',N), ('G4',N), ('C4',B)], # COMPAS 7
                   [('C4',N), ('G4',N), ('C4',B)], # COMPAS 8
                  ]
```

BMMR_CL14_speaker_FJac_3_2.py

```
for comp in range(0, len(compasesNotaDur)):
    for nota in range(0, len(compasesNotaDur[comp])):
        tone(buzzer, round(notesFreq[compasesNotaDur[comp][nota][0]]),
        compasesNotaDur[comp][nota][1])

    silence(buzzer, silCompas)
```

- Cada nota del pentagrama va en una Tupla : (altura, duración) como ('C4',N).
- Cada compas se agrupa en una Lista [] de tuplas
- La canción se agrupa en una lista de listas de compases
- Luego se trata de recorrer con dos **for** las listas anidadas en orden. Cada miembro de la tupla es índice [0] y [1]