# Práctica - Uso del amplificador de audio: Utilizando el módulo DAC y la unidad DMA para reproducir archivos .wav

En prácticas anteriores ya habíamos utilizado el módulo DAC y la unidad DMA del microcontrolador de Ophyra para generar algunas señales. Para esta práctica, utilizaremos algunos recursos de las practicas anteriores, pero enfocados a reproducir archivos .wav a través del amplificador de audio de la tarjeta.

**Material extra.**

- Memoria Micro SD (4 GB recomendado).
- 2 jumpers hembra-hembra.
- Bocina con cable auxiliar o audífonos con conector de 3.5 mm.

Para realizar esta práctica, es necesario utilizar una memoria Micro SD en la tarjeta Ophyra y cargar los archivos de programa y audio (.wav) directamente a esta.

La práctica consistirá en reproducir un archivo .wav a través del amplificador de audio. Para esto, lo dividiremos en 2 partes, primero vamos a explicar la estructura general de la librería *wav_ophyra*, y posteriormente veremos el programa principal y cómo conectar físicamente en la tarjeta el módulo DAC al amplificador.

## Librería *wav_ophyra*.

### Encabezado del código:

```
"""
----- REPRODUCTOR DE AUDIO WAV -----

- WAV_FILE ---> Nombre del archivo .wav
- seg --------> Tiempo en segundos para reproducir el audio
"""

from pyb import DAC
from pyb import delay
```

Importamos las clases DAC y delay de la librería “pyb”. Ambas clases ya las hemos utilizado anteriormente, su uso será similar en esta práctica.

En la siguiente sección creamos la función **play** con dos parámetros de entra (nombre del archivo y los segundos de duración que tendrá la reproducción de audio).

## Configuración de audio / objetos y variables:

```
11  def play(WAV_FILE, seg):
12
13      #-------------CONFIGURACIÓN DE AUDIO---------------
14      dac = DAC(1, bits=8)        #Inicialización del DAC1 en modo de 8 bits / 8 bits de resolución.
15      SAMPLE_RATE_IN_HZ = 8000    #Tasa de muestreo en Hz (recomendado a 8 KHz o 16 KHz)
16
```

Primero creamos un objeto llamado “dac” de la clase DAC y como argumentos, introduciremos el número 1, para activar el canal 1. Y en el segundo argumento configuraremos el DAC para una resolución de 8 bits. En la siguiente línea creamos un objeto con el nombre “SAMPLE_RATE_IN_HZ” el cual será nuestra tasa de muestreo en Hz, aquí recomendamos dejarlo en 8 KHz o 16 KHz, pero se puede modificar dependiendo de las necesidades del usuario (sólo no deben olvidar que esta misma tasa de muestreo debe tener el archivo .wav).

## Cuerpo del programa:

```
17      print("-------- INICIO DE AUDIO --------")
18      lon = 44  #Se ignora el encabezado del .WAV
19      for i in range(seg+1):
20          wav = open(WAV_FILE, "rb") #Apertura del archivo de audio
21          pos = wav.seek(lon)
22
```

Agregamos un mensaje de inicio para que se muestre en la terminal. En la siguiente línea declaramos una variable “lon” con un valor inicial igual a 44, esta variable nos permitirá ir moviendo el curso del archivo a una nueva posición conforme avance el programa, lo iniciamos en 44 para ignorar el encabezado del archivo y pasar directamente al primer segundo de reproducción de audio. En la siguiente línea agregamos el bucle for con un rango desde 0 hasta el número de segundos que el usuario ingreso en el programa principal.

Dentro del bucle abrimos el archivo en modo se sólo lectura “rb” y lo asignamos al objeto “wav”. En la siguiente línea hacemos el salto a la posición donde iniciara la lectura del archivo (definido por “lon”) y lo asignamos al objeto “pos”.

**Declaración de buffers de datos:**

```
23          #---- Declaración de buffers de datos ----
24          wav_samples = bytearray(SAMPLE_RATE_IN_HZ)  #Matriz de bytes equivalentes a 1 seg de reproducción
25          wav_samples_mv = memoryview(wav_samples)
```

Creamos un objeto con el nombre “wav_samples” de tipo bytearray de un tamaño igual a “SAMPLE_RATE_IN_HZ” que, de acuerdo con la configuración elegida, será equivalente a 1 segundo de reproducción de audio. En la siguiente línea