Práctica – Uso del micrófono MP45DT02

La tarjeta de desarrollo Ophyra incorpora un micrófono digital MP45DT02 el cual se comunica con el microcontrolador a través de una interfaz I2S conocida también como “Inter-IC Sound, Integrated Interchip Sound”, es un estándar eléctrico de bus serial usado para interconectar circuitos de audio digital.

Material

- Memoria micro SD (4 GB recomendado)

Esta práctica consiste en grabar un fragmento de audio en formato .wav para después reproducirlo desde cualquier dispositivo que admita este tipo de archivos, o utilizar la librería “wav_ophyra” que puedes descargar desde ---.

Como el micrófono ya viene integrado en la tarjeta no es necesario realizar ninguna conexión extra.

Desarrollo del programa

En el encabezado del archivo “main.py” se importa la clase “MP45DT02” de la librería “ophyra_mp45dt02” (Esta librería ya esta incorporada en el firmware de MicroPython, así que no se necesita agregar ningún archivo adicional a la unidad de almacenamiento.) y la clase “time”.

```
1.  # main.py -- put your code here!
2.  from ophyra_mp45dt02 import MP45DT02
3.  import time
```

A continuación, se configura las características del archivo .wav. En las líneas 4 a la 8 se configura el nombre del archivo el cual debe llevar la extensión .wav, el tamaño de cada muestra en bits, en este caso la librería del micrófono entregas muestras de 16 bits, la velocidad de muestreo y numero de canales es decir si el audio es mono o estéreo en este caso al tener un solo micrófono el audio es mono correspondiente al lado izquierdo.

```
4.  WAV_FILE = "mic.wav"
5.  WAV_SAMPLE_SIZE_IN_BITS = 16
6.  SAMPLE_RATE_IN_HZ = 22000
```

```
7.  NUM_CHANNELS = 1
8.  WAV_SAMPLE_SIZE_IN_BYTES = WAV_SAMPLE_SIZE_IN_BITS // 8
```

La función “create_wav_header” crea el header del archivo .wav con las características descritas anteriormente.

```
9.  def create_wav_header(sampleRate, bitsPerSample, num_channels, num_samples):
10.       datasize = num_samples * num_channels * bitsPerSample //
    8
11.       o = bytes("RIFF", "ascii")
12.       o += (datasize + 36).to_bytes(4, "little")
13.       o += bytes("WAVE", "ascii")
14.       o += bytes("fmt ", "ascii")
15.       o += (16).to_bytes(4, "little")
16.       o += (1).to_bytes(2, "little")
17.       o += (num_channels).to_bytes(2, "little")
18.       o += (sampleRate).to_bytes(4, "little")
19.       += (sampleRate * num_channels * bitsPerSample // 8).to_bytes(4, "little")
20.       o += (num_channels * bitsPerSample // 8).to_bytes(2, "little")
21.       o += (bitsPerSample).to_bytes(2, "little")
22.       o += bytes("data", "ascii")
23.       o += (datasize).to_bytes(4, "little")
24.       return o
```

La clase MP45DT02 tiene los siguientes métodos:

- **.init()** – Permite inicializar la clase.
- **.readino(buf)** – Lee las muestras de audio y las transfiere a “buf”. “buf” debe admitir el protocolo buffer, como bytearray o array. Devuelve el número de bytes leídos.
- **.irq(handler)** – Establece una devolución de llamada cuando buf se llena. La configuración de la devolución de llamada llama nuevamente al método de lectura para una operación sin bloqueo.
- **.deinit()** – Desinicializa la clase.

Para comenzar a utilizar la clase MP45DT02 primero se establecen los estados de funcionamiento en ese caso se recomienda crear estados como grabar, pausar, reanudar y detener.

```
25. RECORD = 0
26. PAUSE = 1
27. RESUME = 2
28. STOP = 3
```

Después se establece una devolución de llamada, es decir lo que se realizará una vez que el buffer proporcionado esté lleno, se declaran como globales las variables para poder ser utilizadas en cualquier parte del programa. A continuación, se pregunta por el estado en el que se encuentra el proceso, en la línea 35 está el estado grabar, en ese caso se escriben los datos leídos en el archivo .wav, después se actualiza el número de bytes escritos y por último se vuelve a realizar la lectura para actualizar las muestras almacenadas en el buffer. En la línea 40 se encuentra el estado pausa en este caso solo se continúa actualizando las muestras, pero no se escriben en el archivo .wav, en la línea 43 el estado reanudar simplemente se regresa al estado grabar. Por último, en la linea 47 el estado detener crea el header del archivo .wav con la función declarada anteriormente y se escribe al principio del archivo, finalmente se cierra el archivo y se detiene la case MP45DT02.

```
29. def mic_callback_rx(arg):
30.     global state
31.     global num_sample_bytes_written_to_wav
32.     global mic_samples_mv
33.     global num_read
34.
35.     if state == RECORD:
36.         num_bytes_written = wav.write(mic_samples_mv[:num_read])
37.         num_sample_bytes_written_to_wav += num_bytes_written
38.         num_read = mic.readinto(mic_samples_mv)
39.
40.     elif state == PAUSE:
41.         num_read = mic.readinto(mic_samples_mv)
42.
43.     elif state == RESUME:
44.         Estate = RECORD
45.         num_read = mic.readinto(mic_samples_mv)
46.
47.     elif state == STOP:
48.         wav_header = create_wav_header(SAMPLE_RATE_IN_HZ,
49.                                        WAV_SAMPLE_SIZE_IN_BITS,
50.                                        NUM_CHANNELS,
51.                                        num_sample_bytes_written_to_wav //
52.                                        (WAV_SAMPLE_SIZE_IN_BYTES *
53.                                        NUM_CHANNELS),)
54.         _ = wav.seek(0)
55.         num_bytes_written = wav.write(wav_header)
56.         wav.close()
57.         mic.deinit()
58.         print("Done")
```

Una vez que se ha configurado la creación del header del archivo y la devolución de llamada, se procede a continuar con la parte principal del programa. En la línea 56 se crea el archivo .wav con el nombre previamente asignado y a continuación se mueve el índice del archivo a la posición 44 en donde inician los datos de audio del archivo.

```
59. wav = open(WAV_FILE, "wb")
60. pos = wav.seek(44)
```

Se crea el buffer con un bytearray (se recomienda que el buffer no sea mayor a 15000 bytes), se utiliza la función memorview para poder acceder al buffer sin la necesidad de que el sistema operativo cree una copia cada que se manipulen los datos.

```
61. mic_samples = bytearray(15000)
62. mic_samples_mv = memoryview(mic_samples)
```

Se inicia en 0 el número de bytes escritos en el archivo.

```
63. num_sample_bytes_written_to_wav = 0
```

Se crea un objeto desde la clase MP45DT02, y se configura la devolución de llamada para la interrupción que se creará cuando el buffer esté lleno, se establece el estado inicial en pausa y en la línea 64 se realiza la primera lectura, este método se declara por única vez y con la interrupción el sistema de grabación continuará funcionando en segundo plano hasta que el estado sea detener.

```
64. mic = MP45DT02()
65. mic.irq(mic_callback_rx)
66. state = PAUSE
67. num_read = mic.readinto(mic_samples_mv)
```

A partir de aquí el control de la grabación se realiza cambiando los estado y realizando pausas con el método .sleep(tiem) de la clase time.

```
68. print("starting recording for 5s")
69. state = RECORD
70. time.sleep(5)
71.
72. print("pausing recording for 2s")
73. state = PAUSE
74. time.sleep(2)
75.
76. print("resuming recording for 5s")
```

```
77. state = RESUME
78. time.sleep(5)
79.
80. print("stopping recording and closing WAV file")
81. state = STOP
```

La reproducción del audio se puede realizar de dos formas:

- En otro dispositivo – Desconectar la tarjeta Ophyra, extraer la memoria SD, insertarla en algún dispositivo que pueda reproducir este tipo de archivos y reproducir.
- En la tarjeta Ophyra – Diríjase al siguiente enlace.