

Laboratorio #2

M.C. Fernando Hermosillo Reynoso
fhermosillo@up.edu.mx

Universidad Panamericana

Sesión #2

28 de Enero del 2020



Prelab

Reportes de Laboratorio

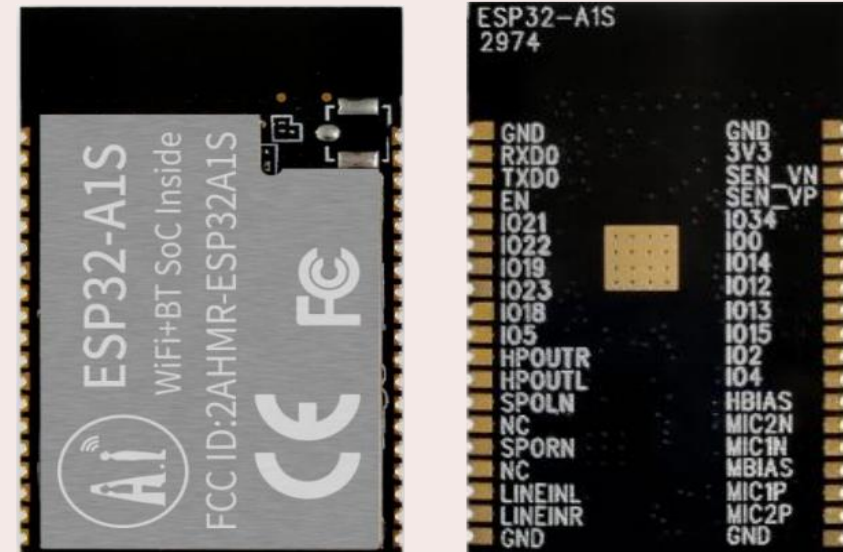
- **Introducción:** Explique brevemente la teoría y los algoritmos detrás de los programas que escribió.
- **Métodos:** Describa los pasos que siguió para implementar los algoritmos con sus propias palabras.
- **Resultados:** Presente los resultados que obtenga para cada actividad. Esta sección debe incluir capturas de pantalla de osciloscopio ilustrativas de los algoritmos DSP en acción. También incluya cualquier código que haya escrito o modificado.
- **Discusión:** Analice las conclusiones de cada laboratorio. Puedes mencionar cualquier intuición que hayas desarrollado. También mencione cualquier problema que haya enfrentado y cómo los solucionó.

ESP32-A1S

● SoC de Audio

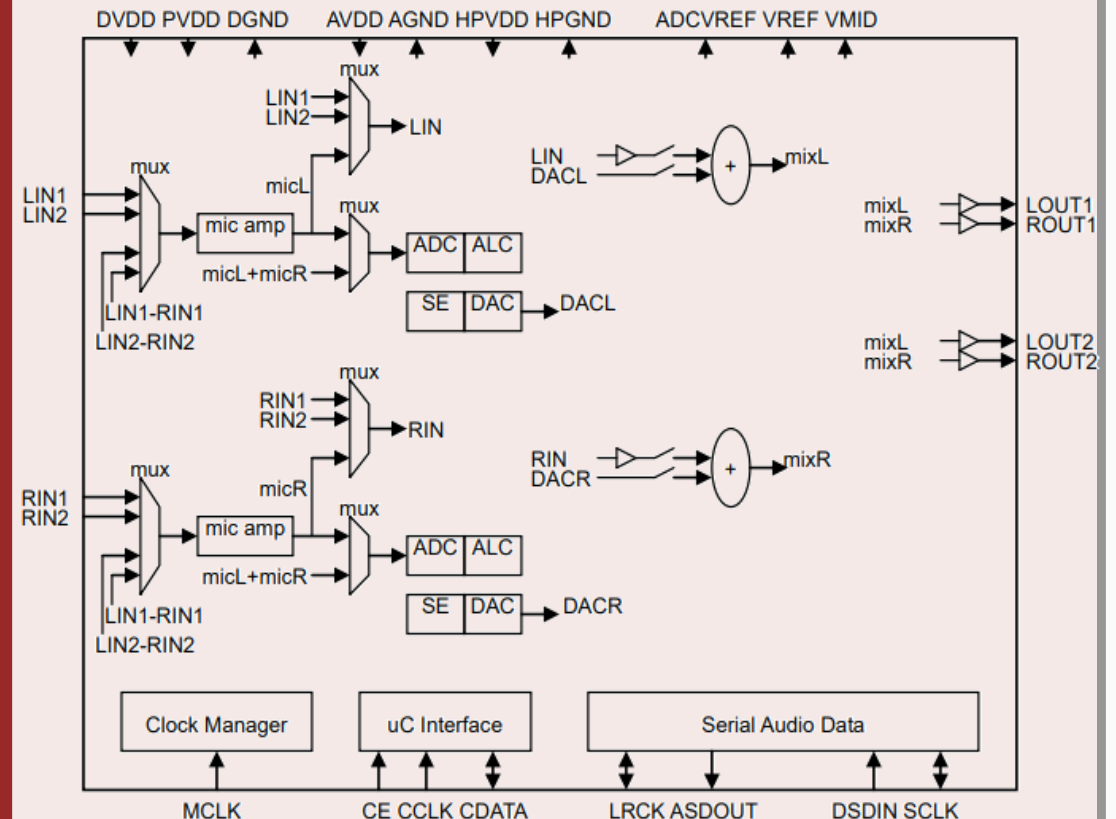
● Componentes

- ESP32
- WiFi y Bluetooth a 2.4GHz, compatible con el estándar Bluetooth tradicional (BR/EDR) y Bluetooth Low Energy (BLE),
- Códec de audio ES8388
- PSRAM



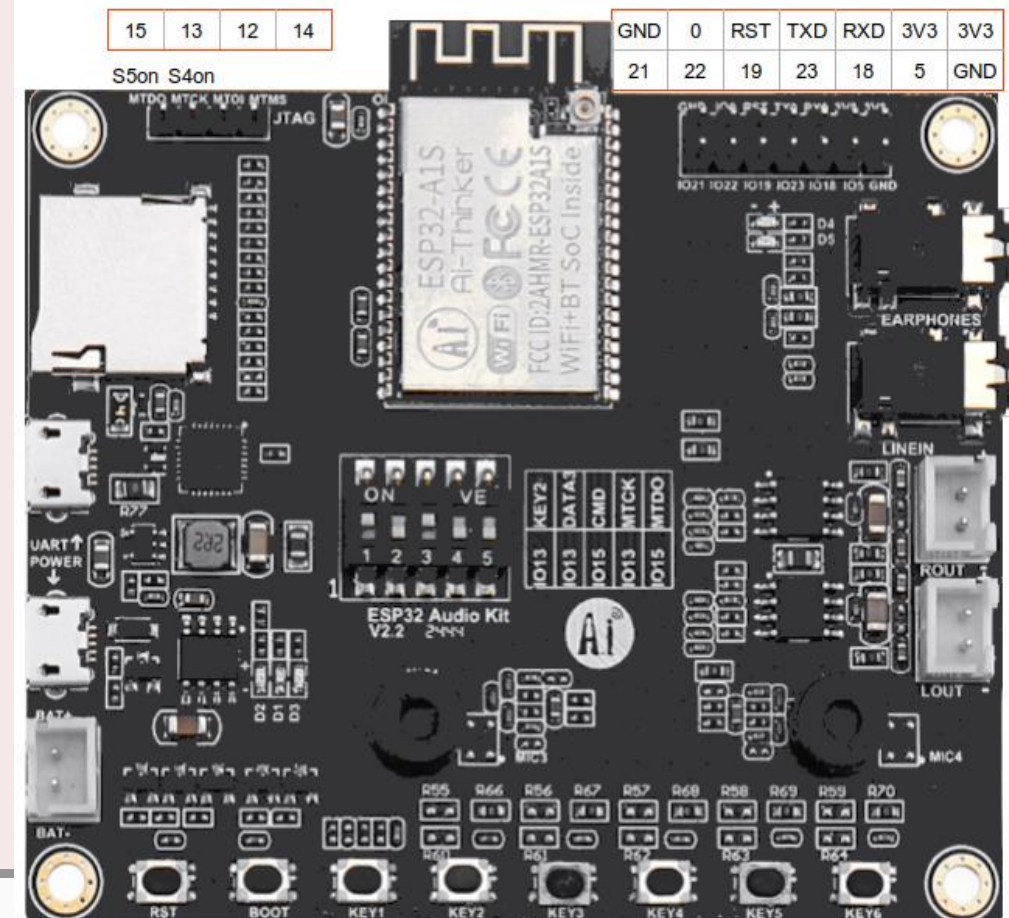
Códec de Audio: ES8388

- 2 ADC de 24 bits
- 2 DAC de 24 bits
- Amplificador de micrófono, auriculares y bocinas,
- Fs de 8KHz hasta 96KHz
- Interfaz
 - Control vía I2C
 - Datos vía I2S



AI-Thinker AudioKit V2.2

- ESP32-A1S
- 2 botones sistema: RST/BOOT
- 6 botones usuario: KEY1-KEY6
- 2 LEDs de usuario: D4 y D5
- 2 Micrófonos
- Auxiliar (AUXIN)
- Auriculares
- 2 Bocinas (Left/Right)
- I/O externas
- UART (USB)
- SD Card (vía SPI)



AI-Thiker AudioKit V2.2: Descripción de I/O

GPIO	Señal	GPIO	Señal	GPIO	Señal
IO0	I2S-MCLK	IO36	KEY1	IO19	LED5
IO25	I2S-WS	IO13	KEY2	IO14	SCK
IO26	I2S-DOUT	IO19	KEY3	IO15	MOSI
IO27	I2S-BCK	IO23	KEY4	IO2	MISO
IO35	I2S-DIN	IO18	KEY5	IO13	CS
IO32	I2C0-SCL	IO5	KEY6	IO34	SD_INTR
IO33	I2C0-SDA	IO22	LED4		

Lenguaje C: Tipos de Datos

- El lenguaje C no provee de una definición exacta en los tipos de datos

● **int**

- Por lo menos 16 bits
- Depende de arquitectura
 - MSP430: 16 bits
 - ESP32: 32 bits

● **#include <stdint.h>**

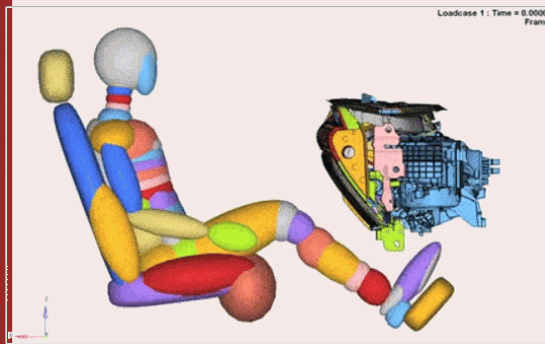
- **uint8_t**: [0,255]
- **int8_t**: [-128,127]
- **uint16_t**: [0,65535]
- **int16_t**: [-32768,32767]
- **uint32_t**: [?]
- **int32_t**: [?]
- **float**: 32 bits
- **double**: 64 bits

Lenguaje C: Tipos de Datos

- En PDS, los tipos de datos varían en dependencia de el hardware empleado
 - float: Fácil implementación pero complejo para el HW
 - Floating-point implement.
 - Entero: Implementación más compleja pero fácil para HW
 - Fixed-point implement.

Sistemas de Tiempo Real

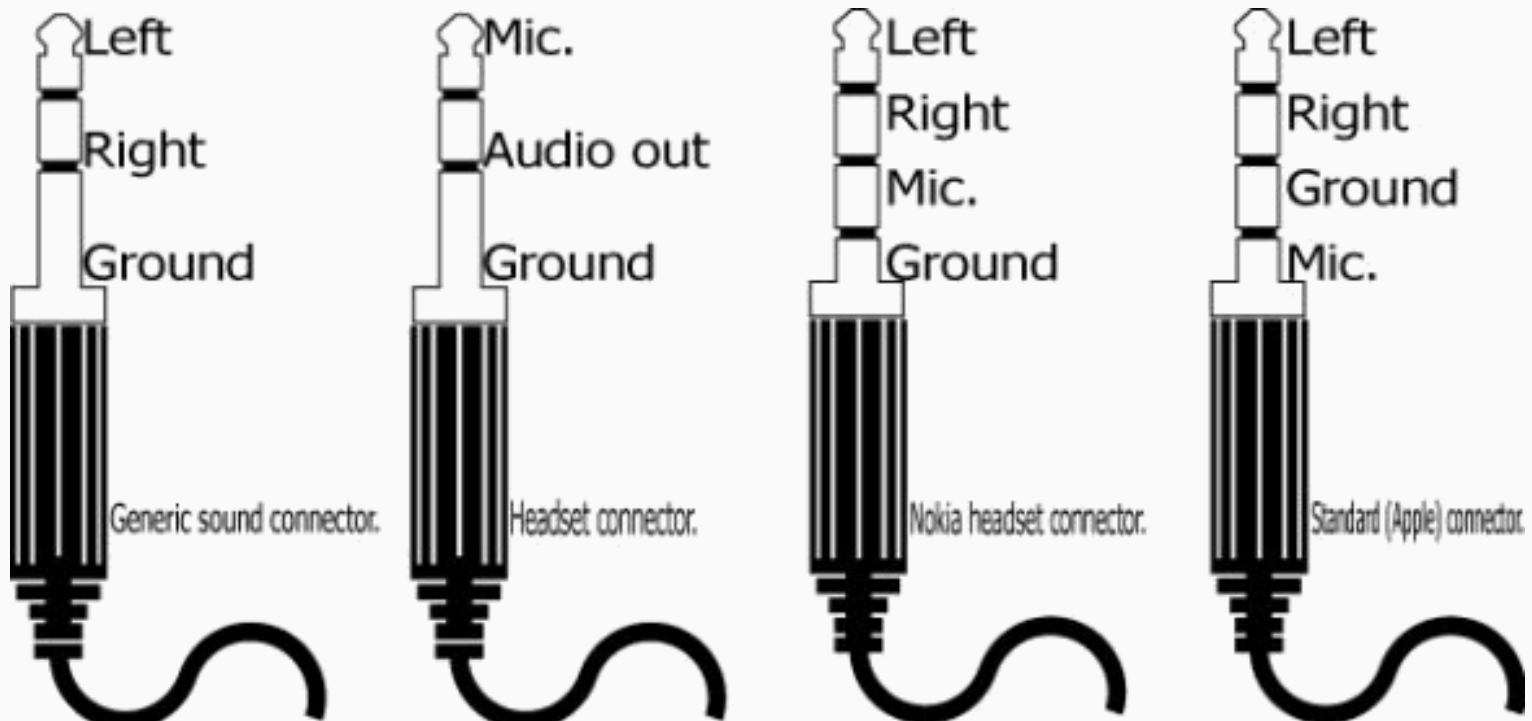
- La mayoría de los sistemas embebidos tienen un **deadline**
- En caso de no cumplirlo
 - **Hard**: Falla total del sistema
 - **Firm**: El sistema presenta error, pero puede recuperarse
 - **Soft**: El sistema presenta errores, pero no causa un impacto grave en el sistema



Sistemas de Tiempo Real

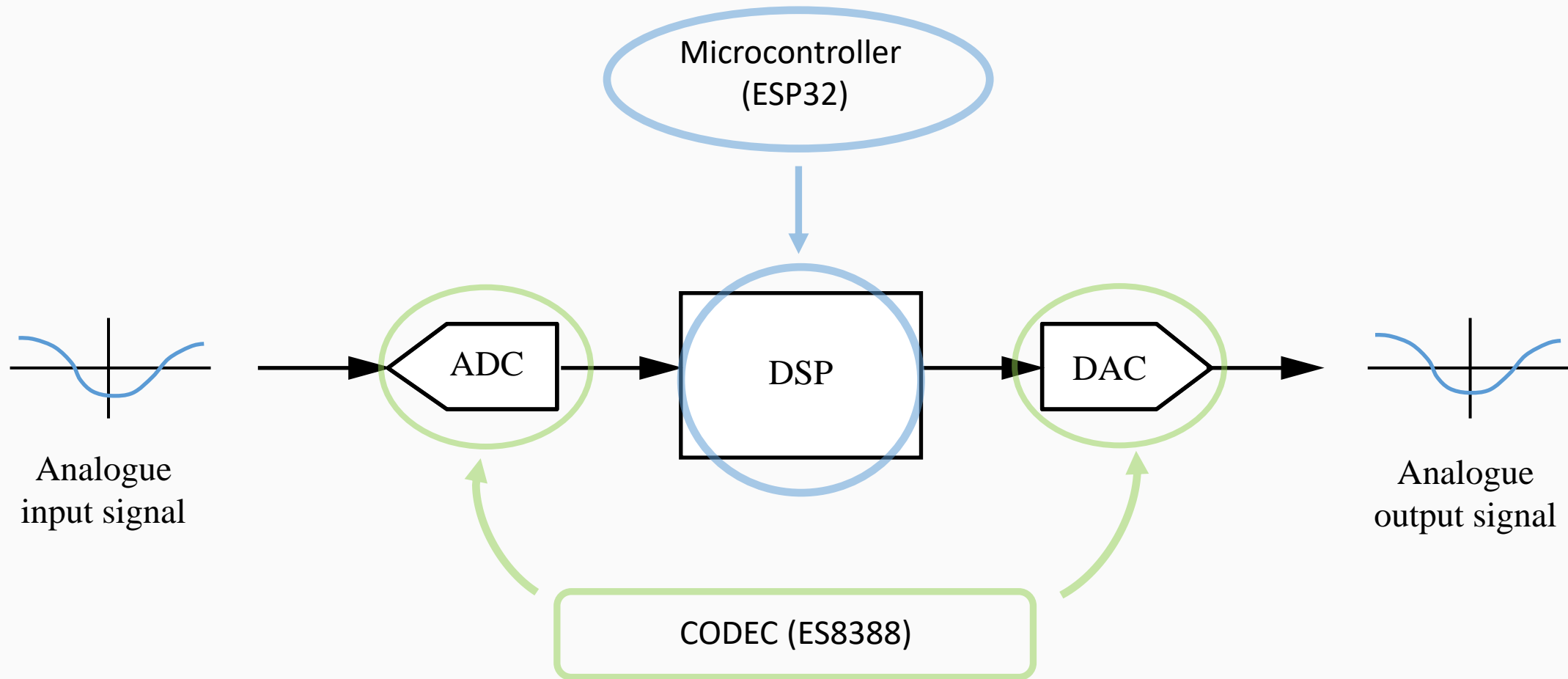
- La mayoría de los sistemas embebidos tienen un **deadline**
- En caso de no cumplirlo
 - **Hard**: Falla total del sistema
 - **Firm**: El sistema presenta error, pero puede recuperarse
 - **Soft**: El sistema presenta errores, pero no causa un impacto grave en el sistema
- Los sistemas con **deadline** son llamados en **tiempo real**
 - Los sistemas soft-deadline pueden ejecutar en un teléfono inteligente, una PC
 - Los sistemas con hard/firm-deadline se ejecutan en dispositivos específicos
 - **Tiempo Real**: Específico de la aplicación: Audio < 20ms

Conector Jack 3.5mm

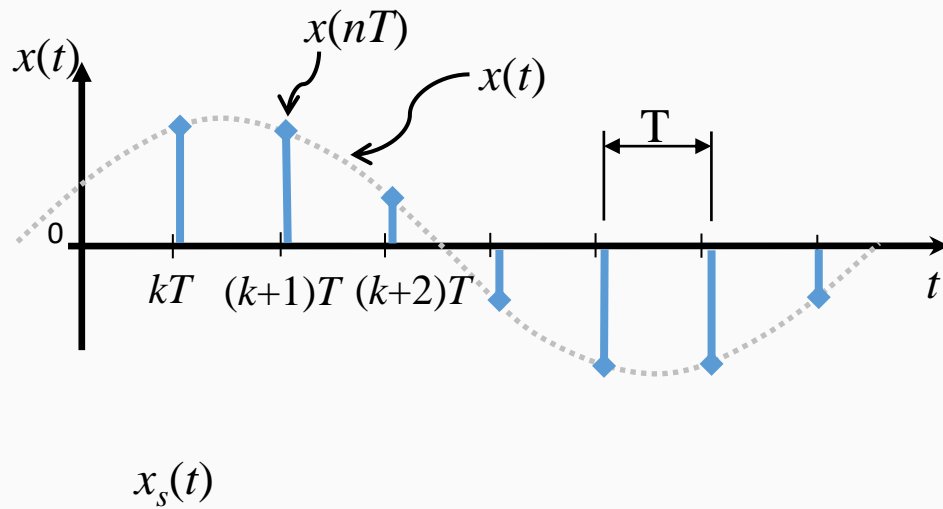


Laboratorio 1. Muestreo y Reconstrucción

Sistema de Procesamiento Digital de Señales

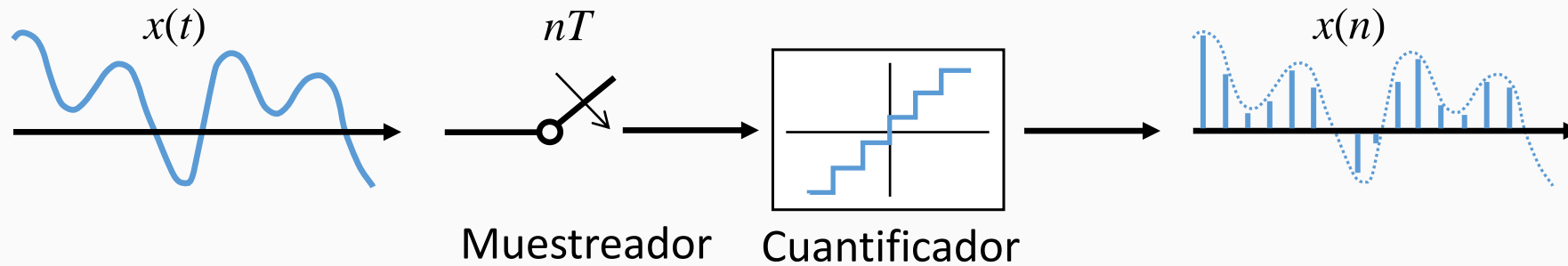


El Proceso de Conversión Analógico a Digital

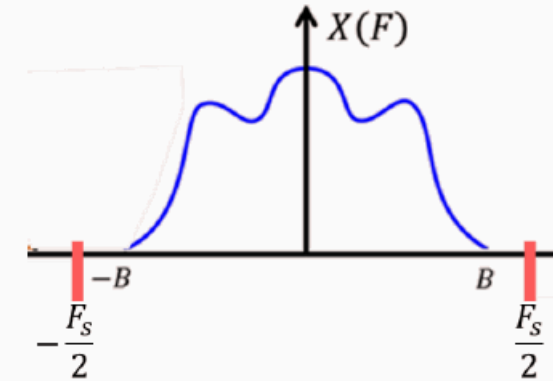
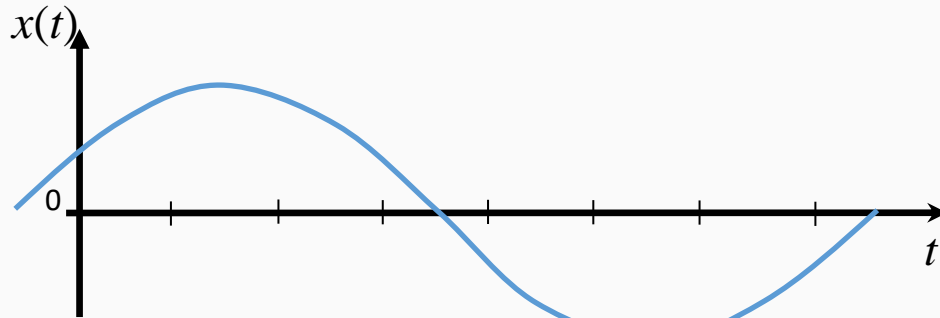


Analíticamente

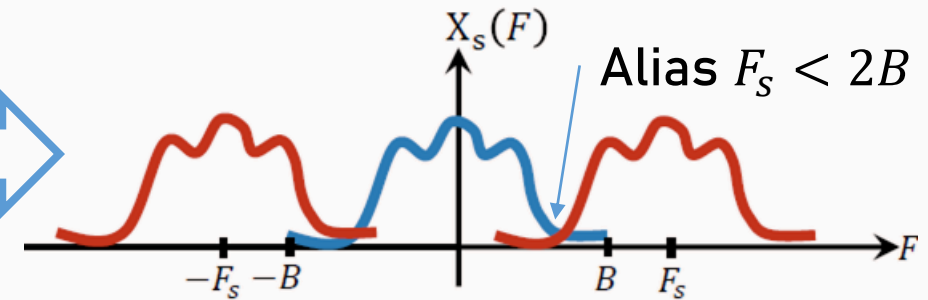
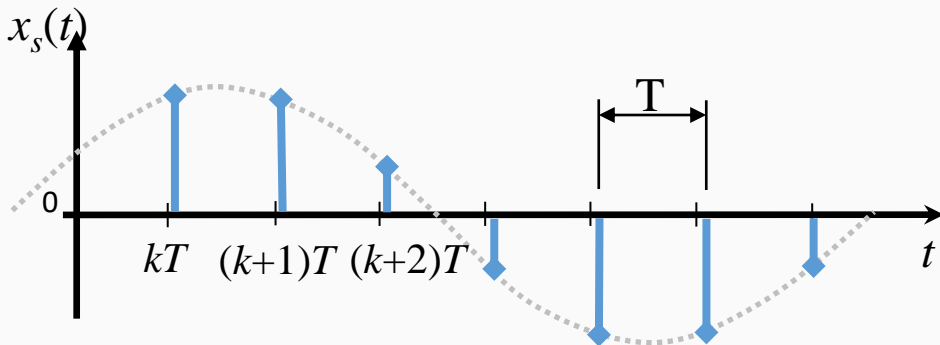
$$x(n) = x(nT) = x(n/Fs)$$



El Proceso de Conversión Analógico a Digital: Efectos en Frecuencia



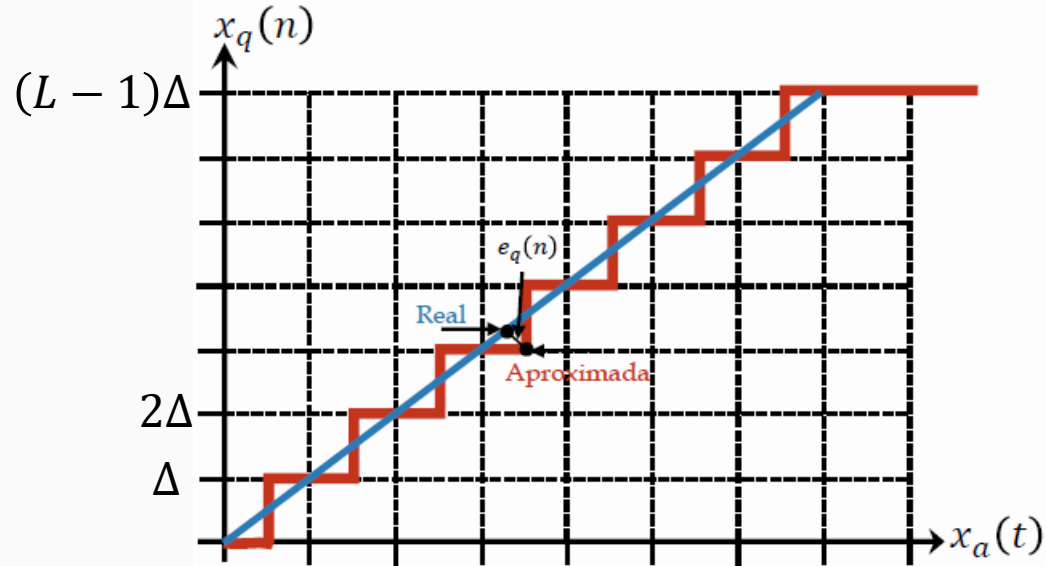
$$F_a(n) = \left| F_0 - nF_s * \left\lceil \frac{F_0}{F_s} \right\rceil \right| \text{ nth Frecuencia Alias de } F_0$$



$$x_s(t) = \sum_{n=-\infty}^{\infty} x_a(nT_s) \cdot \delta(t - nT_s)$$

$$X_s(F) = \frac{1}{T_s} \sum_{n=-\infty}^{+\infty} X_a(F - nF_s)$$

El Proceso de Conversión Analógico a Digital: Cuantificador Uniforme

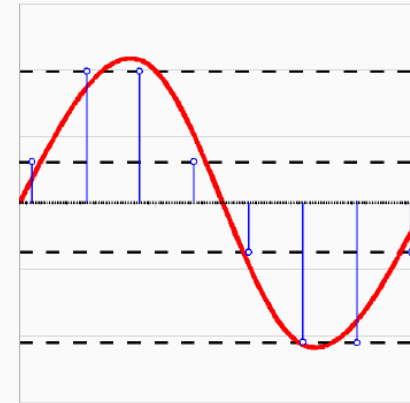


$$\Delta = \frac{(x_q(n)_{max} - x_q(n)_{min})}{L}$$

Δ : Resolución

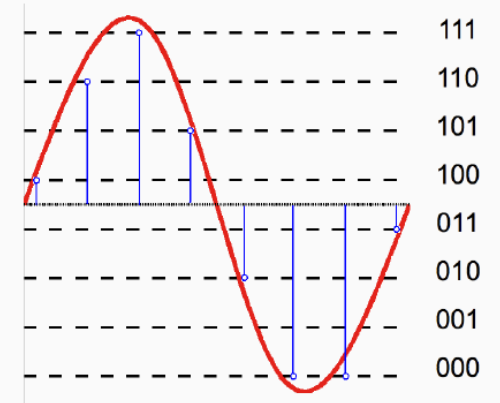
$L = 2^N - 1$: Número de niveles

N: Bits por muestra



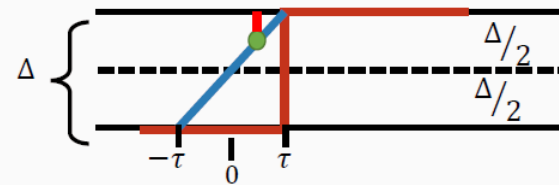
Cuantificación lineal con 2 bits

11
10
01
00



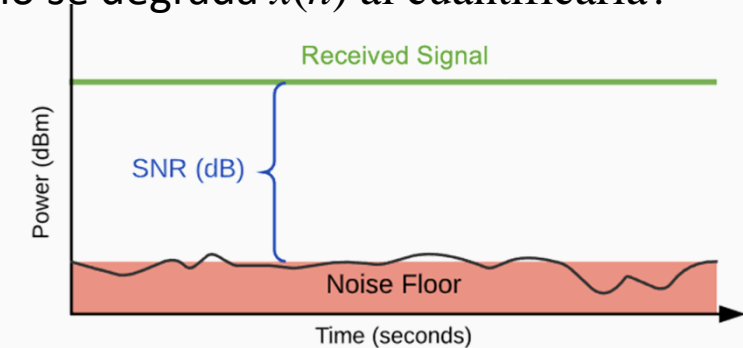
Cuantificación lineal con 3 bits

Error de cuantificación: Como se degrada $x(n)$ al cuantificarla?



$$e(t) = x(n) - x_q(n)$$

$$-\Delta/2 \leq e(t) \leq \Delta/2$$



Signal to Quantization Noise Ratio

$$SQNR = 1.76 + 6.01N$$

Cuestiones Prácticas de Muestreo

Usualmente un Timer establece a T_s

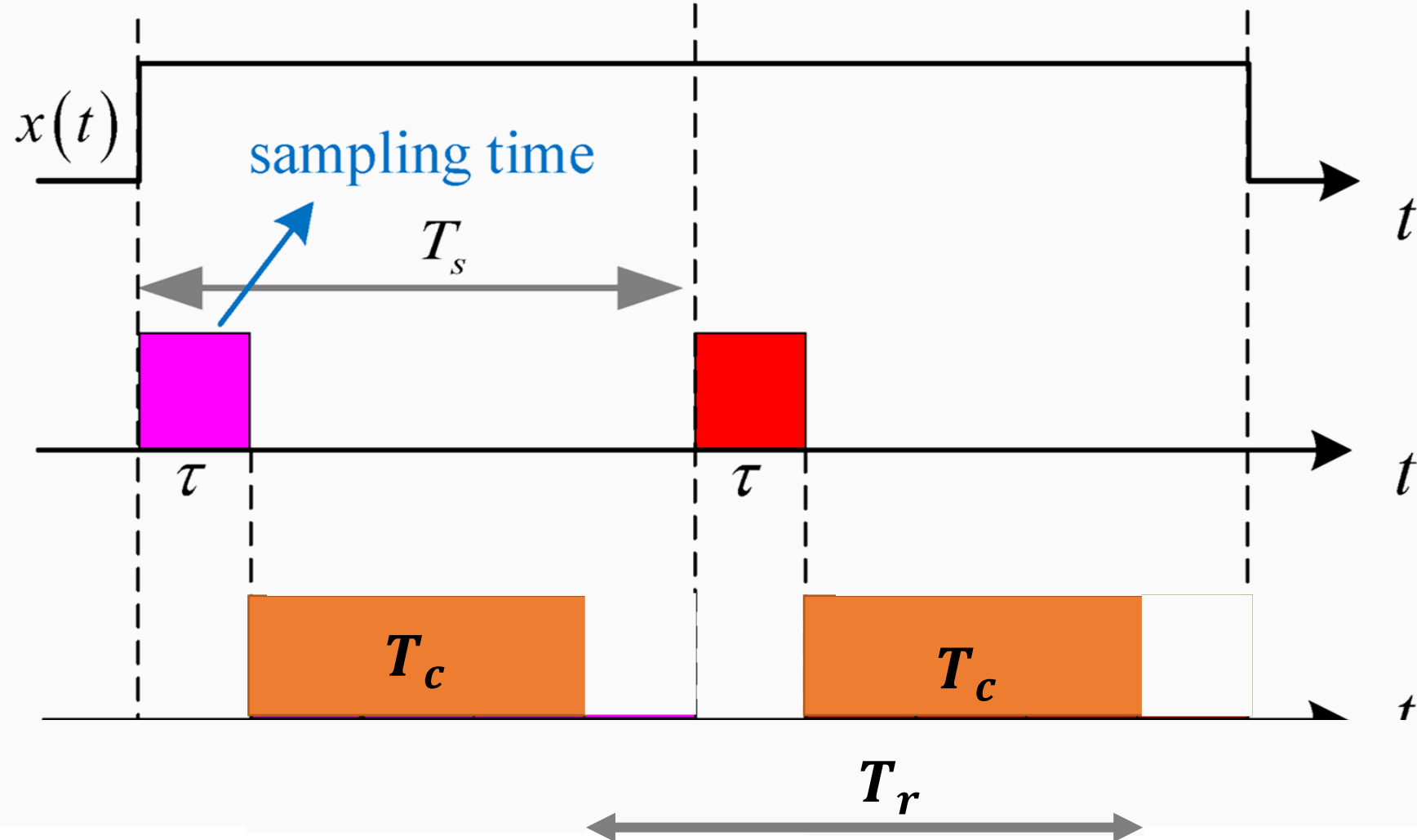
T_s : Periodo de muestreo

τ : Tiempo de adquisición

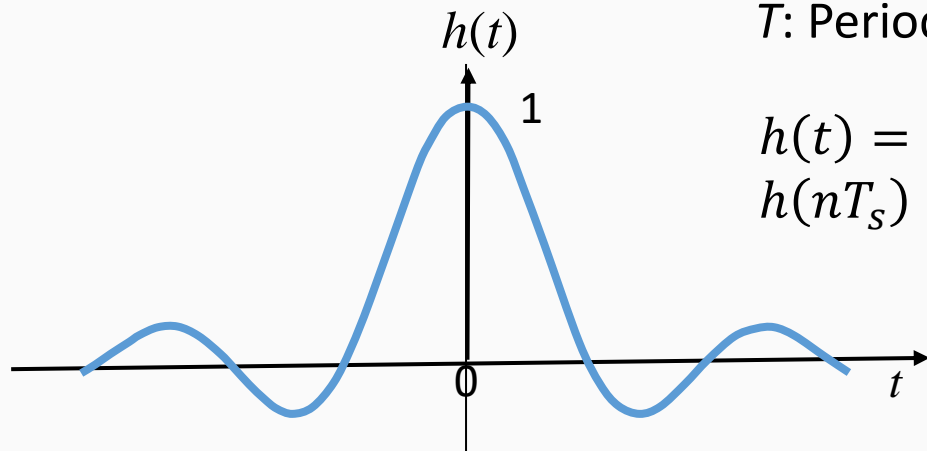
T_c : Tiempo de computo

T_r : de reconstrucción

Ineficiente cuando $F_s \rightarrow \infty$



El Proceso de Conversión Digital a Analógico: Interpolador Ideal



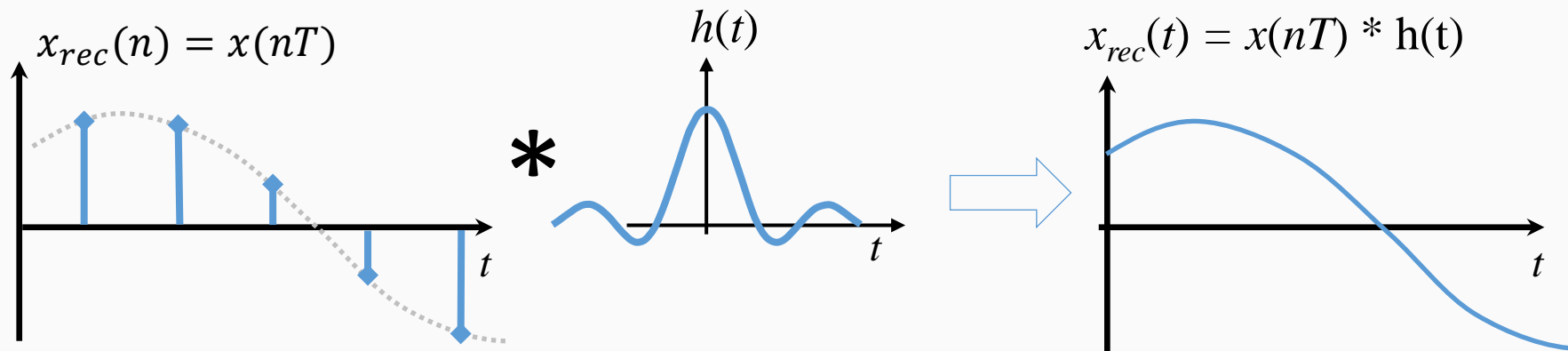
T : Periodo de muestreo

$$h(t) = \text{sinc}(\pi F_s t)$$

$$h(nT_s) = 0$$

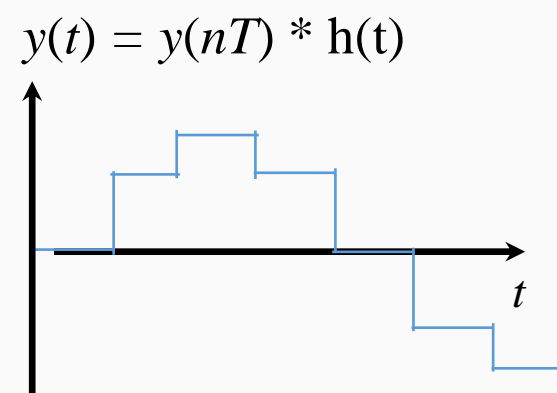
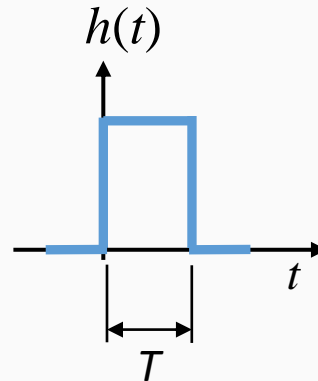
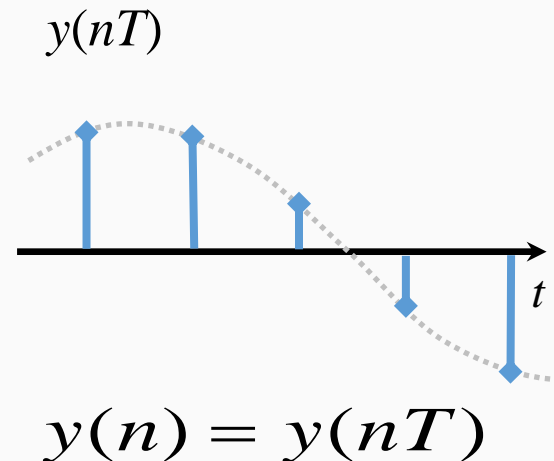
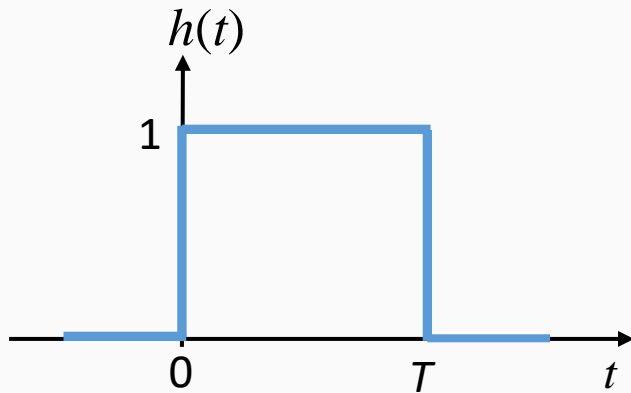
Analíticamente

$$x_{rec}(t) = x(t/T) = x(t F_s)$$



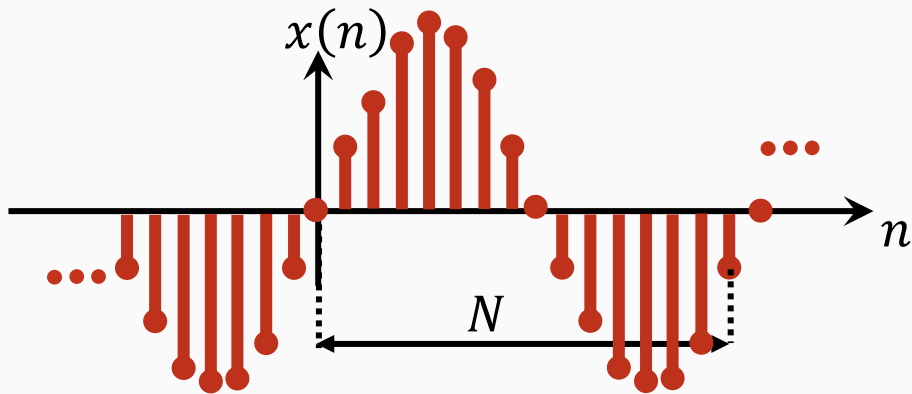
El Proceso de Conversión Digital a Analógico: Retenedor de Orden Cero

T : Periodo de muestreo



Señales Sinusoidales

$$x(n) = \sin(n\omega T)$$



Condición de necesaria para que sea periódica

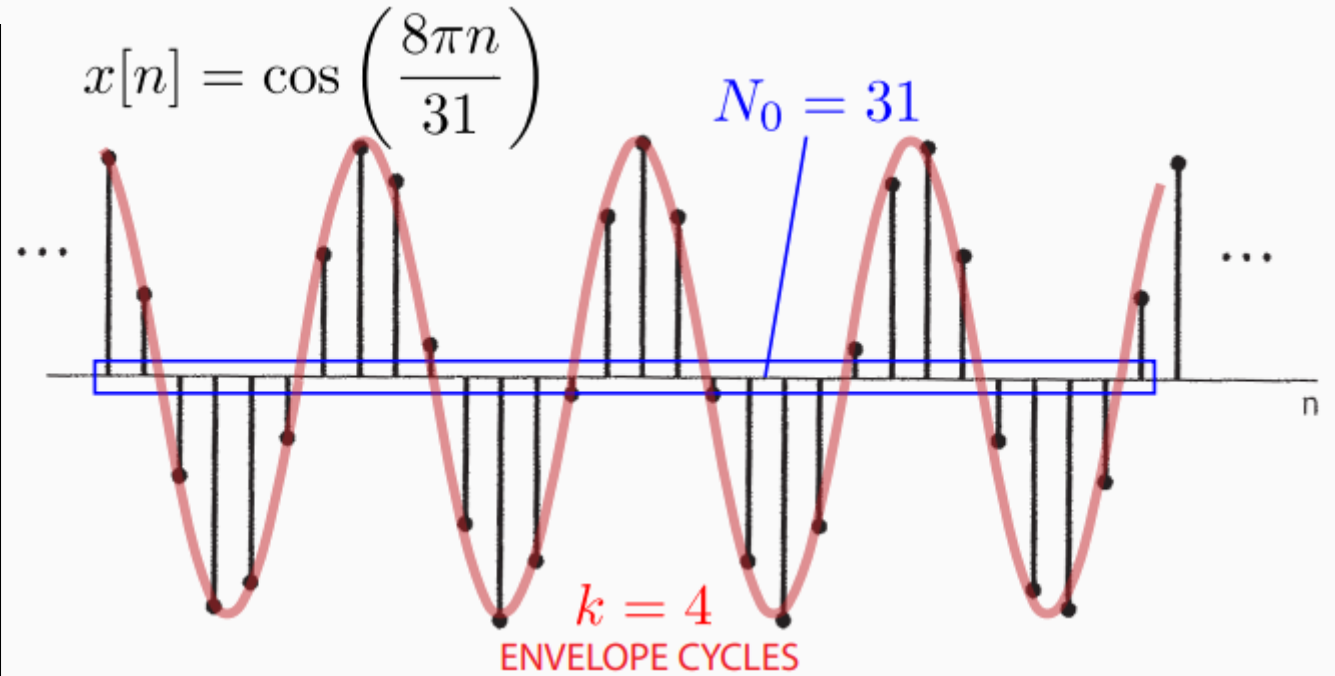
$$\frac{\omega}{2\pi} = \frac{k}{N} = f \in \mathbb{Q}$$

$k \in \mathbb{N}$: Número ciclos para generar periodicidad

$N \in \mathbb{N}$: Número de muestras en k

$\omega \in [-\pi, \pi]$: Frecuencia angular digital

$f \in \left[-\frac{1}{2}, \frac{1}{2}\right]$: Frecuencia digital



Periodo fundamental: $x(n + N_o) = x(n)$

$$N_o = \frac{N}{\text{MCD}\{k, N\}}$$

Protocolo I2S

- Protocolo de comunicación serial para la transmisión
 - Audio
 - Señal de video (cámara)
- Consiste en 3 señales
 - Señal de reloj de bit (BCLK)
 - Selector de canal (LRCK/WS)
 - Señal de datos (SDI/SDO)

I2S \neq I2C

I2S Timings

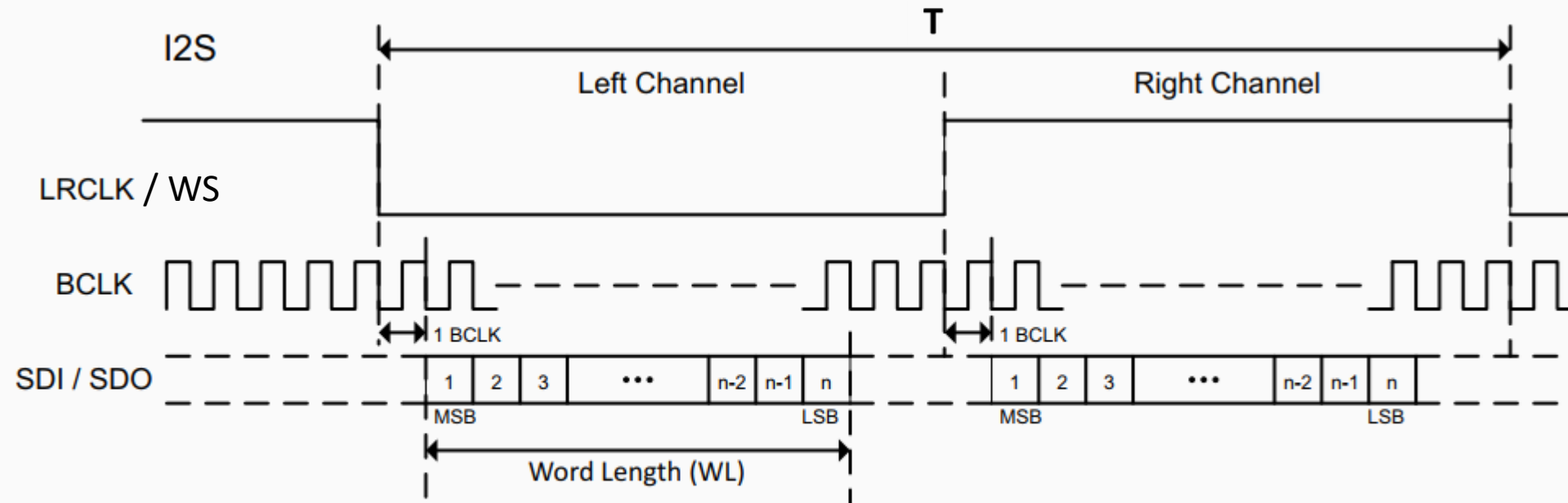
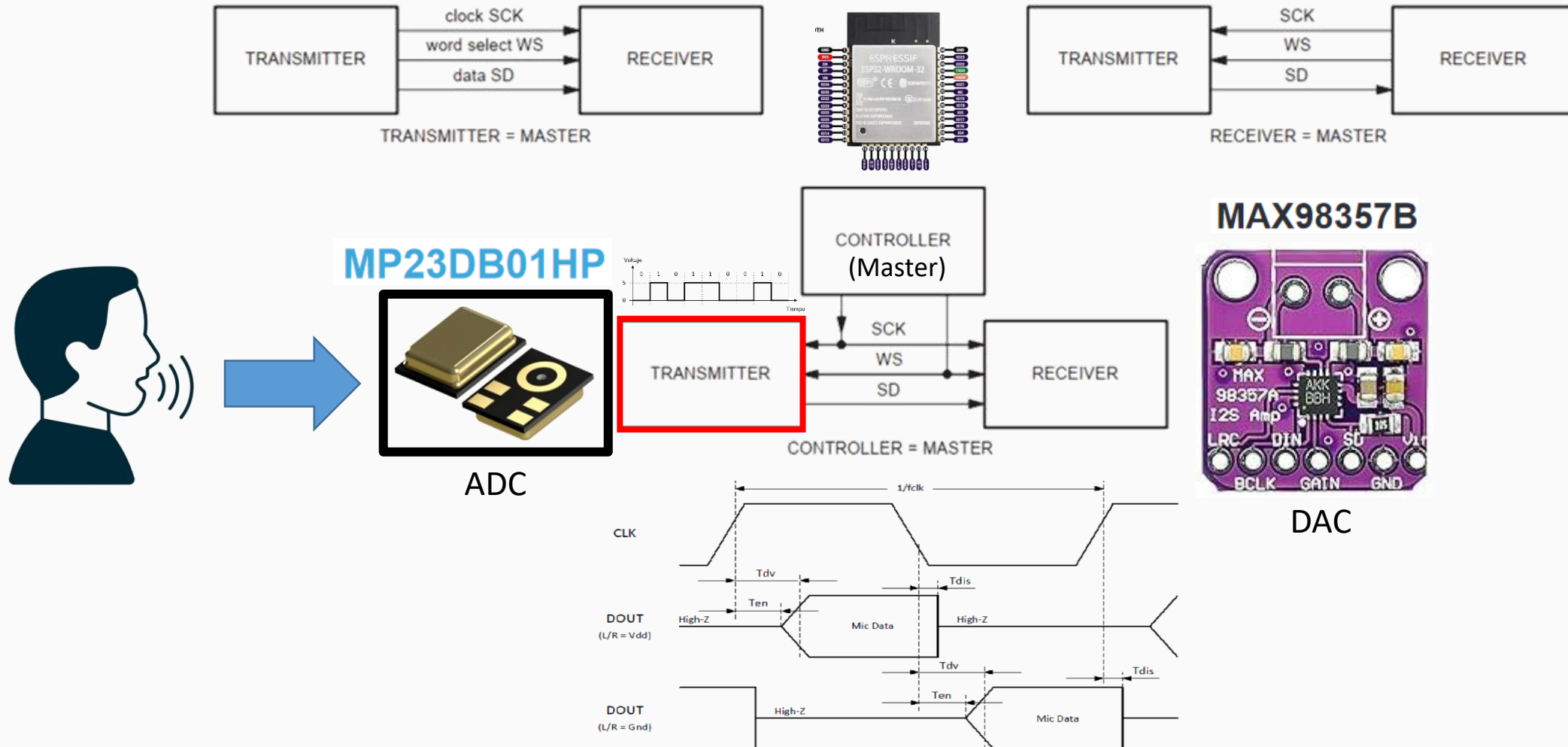


Figure 23. I²S Justified Audio Interface (assuming n-bit word length)

<https://blog.csdn.net/jackailson>

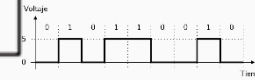
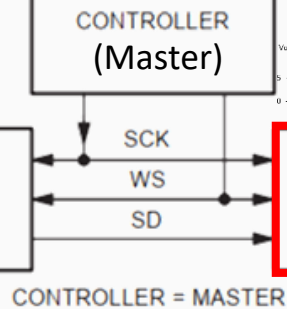
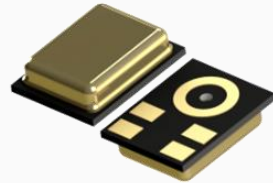
Modos de Comunicación



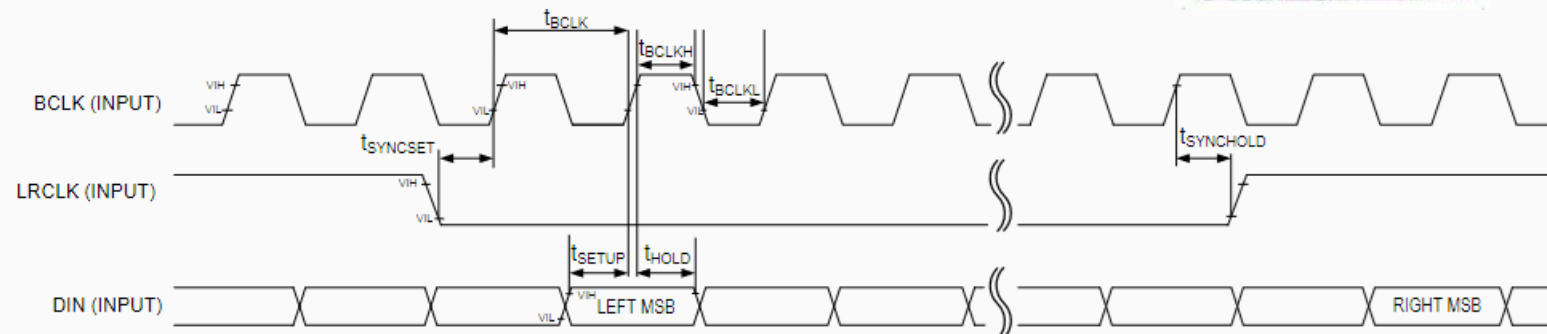
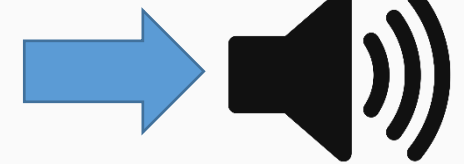
Modos de Comunicación



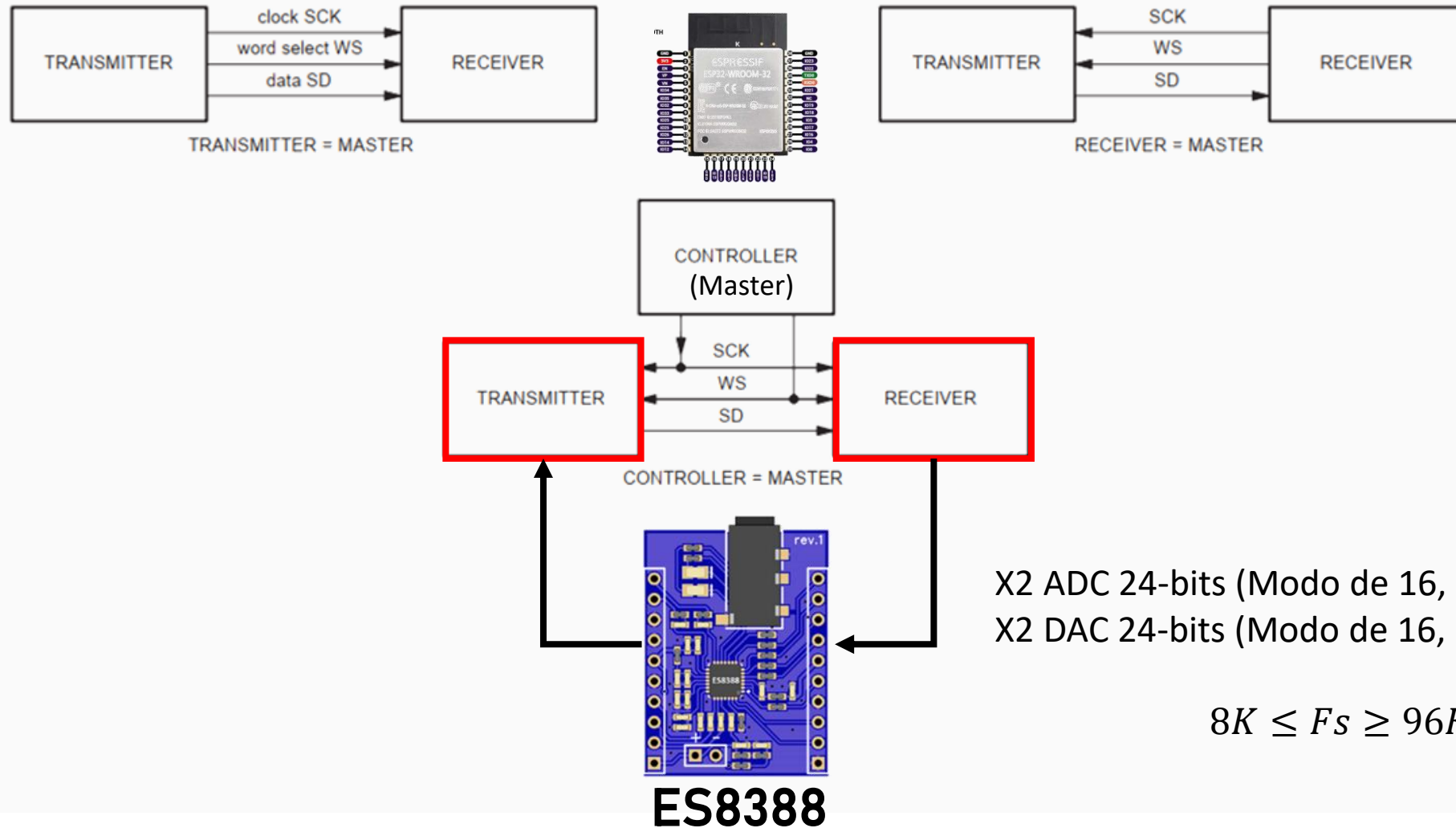
MP23DB01HP



MAX98357B



Modos de Comunicación



Actividad #1: Configuración del Proyecto

- Crear nuevo proyecto en PlatformIO “pds24_lab_1”
- Añadir “AudioToolkit HAL”
- Configurar “platform.io”

- Reemplazar
PATH_TO_AUDIOKIT_LIB
Por ubicación de la biblioteca

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
lib_ldf_mode = deep+
lib_extra_dirs = PATH_TO_AUDIOKIT_LIB
build_flags = -DAUDIOKIT_BOARD=5
monitor_speed = 115200
```

Actividad #2: Sistema de Adquisición

1. Configure códec ES8388

1. Entrada y salida
2. ADC Línea 2
3. DAC Línea 1
4. Frec. Muestreo: 8KHz
5. Bits por muestra: 16
6. Tamaño buffer DMA: 32
7. Número buffers DMA: 2

(En clase)

2. Mida el tiempo de muestreo (aproximado) empleando una señal digital posterior a leer los valores muestreados y un osciloscopio
3. En base a la cantidad de muestras configuradas: Cual es el periodo de muestreo?

Actividad #2: Sistema de Adquisición

```

/* Private Libraries ----*/
#include <Arduino.h>
#include <AudioKitHAL.h>

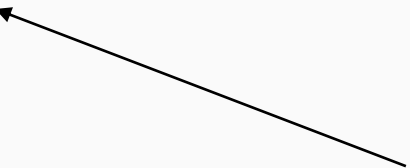
/* Private Defines -----*/
#define DMA_BUFFER_SIZE 32
#define DMA_NUM_BUFFERS 2

/* Private Macros -----*/

/* Private Functions ----*/

/* Private Variables ----*/
AudioKit kit;
int16_t AudioBuffer[DMA_BUFFER_SIZE];
    
```

Siempre en múltiplos de
2 para entrada estéreo



Actividad #2: Sistema de Adquisición

```

/* Setup -----*/
void setup() {
    // ES8388 & I2S config
    auto cfg = kit.defaultConfig(KitInputOutput);
    cfg.dac_output = AUDIO_HAL_DAC_OUTPUT_ALL;
    cfg.sample_rate = AUDIO_HAL_08K_SAMPLES;
    cfg.bits_per_sample = AUDIO_HAL_BIT_LENGTH_16BITS;
    cfg.buffer_size=DMA_BUFFER_SIZE;
    cfg.buffer_count=DMA_NUM_BUFFERS;
    kit.begin(cfg);
}

/* Main task -----*/
void loop() {
    /* Sampling via I2S (ADC) */
    size_t bytesRead = kit.read((uint8_t *)AudioBuffer, sizeof(AudioBuffer));

    /* DSP processing */
    for(int n = 0; n < bytesRead/2; n+=2) {

    }

    /* Signal Reconstruction via I2S (DAC) */
    kit.write((uint8_t *)AudioBuffer, bytesRead);
}

```

Actividad #3: Muestreo y Reconstrucción

1. Configure códec ES8388

1. Entrada y salida
2. ADC Línea 2
3. DAC Línea 1
4. Frec. Muestreo: 8KHz
5. Bits por muestra: 16
6. Tamaño buffer DMA: 32
7. Número buffers DMA: 2

(En clase)

2. Inyecte una señal senoidal de 1000Hz por entrada auxiliar

[ToneGenerator](#)

3. Mida la frecuencia de la señal recuperada en el puerto Jack o en los parlantes empleando un osciloscopio
4. El resultado observado en 3) es esperado? Porque?
5. Replique los pasos 2) a 4) usando una señal senoidal de 4000Hz, 6000Hz y 8000Hz

Actividad #4: Reconstrucción y Generación de Secuencias Discretas

1. Discretice analíticamente una señal senoidal de 1000Hz
2. En un arreglo de entero de 16 bits con signo almacene los valores de un periodo

```
int16_t x = INT16_MAX*sinf(PI/2.0f)
```

3. Por medio del DAC del ES8388, reconstruya la señal discretizada anteriormente

`INT16_MAX` = ?

`INT16_MIN` = ?

4. Compruebe la señal reconstruida en osciloscopio
5. Repita el experimento con una señal de 5000Hz

Actividad #5: Parámetros DMA

1. Incremente el tamaño del buffer de DMA a fin de que la latencia del muestreo sea de
 1. 10ms
 2. 50ms
 3. 100ms
2. Emplee los micrófonos en lugar del auxiliar
3. Que efectos prácticos tiene el tamaño del buffer en el sistema?

- Si se tiene un sistema discreto que tarda $10\mu s$ en procesar una muestra, y el sistema es tolerante a latencias $\leq 500\mu s$
 - Cual es el tamaño máximo que puede tener el buffer DMA para este caso?

Cuestionario

● Responda a las siguientes preguntas

1. En base a los resultados obtenidos, describa brevemente la importancia de muestrear una señal adecuadamente
2. Que entiende por alias?