Universidad La Salle Bajío | Facultad de Ingenierías y Tecnologías

Ingeniería en Electrónica y Telecomunicaciones

Espitia Alejandre Juan David

Silva Palafox Adrián

Trejo De Arcos Felipe Adriel

Velázquez Reyes Jimmy Daniel

“Procesamiento Digital de Señales”

Catedrático: Alan David Blanco Miranda

“**Práctica 2 – Análisis espectral y ventaneo mediante transformada discreta”**

31 de marzo del 2025, León Guanajuato.

- **Objetivo de la actividad:**

Comprender el concepto de transformada discreta de Fourier (TDF) y su aplicación. Analizar el efecto de ventaneo en la resolución espectral.

Implementar diferentes ventanas de ponderación y evaluar su desempeño.

Detectar componentes frecuenciales en señales reales mediante análisis espectral.

- **Procedimiento:**

Código de Python para adquisición y análisis:

```python
import serial
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from time import sleep
import os

def adquirir_datos(puerto='COM3', buffer_size=256):
    try:
        # Configurar puerto serial.
        ser = serial.Serial(puerto, 115200)
        print("Conectando al puerto: ", puerto)
        sleep(2)  # Esperar inicialización.
        print("Esperando datos del micro: ")
        # Leer datos:
        datos = []
        while len(datos) < buffer_size:
            if ser.in_waiting:
                try:
                    linea = ser.readline()
                    valor = int(linea.decode().strip())
                    datos.append(valor)
                except:
                    pass  # Ignorar líneas mal formadas.
        ser.close()
        return np.array(datos)
    except Exception as e:
        print(f"Error al conectar al micro: {e}")
        return None
```

```python
def aplicar_ventana(datos, tipo_ventana='rectangular'):
    N = len(datos)
    if tipo_ventana.lower() == 'rectangular':
        return datos
    elif tipo_ventana.lower() == 'hanning':
        return datos * signal.windows.hann(N)
    elif tipo_ventana.lower() == 'hamming':
        return datos * signal.windows.hamming(N)
    elif tipo_ventana.lower() == 'blackman':
        return datos * signal.windows.blackman(N)
    else:
        print("Tipo de ventana no reconocido, usando rectangular")
        return datos

def analizar_espectro(datos, fs, tipo_ventana='rectangular'):
    datos = datos - np.mean(datos)  # Remover componente DC
    datos_ventana = aplicar_ventana(datos, tipo_ventana)
    N = len(datos)
    X = np.fft.fft(datos_ventana)
    X_mag = np.abs(X) / N
    f = np.fft.fftfreq(N, 1/fs)
    return f[:N//2], 2 * X_mag[:N//2]  # Parte positiva del espectro
```

```python
def guardar_resultados(datos, fs, nombre_archivo='resultados_espectrales'):
    if not os.path.exists('resultados'):
        os.makedirs('resultados')
    ventanas = ['rectangular', 'hanning', 'hamming', 'blackman']
    plt.figure(figsize=(12, 8))

    for i, ventana in enumerate(ventanas, 1):
        f, magnitud = analizar_espectro(datos, fs, ventana)
        plt.subplot(2, 2, i)
        plt.plot(f, magnitud)
        plt.title(f'Ventana {ventana.capitalize()}')
        plt.xlabel('Frecuencia (Hz)')
        plt.ylabel('Magnitud')
        plt.grid(True)
    plt.tight_layout()
    plt.savefig(f'resultados/{nombre_archivo}.png')
    plt.close()
    print(f"Resultados guardados en resultados/{nombre_archivo}.png")

def main():
    PUERTO = 'COM3'
    BUFFER_SIZE = 256
    FS = 5000  # Frecuencia de muestreo en Hz

    print(" --- Análisis espectral con TDF --- ")
    print("1. Adquirir datos y analizar")
    print("2. Salir")
    opcion = input("Seleccione una opción: ")
```

```python
    if opcion == '1':
        puerto = input(f"Ingrese el puerto COM (default: {PUERTO}): ") or PUERTO
        datos = adquirir_datos(puerto, BUFFER_SIZE)

        if datos is not None:
            print(f"Se adquirieron {len(datos)} muestras")

            # Mostrar señal en el tiempo
            plt.figure(figsize=(10, 6))
            t = np.arange(0, len(datos)/FS, 1/FS)
            plt.plot(t, datos)
            plt.title('Señal adquirida en el tiempo')
            plt.xlabel("Tiempo (s)")
            plt.ylabel('Amplitud (ADC)')
            plt.grid(True)
            plt.show()

            # Mostrar espectros con distintas ventanas
            ventanas = ['rectangular', 'hanning', 'hamming', 'blackman']
            plt.figure(figsize=(12, 8))
            for i, ventana in enumerate(ventanas, 1):
                f, magnitud = analizar_espectro(datos, FS, ventana)
                plt.subplot(2, 2, i)
                plt.plot(f, magnitud)
                plt.title(f'Ventana {ventana.capitalize()}')
                plt.xlabel("Frecuencia (Hz)")
                plt.ylabel('Magnitud')
                plt.grid(True)
            plt.tight_layout()
            plt.show()

            # Guardar resultados
            guardar = input("¿Desea guardar los resultados? (s/n): ")
            if guardar.lower() == "s":
                nombre = input("Nombre del archivo (sin extensión): ") or 'resultados_espectrales'
                guardar_resultados(datos, FS, nombre)

if __name__ == "__main__":
    main()
```

Código utilizado en el micro:

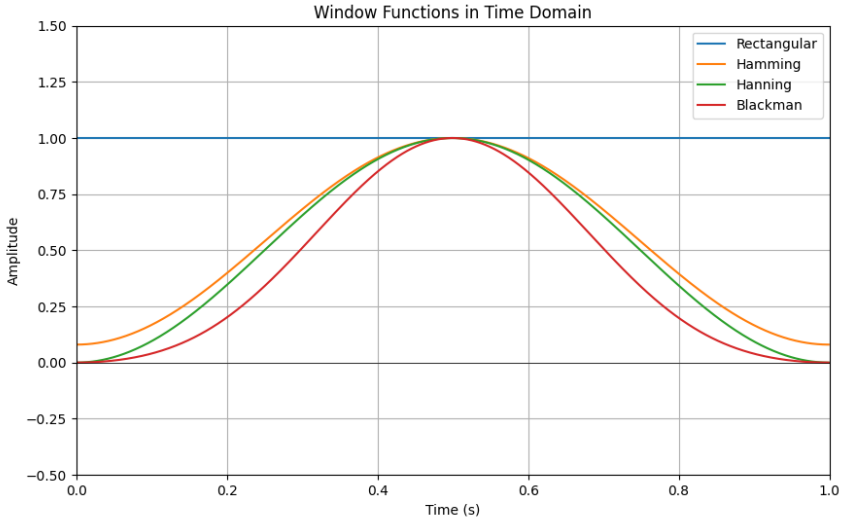
```c
 * Author: Adrián Silva Palafox
 * Date: 2025-03-06
 */
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/uart.h"
#include "hardware/adc.h"
// UART defines
#define BAUD_RATE 115200
#define UART0_TX_PIN 0 ///< UART0 TX pin
// ADC defines
#define ADC_PIN 26        ///< ADC pin
#define BUFFER_LENGTH 256 ///< Buffer length
#define TSAMPLE_RATE 200  ///< Sample rate in microseconds

volatile uint16_t adc_buffer[BUFFER_LENGTH]; ///< Buffer for ADC values
volatile uint16_t buffer_index = 0;          ///< Index for the buffer
volatile bool full_buffer = false;           ///< Flag to indicate if the buffer is full
struct repeating_timer timer;                ///< Timer instance
// Function to initialize the ADC buffer
void initialize_adc_buffer()
{
    for (int i = 0; i < BUFFER_LENGTH; i++)
    {
        adc_buffer[i] = 0;
    }
}
```

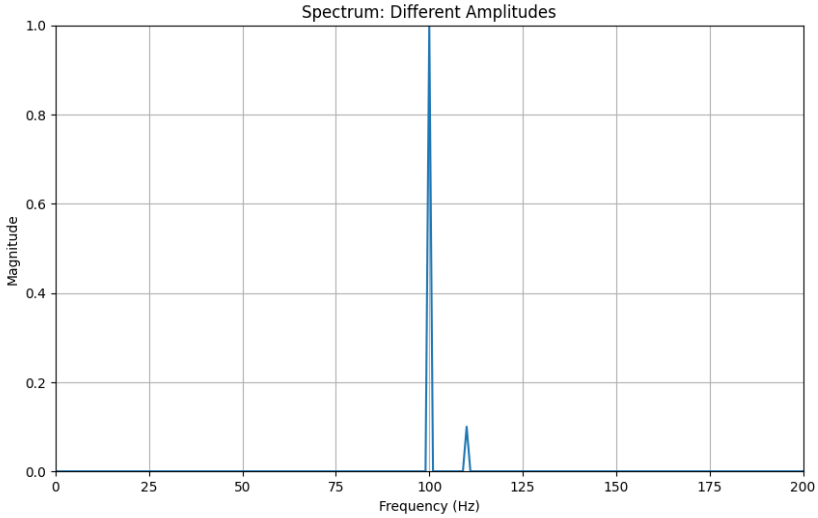
```c
// Repeating timer callback
bool repeating_timer_callback(struct repeating_timer *t)
{
    // Read ADC value and store it in the buffer
    adc_buffer[buffer_index] = adc_read();
    buffer_index++;
    if (buffer_index >= BUFFER_LENGTH)
    {
        full_buffer = true;        // Set the flag to indicate that the buffer is full
        cancel_repeating_timer(t); // Stop the timer
    }
    return true; // Return true to keep the timer running (if not canceled)
}
int main()
{
    // Initialize stdio for USB output
    stdio_init_all();
    // Add a delay to ensure USB is ready
    sleep_ms(2000);
    // Debug: Print a message to confirm the program has started
    // printf("Program started\n");
    // Initialize ADC
    adc_init();
    adc_gpio_init(ADC_PIN);
    adc_select_input(0); // Select ADC input 0
    // Debug: Print ADC initialization message
    // printf("ADC initialized on pin %d\n", ADC_PIN);
    // Initialize the adc buffer
    initialize_adc_buffer();
    // Create a repeating timer for periodic sampling (200us -> 5kHz)
    add_repeating_timer_us(-TSAMPLE_RATE, repeating_timer_callback, NULL, &timer);
    while (true)
    {
        // Main loop can process the buffer if needed
        if (full_buffer)
        {
            // printf("Buffer full, sending data...\n");
            // Send the buffer over USB (stdout)
            for (int i = 0; i < BUFFER_LENGTH; i++)
            {
                printf("%d\n", adc_buffer[i]);
            }
            // Reset the flag after processing
            full_buffer = false;
            initialize_adc_buffer();
            buffer_index = 0; // Reset buffer index
            // Restart the timer
            add_repeating_timer_us(-TSAMPLE_RATE, repeating_timer_callback, NULL, &timer);
        }
    }
}
```

Imágenes obtenidas:

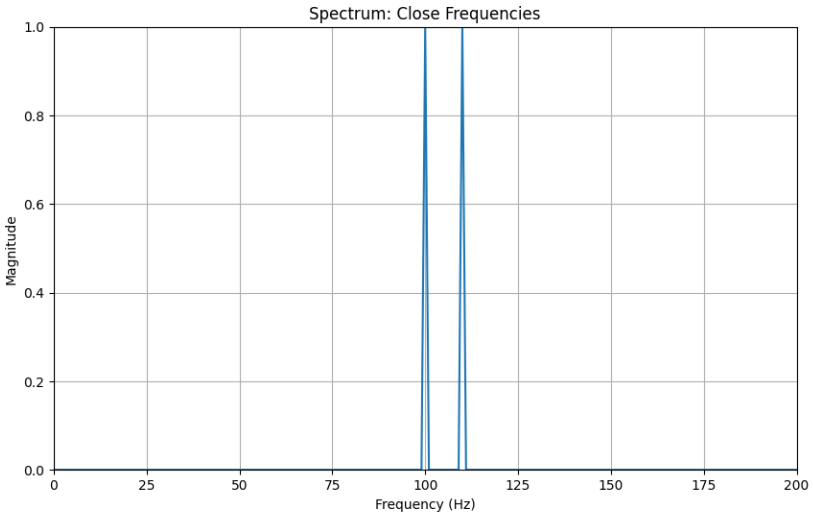
- Ventanas:



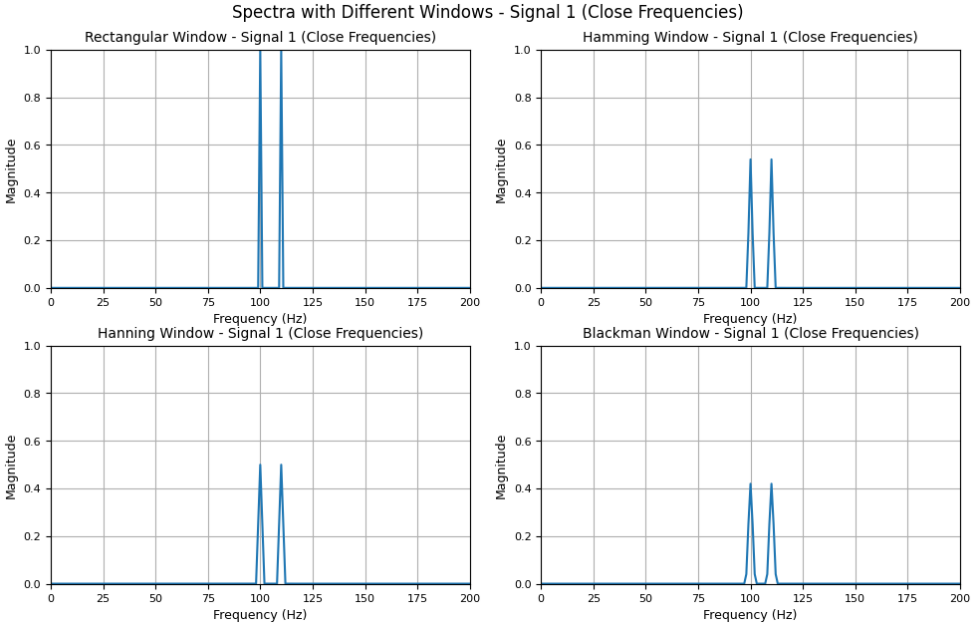
- Espectro con diferente amplitud:

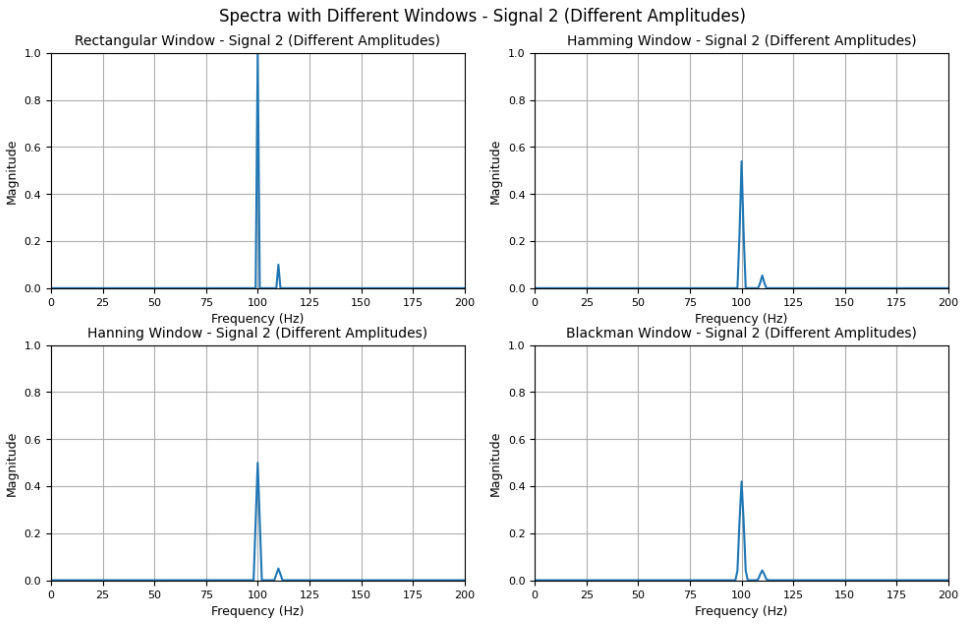


- Espectro con frecuencia cerrada:

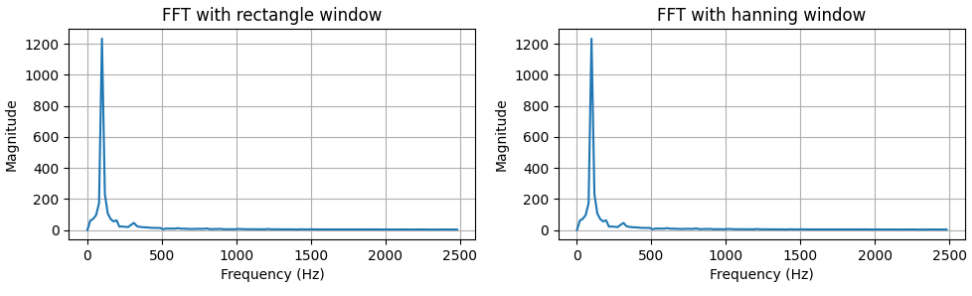


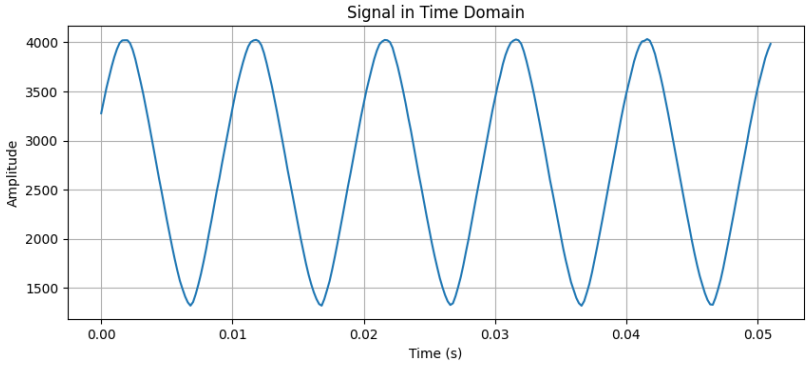
- Espectro con diferentes ventanas señal 1 y 2





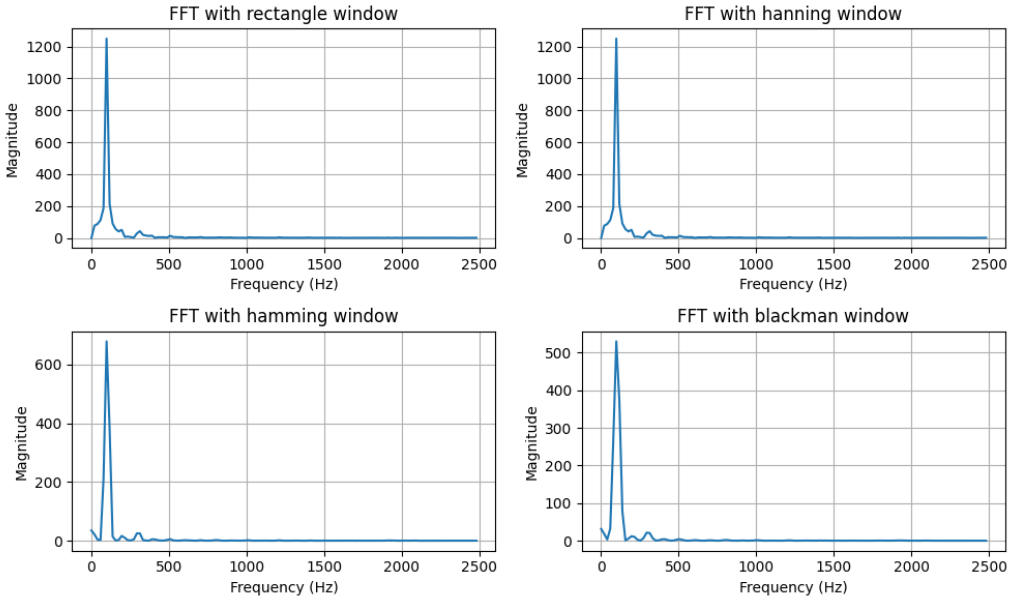
- **Experimento base:** 100 Hz

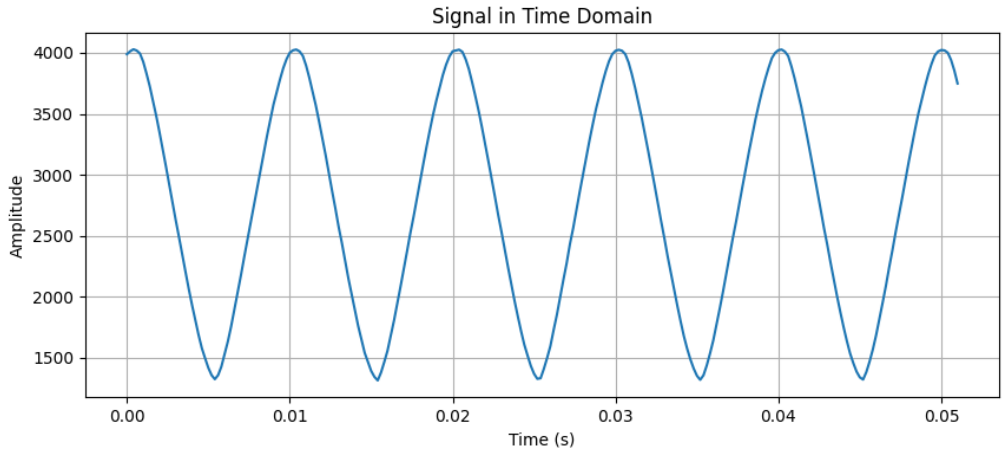




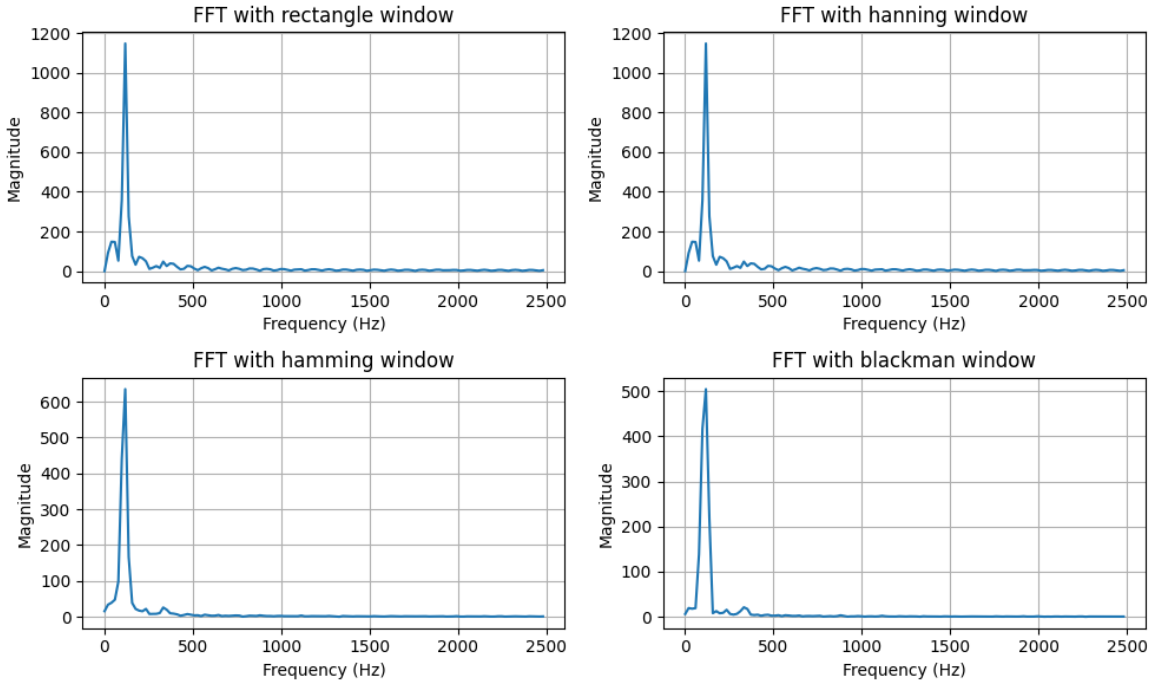
- **Experimento con dos frecuencias cercanas:**

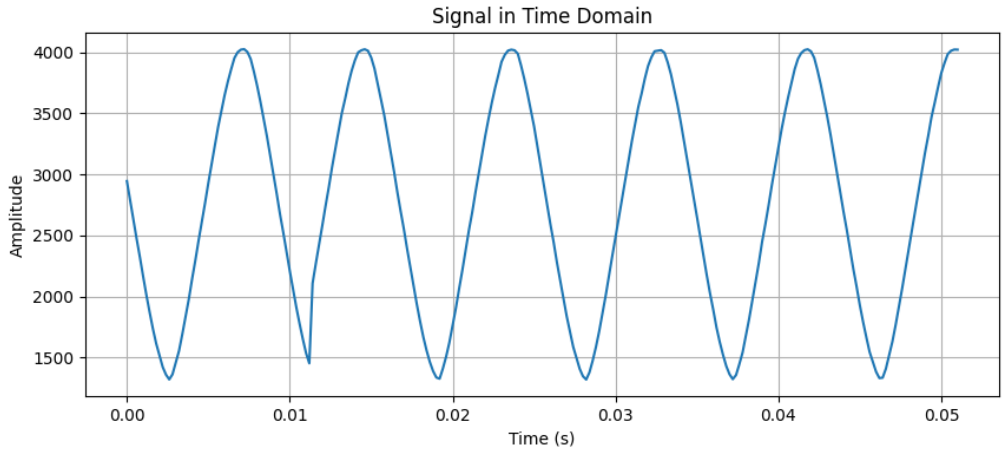
100 Hz



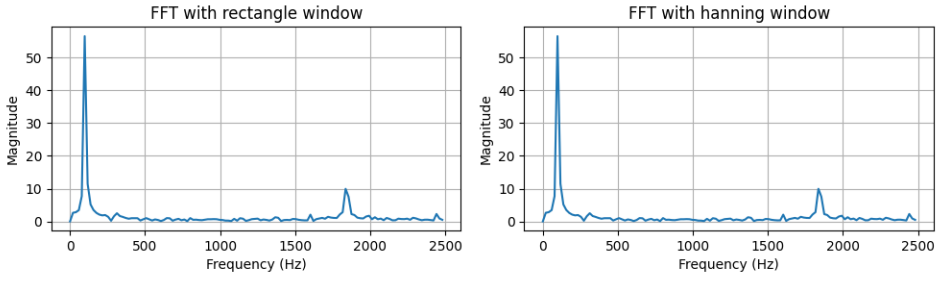

Signal in Time Domain
Amplitude
Time (s)

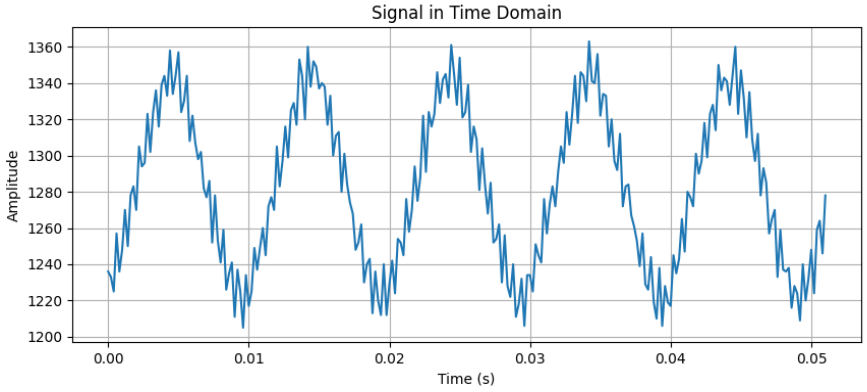
110 Hz


FFT with rectangle window
FFT with hanning window
FFT with hamming window
FFT with blackman window
Magnitude
Frequency (Hz)

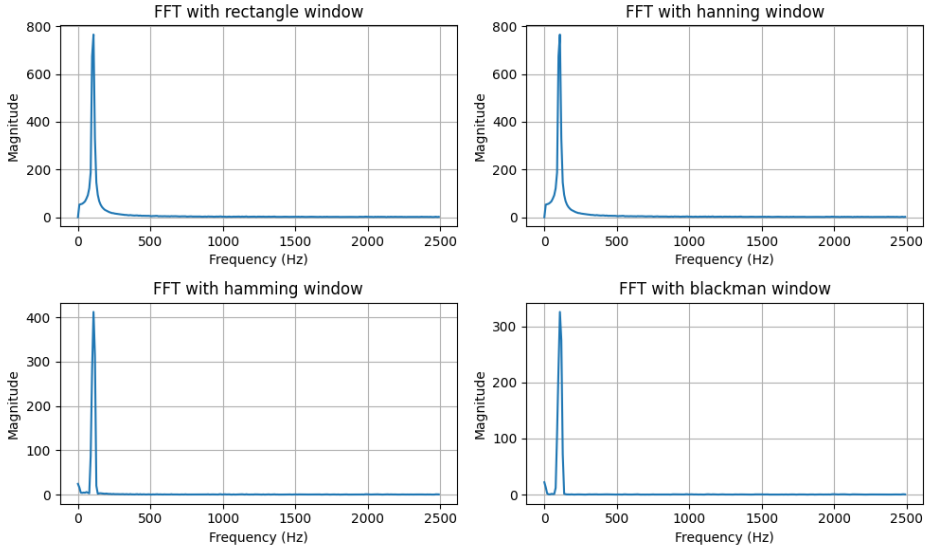

Signal in Time Domain
Amplitude
Time (s)

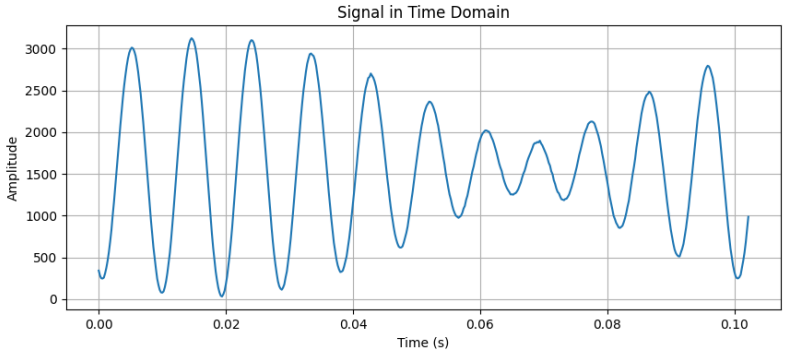
- **Experimento con ruido:**



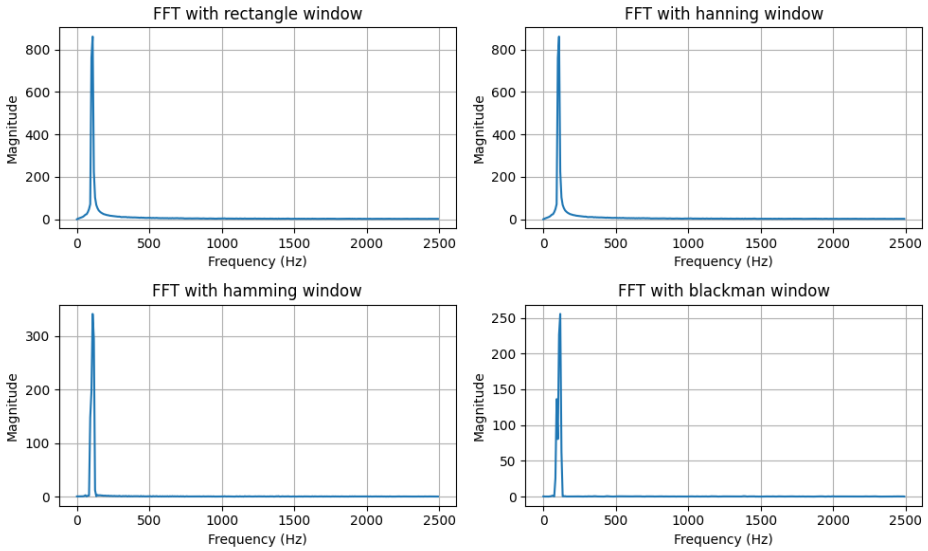


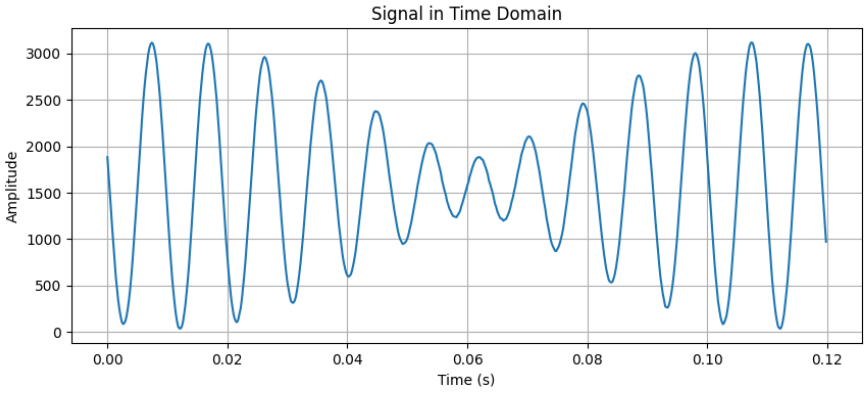
Buffer 512



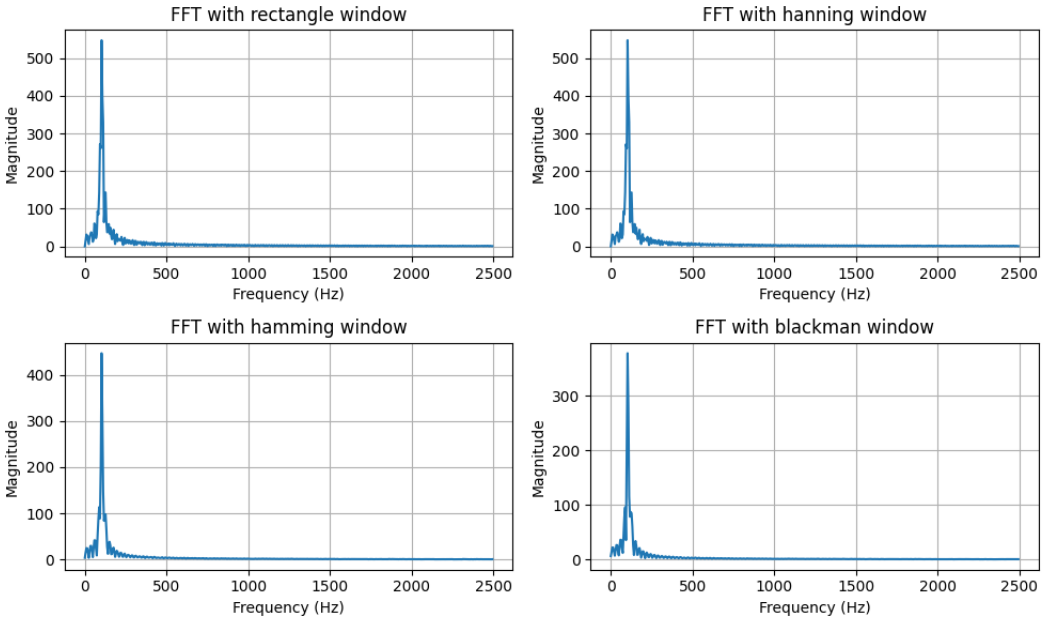

Signal in Time Domain
Amplitude
Time (s)

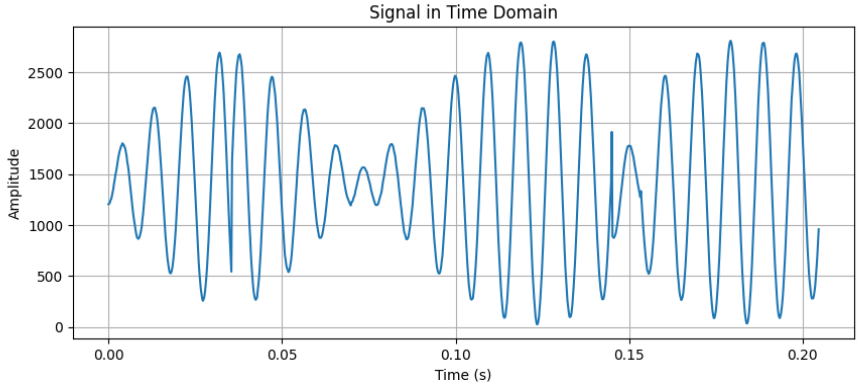
Buffer 600

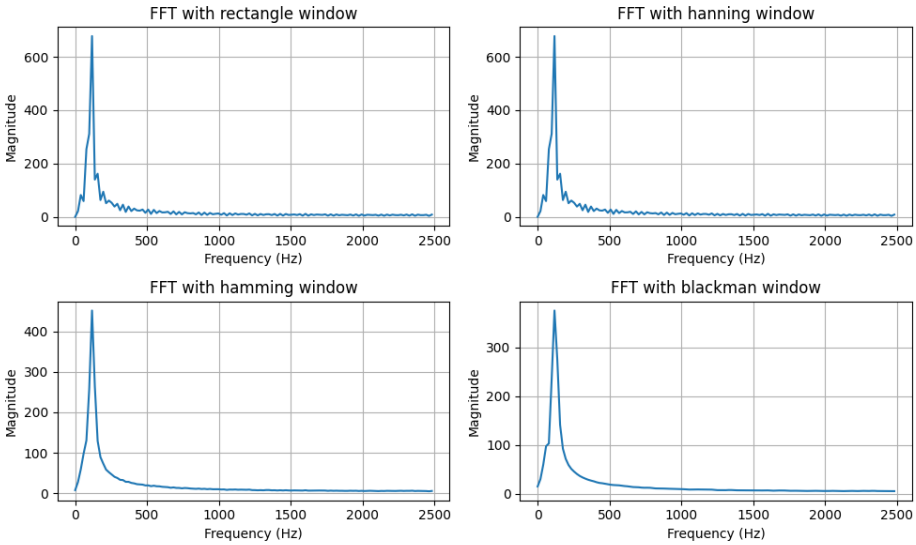

FFT with rectangle window
FFT with hanning window
FFT with hamming window
FFT with blackman window
Magnitude
Frequency (Hz)

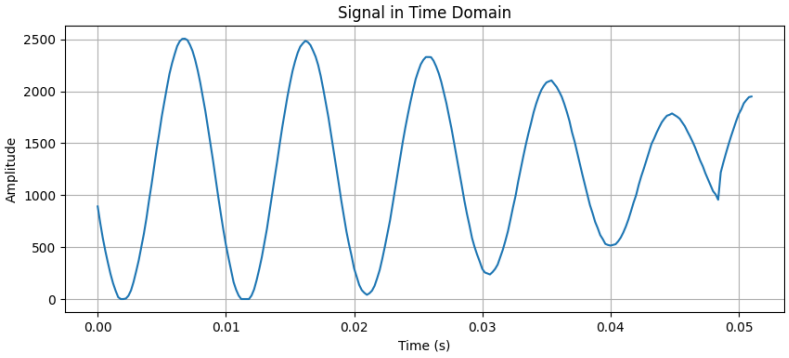

Signal in Time Domain
Amplitude
Time (s)

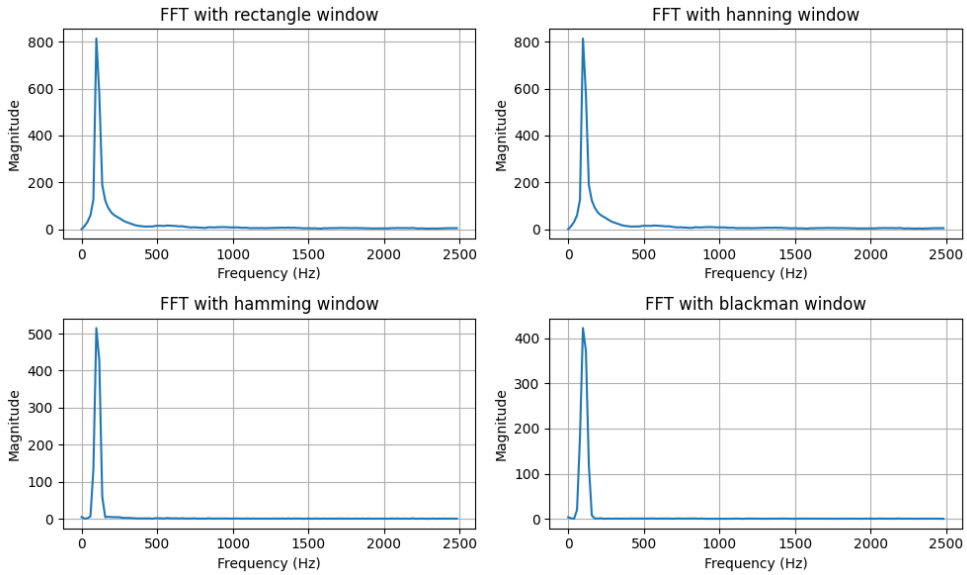
Resultados de espectro:

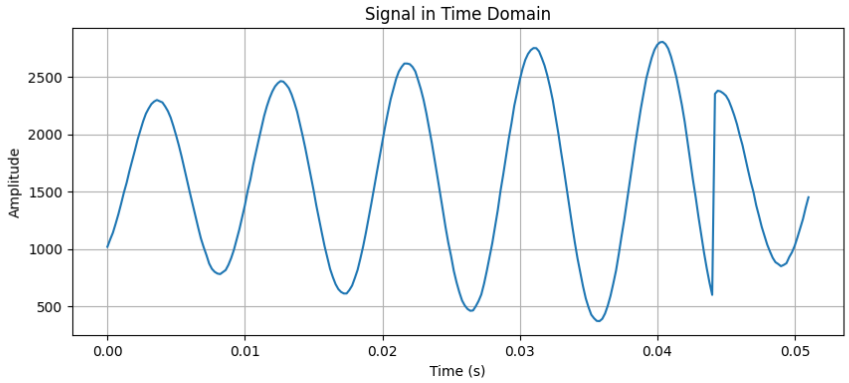












Análisis de resultados:

1. Comparación entre ventanas:

- ¿Qué ventana ofrece la mejor resolución espectral?

El rectángulo no afecta en la resolución. Hanning, Hamming, y en Blackman suprimen más el ruido espectral, en ese orden ascendente (baja a alta frecuencias), sacrificando la integridad de las componentes de interés, pues también atenúan todo el espectro.

- ¿Qué ventana tiene mejor supresión de lóbulos secundarios?

La ventana Black Man ya que su prioridad es minimizar la fuga espectral sin sacrificar resolución.

- ¿Cuál es el compromiso entre resolución y fuga espectral?

Dependiendo que, a mayor resolución espectral, tendrán mayor posibilidad de no tener fuga espectral, e igualmente en viceversa. Un aspecto para considerar también para mejorar la resolución es que se necesitan ventanas más grandes.

2. Análisis del efecto del tamaño de la muestra (N):

- ¿Cómo afecta el tamaño del buffer a la resolución espectral?

El tamaño del buffer nos pondrá la condición para capturar las frecuencias, entre más pequeño sea el buffer menor frecuencias capturaremos (menor resolución espectral) y entre más grande se capturarán mayor número de frecuencias.

- ¿Existe un punto óptimo entre el tiempo de cómputo y calidad?

Sí.

3. Evaluación de capacidad de detección:

- ¿Qué ventana funciona mejor para detectar señales débiles en presencia de ruido?

La ventana Black Man.

- ¿Qué ventana es óptima para discriminar frecuencias cercanas?

La ventana Hamming.

4. Aplicación del teorema de Nyquist:

Verifique si la frecuencia de muestreo cumple con el teorema de Nyquist. Ts = 200us -> Fs = 5Khz Fnysquist >= 2*Fo 2*100hz = 200Hz <= Fs = 5Khz

- Relacione con el fenómeno de aliasing estudiado en prácticas previas.

No solo es importante la frecuencia de muestreo, sino también la cantidad de muestras que debemos tomar, mínimo un periodo de la componente más pequeña de la señal a medir.

Preguntas de compresión:

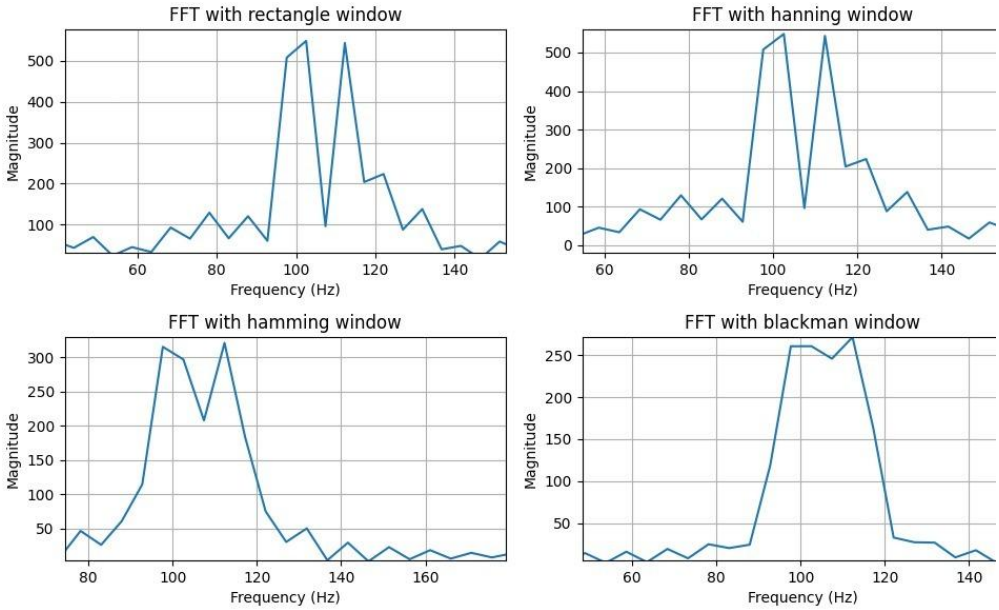
1. ¿Qué es la transformada discreta de Fourier y por qué es importante en DSP?

La transformada discreta de Fourier es una forma de representar que una señal cambia con respecto al tiempo y que se convierte este dominio de tiempo en un dominio de frecuencia, básicamente nos permite ver cambios o alteraciones que no son tan fáciles de detectar a simple vista. La DFT es importante en DSP ya que permite filtrar los ruidos, comprimir datos, reconocer patrones o inclusive recrear o sintetizar nuevas señales.

2. Explique el fenómeno de “fuga espectral” y como las ventanas de ponderación ayudan a mitigarlo.

El término de ‘fuga espectral’ se utiliza cuando una señal no encaja del todo en un intervalo de análisis, ya que en una señal se pueden generar ruidos o perturbaciones, así que se recurre a utilizar ventanas de ponderación que es una técnica que nos permite suavizar estas señales (principalmente los bordes) y así evitar tener un espectro sucio o con malformaciones que afectan la calidad de la señal.

3. Compare y contraste las ventanas Rectangular, Hanning, Hamming y Blackman en términos de:
    - Resolución espectral
    - Supresión de lóbulos secundarios
    - Aplicaciones ideales para cada una



4. ¿Cómo se relaciona la resolución en frecuencia con el número de muestras y la frecuencia de muestreo?

Aquí la más optima es la siguiente, ya que si tenemos mayor cantidad de muestras el resultado será que tendremos una mejor resolución en frecuencia, por otro lado, entre mayor sea la frecuencia de muestreo en la misma cantidad de número de muestras el resultado será a igual a una peor resolución.

5. ¿Por qué el espectro de una señal discreta es periódico?

El espectro en una señal discreta siempre será periódico ya que siempre se muestrea en el tiempo en el que se genera la repetición en el dominio de la frecuencia.