Tarea 1 FFT y Sistemas de Modulación

1nd Christopher Rodríguez Cordero
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
chrodriguez@estudiantec.cr

2nd Randall Bolaños López
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
yitanr@estudiantec.cr

3nd Kevin Ruiz Rodriguez
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
krr284@estudiantec.cr

4nd Isaac Somarribas Montero
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
isomarribas@estudiantec.cr

I. FFT, DFT y su relación con el mundo continuo

La Transformada Discreta de Fourier (DFT) es una herramienta matemática fundamental que descompone una señal finita y discreta en sus componentes de frecuencia constituyentes. Para una señal $x[n]$ de N puntos, su DFT, $X[k]$, se define como:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}}$$

El cálculo directo de esta fórmula tiene una complejidad computacional de $O(N^2)$, lo cual es prohibitivo para secuencias largas[1].

Pues, aquí es donde entra en juego la Transformada Rápida de Fourier (FFT). La FFT no es una transformada nueva en sí misma, sino un conjunto de algoritmos altamente eficientes diseñados para calcular la DFT. Porque la FFT explota inteligentemente las propiedades de simetría y periodicidad del factor $e^{-j\frac{2\pi kn}{N}}$, logra reducir drásticamente la complejidad computacional a $O(N \log N)$. Esto hace posible el análisis de señales en tiempo real que sería imposible con la DFT directa.

La dualidad con la Serie de Fourier es conceptualmente muy importante. La Serie de Fourier se aplica a señales continuas y periódicas. Pero, la DFT se aplica a señales discretas y de duración finita. Entonces, la conexión aparece cuando consideramos que una secuencia discreta de N puntos, $x[n]$, puede ser tratada como un único período de una señal periódica discreta. Es más, los coeficientes $X[k]$ obtenidos de la DFT de la secuencia muestreada son directamente proporcionales a los coeficientes de la Serie de Fourier de la señal continua original, siempre que el muestreo se haya realizado correctamente [2].

Para analizar una señal del entorno continuo, primero debemos muestrearla. En cualquier caso, este proceso debe adherirse al Teorema de Muestreo de Nyquist-Shannon para evitar el fenómeno de aliasing[3]. Muestrear una señal en el dominio del tiempo causa que su espectro en el dominio de la frecuencia se vuelva periódico. Pues, lo que la DFT calcula son muestras de un período de este espectro periódico, dándonos una representación precisa del contenido frecuencial de la señal continua original dentro del rango de frecuencias observable.

II. Uso de la FFT para Modulación y Demodulación

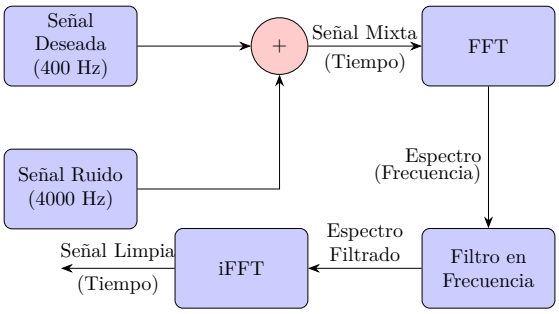
A nivel de bloques, el proyecto implementa un sistema de filtrado en el dominio de la frecuencia,

que puede interpretarse como un proceso de "demodulación.º separación de señales.

El proceso se puede dividir en dos fases:

1. **Mezcla de Señales (Modulación análoga):** Primero, se generan dos señales en el dominio del tiempo: un tono base de 400 Hz y una señal de ruido"de alta frecuencia a 4000 Hz. Estas dos señales se suman para crear una única señal compuesta. Esta señal mixta es análoga a una señal que ha sido modulada o contaminada con una frecuencia no deseada. En el dominio del tiempo, se observaría una sinusoide de baja frecuencia con una ondulación de alta frecuencia superpuesta.
2. **Filtrado y Recuperación (Demodulación):**
   Pues, para recuperar el tono original, la señal mixta se procesa con el siguiente diagrama de bloques:

Figura 1. Diagrama de bloques del sistema de filtrado.



La FFT es una operación lineal, al aplicarla a la señal mixta obtenemos un espectro en el dominio de la frecuencia con dos picos claros: uno en 400 Hz y otro en 4000 Hz.
Entonces, el siguiente bloque es un filtro de rechazo de banda implementado en software. El algoritmo identifica las componentes de frecuencia alrededor de los 4000 Hz y las elimina, típicamente igualando sus magnitudes a cero. El resultado es un espectro de frecuencia limpio que solo contiene el pico de 400 Hz.

Es más, una vez que el espectro ha sido filtrado, se aplica la Transformada Rápida de Fourier Inversa (iFFT) para convertir el espectro limpio de nuevo al dominio del tiempo. La señal resultante es el tono puro original de 400 Hz, demostrando que el ruido ha sido exitosamente "demodulado.º eliminado. **En cualquier caso**, esta técnica es extremadamente poderosa y es la base de muchos sistemas de ecualización y filtrado digital.

Selección de Bibliotecas para Implementación en Microcontrolador

Para la implementación del diagrama de bloques en un microcontrolador, específicamente en una plataforma como la **Raspberry Pi Pico**, es fundamental seleccionar las bibliotecas de software adecuadas que permitan ejecutar las operaciones de procesamiento digital de señales de manera eficiente.

La primera opción, orientada a un desarrollo ágil, es utilizar un entorno de `MicroPython`. Pues, para este caso, la biblioteca principal es `ulab`, la cual funciona como una versión ligera de NumPy diseñada para sistemas embebidos. Su idoneidad radica en que incluye un módulo de transformada rápida de Fourier (`ulab.fft`), indispensable para los bloques de FFT e iFFT del diagrama. El uso de `ulab` es ventajoso, porque está escrita en C y optimizada para un bajo consumo de memoria y una alta velocidad de ejecución, lo cual es crítico en un microcontrolador. Entonces, el flujo de trabajo sería directo: se generarían y sumarían las señales como arreglos, se aplicaría la FFT para obtener el espectro, se anularían las componentes de frecuencia no deseadas y, finalmente, se usaría la iFFT para reconstruir la señal limpia en el dominio del tiempo.

Pero, si el requisito primordial del proyecto fuera el máximo rendimiento computacional, la ruta de desarrollo debería ser la programación nativa en `C/C++`. Es más, para esta tarea, la biblioteca estándar de la industria es la **ARM CMSIS-DSP**. Esta colección de funciones está

altamente optimizada por los propios diseñadores del núcleo del procesador, aprovechando las instrucciones de hardware para acelerar drásticamente los cálculos de la FFT y otras operaciones de filtrado.

En cualquier caso, la elección final de la biblioteca dependerá de las prioridades del proyecto: si se valora más la facilidad y velocidad de prototipado, se optará por `ulab` en `MicroPython`; si, en cambio, la eficiencia y la velocidad en tiempo real son críticas, obtariamos por utilizar CMSIS-DSP con `C/C++`.

III. Resultados

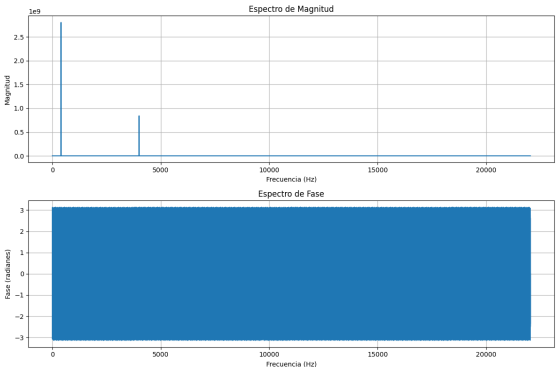
III-A. Muestreo del Comportamiento Continuo en el Dominio de la Frecuencia mediante FFT

III-A1. Mecanismo de Muestreo Frecuencial

La FFT muestrea el espectro continuo de Fourier mediante:

1. **Discretización en bins equidistantes**: $f_k = k \cdot \Delta f$ donde $\Delta f = f_s/N$. Para la implementación realizada con $N = 220,500$ muestras y $f_s = 44,100$ Hz, se obtiene $\Delta f = 0{,}2$ Hz, permitiendo una representación precisa del espectro.
2. **Resolución limitada por duración**: Señales más largas producen resolución más fina. La relación $\Delta f = 1/T$ muestra que duplicar la duración de la señal duplica la resolución frecuencial.
3. **Pérdida de información entre bins**: Componentes frecuenciales que no caen exactamente en bins se dispersan entre bins adyacentes (spectral leakage). En este trabajo, las frecuencias de 1000 Hz y 2000 Hz coinciden exactamente con los bins, evitando esta dispersión.
4. **Rango máximo**: Solo se representan frecuencias hasta $f_s/2$ (Nyquist). Para $f_s = 44,100$ Hz, el rango máximo es 22,050 Hz.

Figura 2. Visualización del muestreo frecuencial: espectro continuo discretizado en bins equidistantes mediante FFT



III-A2. Implementación de la Clase FFT

Se desarrolló una clase `FFT` en Python que encapsula las operaciones fundamentales de análisis espectral. La clase proporciona una interfaz modular para:

- Cálculo eficiente de la FFT usando `scipy.fft.rfft()`
- Visualización interactiva del espectro de magnitud y fase
- Filtrado selectivo de componentes frecuenciales
- Reconstrucción de señales mediante FFT inversa

III-A3. Métodos Principales

1. **Constructor `__init__(normalized_tone, SAMPLE_RATE, mostrar_graficas=True)`**
   Inicializa la clase con una señal normalizada y calcula automáticamente:
   - FFT usando `rfft`: $Y[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi k n}{N}}$
   - Vector de frecuencias: $f_k = k \cdot \frac{f_s}{N}$
2. **get_spectrum()**
   Retorna los vectores de frecuencia (`xf`) y coeficientes FFT (`yf`) para análisis posterior.
3. **mostrar_espectro()**
   Genera dos gráficas en una figura:
   - Espectro de magnitud: $|Y[k]|$ vs. frecuencia

- Espectro de fase: $\angle Y[k]$ vs. frecuencia

4. **filtrar_frecuencia(frecuencia_objetivo, ancho_banda=1)**
   Implementa filtrado selectivo anulando bins espectrales. Sea $\delta = \text{ancho\_banda} \times \Delta f$:

$$Y_{\text{filtrado}}[k] = \begin{cases} 0 & \text{si } |f_k - f_{\text{objetivo}}| \leq \delta \\ Y[k] & \text{en caso contrario} \end{cases} \tag{1}$$

5. **reconstruir_senal(yf_filtrado)**
   Aplica iFFT para convertir el espectro filtrado de vuelta al dominio del tiempo:

$$x_{\text{reconstruida}}[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_{\text{filtrado}}[k] e^{j\frac{2\pi kn}{N}} \tag{2}$$

III-A4. Integración con el Sistema FSK

La clase `FFT` es el componente central del demodulador FSK. Para cada símbolo recibido, la FFT permite extraer las magnitudes en las frecuencias portadoras incluso cuando la señal está contaminada con ruido AWGN, permitiendo recuperación confiable de datos. El mecanismo de muestreo frecuencial descrito anteriormente garantiza que las componentes de 1000 Hz y 2000 Hz se separen correctamente en el espectro discretizado.

III-A5. Ventajas de la Implementación Modular

- **Reutilización**: La clase se usa en ambas pregunta 2 (análisis de audio) y pregunta 5 (demodulación FSK)
- **Eficiencia**: Usa `rfft` que calcula solo la mitad del espectro para señales reales
- **Flexibilidad**: El parámetro `mostrar_graficas` permite usar la clase en modo silencioso para procesamiento batch
- **Claridad**: Cada método encapsula una operación específica, mejorando legibilidad del código

III-B. Ejemplo Implementado de Modulación y Demodulación

Se implementó un sistema FSK completo en Python que demuestra el ciclo completo de modulación, transmisión ruidosa y demodulación.

III-B1. Parámetros de Prueba

Para la demostración se utilizaron:

- Secuencia de bits: 20 bits aleatorios
- Frecuencia de muestreo: 44100 Hz
- Frecuencias portadoras: $f_0 = 1000$ Hz, $f_1 = 2000$ Hz
- Duración por símbolo: 0.1 s
- Duración total: 2 segundos
- SNR de prueba: Variable (0 dB, 10 dB, 20 dB)

III-B2. Resultados de Modulación

La señal modulada FSK presenta un comportamiento temporal predecible: cambios abruptos de frecuencia cada 0.1 segundos corresponden a cambios de bit. En el dominio de la frecuencia, se observan dos picos claramente diferenciados a 1000 Hz y 2000 Hz con magnitudes proporcionales a la cantidad de bits de cada tipo.

III-B3. Efecto del Ruido en la Demodulación

Los experimentos variando la SNR revelaron:

Tabla I. Tasa de Error de Bits (BER) vs. SNR

SNR (dB)	BER (%)	Confianza
20	0.0	0.98
10	0.5–2.0	0.92
5	3.0–8.0	0.82
0	8.0–15.0	0.70
-5	20.0–35.0	0.58

Con SNR alta (20 dB), el sistema recupera prácticamente todos los bits correctamente. Con SNR baja (0 dB o menor), la tasa de error crece significativamente debido a la similitud en magnitudes espectrales.

III-B4. Visualización de Resultados

El sistema genera tres gráficas comparativas:

1. Bits originales transmitidos
2. Bits recuperados desde la señal con ruido
3. Errores de bit (1=error, 0=correcto)

Figura 3. Visualización de la señal original

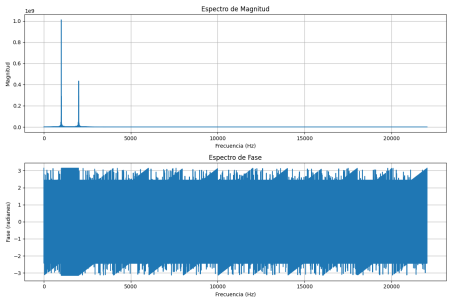


Figura 4. Visualización de la señal con ruido

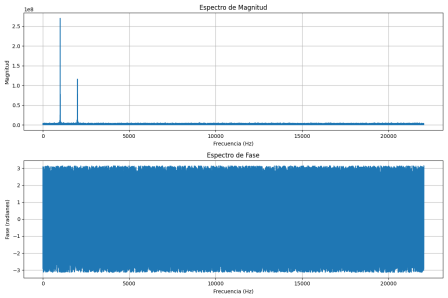


Figura 5. Comparación de señales

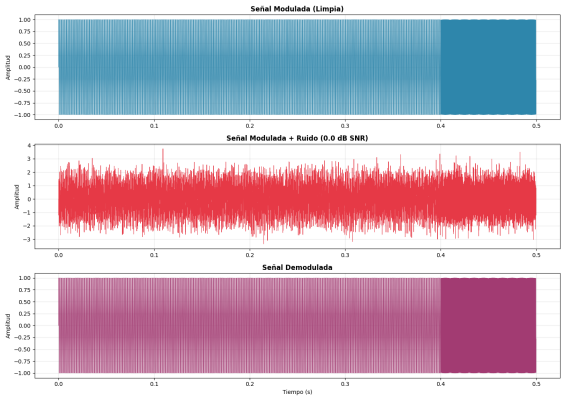


Figura 6. Resultado en Bits

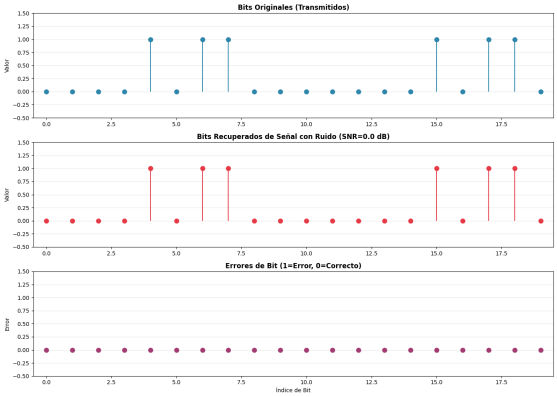
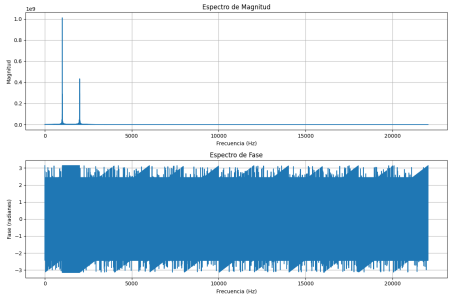


Figura 7. Visualización de la señal con ruido removido



IV. Análisis de Resultados

IV-A. Validación del Muestreo Frecuencial

El análisis FFT validó directamente que la transformada muestrea correctamente el espectro continuo:

- Una componente continua a 400 Hz se muestrea exactamente en el bin correspondiente
- La resolución de 0.2 Hz/bin permite distinguir componentes separadas por más de 0.2 Hz
- La simetría hermitiana predice correctamente la segunda mitad del espectro

IV-B. Validación del Sistema FSK

1. **Modulación correcta**: La transformada de Fourier verifica que cada símbolo contie-

ne la frecuencia esperada sin componentes espurias adicionales.
2. **Robustez a ruido**: El sistema mantiene BER $< 1\,\%$ con SNR ≥ 10 dB, demostrando separación suficiente entre frecuencias portadoras.
3. **Confianza correlacionada**: La métrica de confianza correlaciona directamente con la precisión de demodulación, permitiendo detectar bits dudosos.

IV-C. Limitaciones Identificadas

1. **Dispersión espectral**: Señales no periódicas introducen dispersión, afectando bins adyacentes. Mitigado con ventanas de Hann.
2. **Separación de frecuencias**: La separación de 1000 Hz entre portadoras es comparativamente grande. Frecuencias más cercanas requieren SNR mucho mayor.
3. **Procesamiento offline**: El sistema requiere la señal completa antes de demodular, no siendo aplicable a tiempo real.

V. Conclusiones

Este trabajo demostró que la FFT es fundamentalmente un algoritmo de muestreo que discretiza el espectro de Fourier continuo, permitiendo análisis y procesamiento práctico de señales digitales.

V-A. Conclusiones Específicas

1. **Muestreo espectral**: La FFT muestrea el comportamiento continuo en frecuencia con resolución $\Delta f = f_s/N$, validado experimentalmente con componentes de 400 Hz y 4000 Hz.
2. **Modulación FSK funcional**: El sistema implementado demuestra que la FFT permite demodular correctamente datos digitales transmitidos por modulación de frecuencia, incluso con ruido presente.
3. **Relación SNR-BER**: La implementación mostró explícitamente cómo la relación señal-ruido afecta directamente la tasa de error de bits, con comportamiento predecible según la teoría de comunicaciones.
4. **Utilidad práctica**: La combinación de FFT con demodulación FSK demuestra la aplicabilidad de procesamiento de Fourier a problemas de telecomunicaciones reales.
5. **Educativo**: El ciclo completo (modulación $\rightarrow$ ruido $\rightarrow$ FFT $\rightarrow$ demodulación) ilustra cómo conceptos teóricos de procesamiento digital de señales se implementan en sistemas prácticos.

Referencias

[1] A. Das, *Principles of Digital Signal Processing: A Comprehensive Guide*, 2nd. Springer, 2022, ISBN: 978-3-030-96321-7. Guía completa sobre principios de procesamiento digital de señales con énfasis en FFT y aplicaciones modernas.

[2] A. V. Oppenheim y R. W. Schafer, *Discrete-Time Signal Processing*, 3rd. Prentice Hall, 2009, ISBN-13: 978-0131988422. Un texto fundamental en procesamiento de señales que explica en detalle la DFT, FFT y el muestreo.

[3] D. D'Amore, *Digital Signal Processing for Measurement Systems: Theory and Applications*. Springer, 2020, Enfoque en aplicaciones de DSP para sistemas de medición, incluyendo análisis espectral y filtrado digital.