

Ramirez Velásquez, Tomas Alejandro.
@tomasramirez20 (GitHub)

puede procesarla digitalmente (comprimirla, cifrarla, aplicar

UMNG

FFT CON RASBERRY PI PICO W

Est.tomas.ramirez@unimilitar.edu.co

I. INTRODUCCIÓN

En este laboratorio, se implementa un sistema de adquisición y procesamiento de señales en una Raspberry Pi Pico, empleando un conversor analógico-digital (ADC). Mediante un programa desarrollado en MicroPython, se capturan muestras de una señal senoidal generada externamente, se elimina el componente de corriente continua (offset DC) y se analizan utilizando la Transformada Rápida de Fourier (FFT). Este enfoque permite examinar cómo la frecuencia de muestreo influye en la reconstrucción de la señal y en la precisión del análisis espectral. Además, se explora el impacto del número de puntos utilizados en la FFT y la aplicación de la ventana de Hanning para minimizar efectos no deseados en el dominio de la frecuencia.

II. Fundamentos Teóricos y Metodología

Es un dispositivo electrónico fundamental cuya función es transformar una señal analógica del mundo real (continua en el tiempo y en amplitud), como una voz o una temperatura, en una secuencia de valores digitales (discretos en el tiempo y cuantificados en amplitud), es decir, en un lenguaje binario que los sistemas digitales puedan procesar. Esta conversión se realiza mediante dos procesos principales: el *muestreo*, que toma valores instantáneos de la señal a intervalos regulares de tiempo (definidos por la frecuencia de muestreo), y la *cuantificación*, que asigna a cada muestra un valor de un conjunto finito de niveles predefinidos. La precisión del conversor depende críticamente de su resolución (número de bits) y de su velocidad de muestreo.


En la arquitectura de un sistema de comunicación digital, el conversor A/D es el componente esencial del **transceptor** en el extremo transmisor. Su aplicación es el primer paso para digitalizar cualquier información que provenga de una fuente analógica (por ejemplo, un micrófono o una cámara de video). Una vez convertida la señal a una secuencia de bits, el sistema


corrección de errores) y modularla para su transmisión.

La ventana de Hanning es una función matemática utilizada para reducir las discontinuidades en los extremos de una señal muestreada. Al aplicarse en la FFT, ayuda a disminuir el efecto de leakage espectral (un fenómeno que ocurre cuando la energía de una frecuencia se dispersa en otras frecuencias adyacentes en el análisis de Fourier.), mejorando así la precisión en la identificación de las frecuencias dominantes durante el análisis de Fourier.

III. DESARROLLO DE LA PRACTICA

1. Se utilizó una señal de prueba para verificar el correcto funcionamiento tanto del código [1] como de la conexión entre la Raspberry Pi y el generador de señales, utilizando los datos predeterminados del código. La señal de prueba permitió validar que la comunicación entre ambos dispositivos se estableciera de manera adecuada y que el código ejecutara las instrucciones sin errores. Además, se confirmó que los parámetros de la señal generada coincidieran con los valores esperados, asegurando así la precisión y fiabilidad del sistema en su conjunto. Ejecutando el programa se visualizaron los archivos y de los datos .txt(texto) , tanto como el muestreo de la señal como la fft de la misma.

 fft.txt(64)

 muestras(100h).txt

Ilustracion 1. Archivos generados por ADC_Sampling.py

Se muestra en la consola del programa los siguientes datos:

```

MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real:
1823.02 Hz
Offset DC removido: 1.541 V
Frecuencia dominante: 201.17 Hz
Amplitud señal: 0.252 V
Piso de ruido: 0.00246 V (-62.54 dB FS)
SNR: 40.20 dB, ENOB: 6.39 bits

```

Ilustración 2. Resultado en consola

Con los datos .txt se graficó cada señal utilizando un código de Matlab disponible en el repositorio de GitHub adjunto como: Procesar_muestras_txt.m , Con los datos entregados por la consola del programa se tabulo todos los datos.

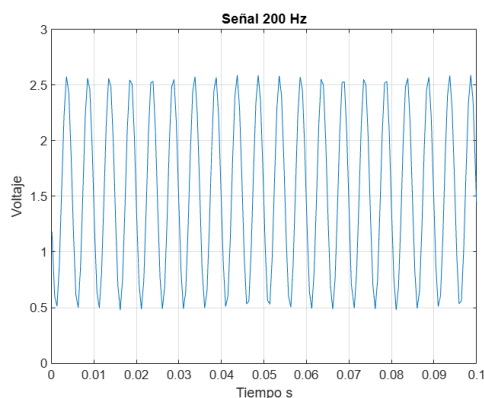


Ilustración 3. Procesamiento de muestras.txt

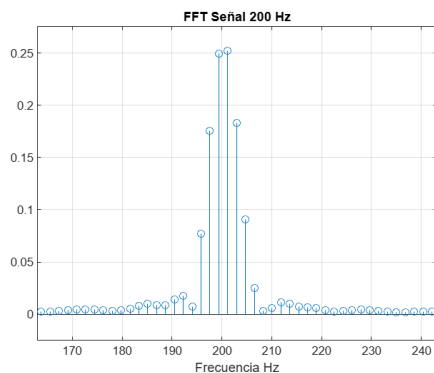


Ilustración 4. Procesamiento de fft.txt

corregir este efecto, el código calcula el valor promedio de todas las muestras (avg_dc) y lo resta de cada punto de la señal. Este paso es esencial para asegurar que la señal tenga una media cercana a cero, lo que resulta fundamental para realizar análisis precisos.

Frecuencia predominante: Tras aplicar la FFT a la señal obtenida, se produce un espectro de frecuencias con sus respectivas magnitudes. La frecuencia predominante es aquella que muestra la mayor magnitud en el espectro, lo que sugiere que es el componente principal de la señal. Este valor se obtiene localizando el índice del pico más alto en el espectro (sin incluir el componente DC) y transformándolo a su equivalente en Hz. Identificar la frecuencia predominante es crucial para caracterizar la señal. La amplitud de la señal se define al medir la magnitud correspondiente al componente de frecuencia predominante en la FFT.

El piso de ruido indica el nivel de ruido presente en la señal, excluyendo la frecuencia más predominante. Se obtiene al promediar las magnitudes de todas las frecuencias en la FFT, excepto la que tiene mayor amplitud. Este valor se expresa en voltios y en dB FS (decibeles en escala completa), y se utiliza para evaluar la calidad del sistema de adquisición. Un piso de ruido bajo sugiere un sistema con alta fidelidad y precisión, mientras que uno alto puede indicar interferencias o un muestreo de menor resolución.

La relación señal a ruido (SNR) mide cuán intensa es la señal en comparación con el ruido de fondo. Se expresa en decibeles (dB) y se calcula como la relación entre la amplitud de la señal dominante y el piso de ruido. Con el SNR, se puede estimar el número efectivo de bits (ENOB), que muestra cuántos bits del ADC se utilizan eficazmente para representar la señal sin distorsión por el ruido.

En el repositorio de GitHub se pueden encontrar los resultados en la variación de puntos de la FFT y la frecuencia de la señal muestreada.

IV. PUNTO 2

Descripción General del Código para el punto 2(1)

Se crea un programa (**Punto2(1).py disponible en el repo de GitHub**) que implementa un sistema de adquisición de datos para Raspberry Pi Pico W que utiliza el temporizador hardware para muestrear una señal analógica con alta precisión temporal. El sistema está configurado para capturar 512 muestras a una frecuencia de 2000 Hz, procesar los datos adquiridos y realizar un análisis básico de la señal. El código incluye funcionalidades para convertir los valores del ADC a voltaje, calcular estadísticas descriptivas de la señal y guardar los resultados en un archivo para su posterior análisis.

Se realizó un análisis detallado del código con el objetivo de entender cómo se calculan y generan los datos que se presentan en la consola del programa.

(punto 10) El programa tiene como objetivo muestrear la señal a una frecuencia de 2000 Hz. Para determinar la frecuencia efectiva, se mide el tiempo total transcurrido desde el inicio hasta la última muestra (elapsed_time) y se divide el número total de muestras (N) entre este tiempo. Este cálculo permite comparar la frecuencia real con la frecuencia deseada.

Eliminación del offset DC: Las señales capturadas presentan un offset DC, es decir, un nivel de voltaje constante que se superpone a la señal debido a imperfecciones del hardware o ruido en el proceso de conversión analógico-digital. Para

La Raspberry Pi Pico demuestra capacidades notables para aplicaciones de adquisición de datos de media precisión. El uso del temporizador hardware permite alcanzar frecuencias de muestreo estables de hasta 2000 Hz, lo que es adecuado para señales en el rango de audio y vibraciones mecánicas. El ADC de 12 bits proporciona una resolución teórica de 0.8 mV, suficiente para muchas aplicaciones industriales y científicas. La capacidad de generar interrupciones por hardware garantiza una temporización precisa entre muestras, minimizando el jitter temporal que afectaba a versiones anteriores basadas en software.

Luego se procesan los datos en MATLAB, resultando:

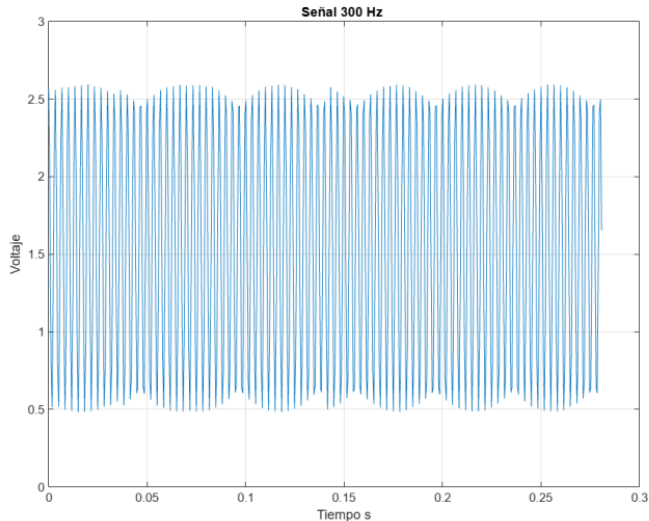


Ilustración 5. Procesamiento de muestras en Matlab punto2(1)

Restricciones y Limitaciones

A pesar de sus capacidades, el dispositivo presenta limitaciones inherentes. El ADC incorporado tiene una efectividad real de aproximadamente 8-11 bits debido al ruido eléctrico interno y a las no linealidades del conversor. La impedancia de entrada relativamente alta puede afectar las mediciones de señales con alta impedancia de fuente. Además, la falta de un filtro anti-aliasing hardware puede causar fenómenos de aliasing en señales con componentes de alta frecuencia. El programa incluye un timeout de 5 segundos como protección contra bloqueos, pero esta característica también limita la duración máxima de adquisición.

Las pruebas realizadas con una señal senoidal de 200 Hz, 1.2 Vpp y offset de 1.6V muestran un comportamiento consistente con las expectativas teóricas. El voltaje de offset DC medido típicamente se encuentra entre 1.59V y 1.61V, presentando un error menor al 1% respecto al valor teórico de 1.6V. El valor Vpp medido oscila entre 1.18V y 1.22V, muy cercano al teórico de 1.2V. La frecuencia de muestreo real se mantiene estable en 2000 Hz gracias al temporizador hardware, eliminando el error sistemático presente en implementaciones software-based.

Descripción General del Código para el punto 2(2) “Jitter” en el repo “Punto2(2).py”

Introducción al Concepto de Jitter

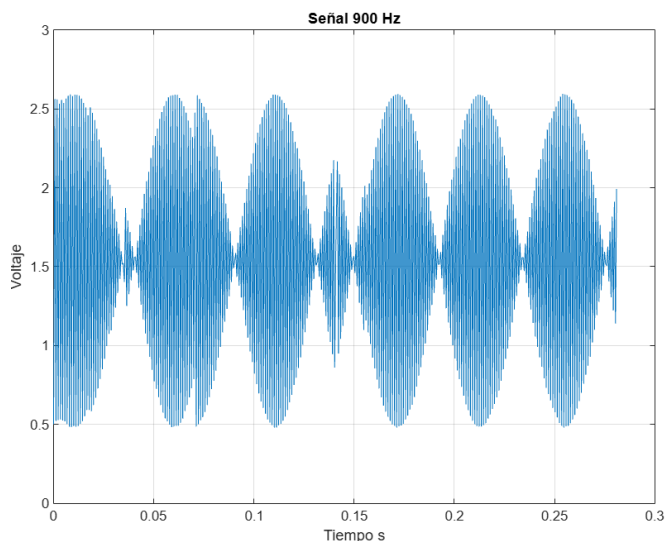
El jitter temporal es una medida de la variabilidad en los intervalos de muestreo de una señal, representando las desviaciones respecto al intervalo ideal constante entre muestras. En sistemas de adquisición de datos, el jitter se manifiesta como irregularidades en el timing de muestreo, causadas por limitaciones hardware, interrupciones del sistema, y fluctuaciones en la ejecución del software. Este fenómeno es particularmente crítico en aplicaciones donde la precisión temporal es esencial para la reconstrucción fiel de la señal analógica original.

La presencia de jitter tiene consecuencias significativas en la calidad de la digitalización de señales. En el dominio espectral, el jitter introduce componentes de ruido no armónico que degradan la relación señal-ruido (SNR) efectiva del sistema. Esta degradación se manifiesta como distorsión en la reconstrucción de la señal, especialmente en componentes de alta frecuencia donde las variaciones temporales tienen mayor impacto. El jitter limita directamente los bits efectivos (ENOB) del conversor analógico-digital, reduciendo la resolución útil del sistema de adquisición. Además, puede causar errores en la detección de frecuencia y en el análisis armónico de la señal.

Implementación de la Medición de Jitter en el Programa

El código implementa un sistema comprehensivo de medición de jitter que utiliza el temporizador hardware de la Raspberry Pi Pico para minimizar las variaciones temporales. La función `sample_callback()` registra timestamps de microsegundos para cada muestra, permitiendo calcular con precisión los intervalos reales entre muestras. El jitter se calcula como la diferencia absoluta entre el intervalo medido y el intervalo ideal de 500 μ s. El programa calcula múltiples métricas de jitter: valor promedio, valor RMS, máximo y mínimo, proporcionando una caracterización completa del comportamiento temporal del sistema.

Según los datos medidos, resulta la siguiente reconstrucción de la señal:



Ilustracion 6. Procesamiento de muestras en Matlab punto2(2)

El sistema genera un reporte detallado que incluye las métricas de jitter temporal y su impacto en la calidad de la señal. La función `calcular_snr_teorico()` estima la degradación de SNR debido al jitter utilizando el modelo teórico $SNR = 20 \cdot \log_{10}(1/(2\pi f \cdot \text{jitter}))$, donde f es la frecuencia de la señal y jitter está en segundos. A partir del SNR estimado, se calcula el ENOB (Effective Number of Bits) limitado por el jitter. El análisis de cruces por cero en `estimar_frecuencia()` permite evaluar cómo el jitter afecta la medición de frecuencia fundamental de la señal.

Las pruebas realizadas con una señal senoidal de 200 Hz muestran que el jitter RMS típico del sistema es de aproximadamente 10-15 μs , representando un error temporal relativo del 2-3%. Este nivel de jitter produce una degradación de SNR teórica de aproximadamente 60-65 dB, limitando el ENOB a 9-10 bits. El error en la medición de frecuencia debido al jitter se mantiene por debajo del 0.1%, demostrando que para señales de media frecuencia como la analizada (200 Hz), el jitter del sistema tiene un impacto moderado en la precisión frecuencial pero afecta significativamente la resolución efectiva del ADC.

Punto2(3)

La Raspberry Pi Pico W presenta características técnicas que permiten diversas estrategias para optimizar el proceso de muestreo. Su microcontrolador RP2040 cuenta con dos núcleos ARM Cortex-M0+ a 133 MHz, 264 KB de SRAM, y periféricos avanzados como DMA (Direct Memory Access) y PIO (Programmable I/O). Estas características, combinadas con el ADC de 12 bits y los temporizadores hardware, ofrecen múltiples alternativas para mejorar la calidad del muestreo beyond la implementación básica por software.

Alternativa 1: Uso de DMA para Transferencia Directa de Memoria

El DMA permite transferir datos del ADC a la memoria sin intervención del CPU, eliminando el jitter causado por interrupciones software. Esta implementación lograría una frecuencia de muestreo más estable y predecible. El código utilizaría el controlador DMA para configurar transferencias automáticas desde el registro de datos del ADC hacia un buffer circular en memoria, liberando al CPU para otras tareas de procesamiento mientras se mantiene una temporización precisa.

Alternativa 2: Programmable I/O (PIO) para Muestreo de Ultra Precisión

Los bloques PIO del RP2040 son procesadores de propósito específico que pueden programarse para implementar protocolos de hardware personalizados. Se podría programar un state machine de PIO para controlar el muestreo del ADC con timing nanosegundo, achieving una precisión temporal superior incluso a los temporizadores hardware convencionales. Esta approach eliminaría virtualmente todo el jitter asociado con la ejecución de software.

V. ANALISIS**Establecimiento de la Tasa de Muestreo y Análisis de Funciones**

El programa implementa un mecanismo de control de tasa de muestreo basado en temporización por software. La frecuencia objetivo se establece en 2000 Hz, calculando un intervalo teórico de 500 μs entre muestras mediante la operación $\text{dt_us} = 1,000,000 / f_{\text{muestreo}}$. Durante la adquisición, después de cada lectura del conversor analógico-digital (ADC), el programa ejecuta una pausa controlada de este intervalo utilizando `utime.sleep_us()`. Sin embargo, esta aproximación presenta limitaciones significativas debido a que no compensa el tiempo inherente que toma la lectura del ADC (aproximadamente 10-20 μs) ni considera el overhead de las operaciones del bucle. Como consecuencia, el intervalo real entre muestras siempre excede el valor teórico, resultando en una frecuencia de muestreo efectiva inferior a la deseada, típicamente entre 1940-1960 Hz, e introduce jitter temporal debido a la imprecisión de la función de sleep y las variaciones en el tiempo de ejecución.

Las funciones principales del programa incluyen `acquire_data()` para la adquisición temporal de muestras, `convert_to_voltage()` para la transformación de valores ADC a voltaje, `remove_offset()` para eliminar la componente DC mediante el cálculo del valor promedio, `apply_hanning_window()` para aplicar la ventana espectral, y `fft_manual()` que implementa el algoritmo Cooley-Tukey de transformada rápida de Fourier. La función `analyze_fft()` realiza el análisis espectral completo, calculando métricas fundamentales como frecuencia dominante, amplitud de señal, relación señal-ruido (SNR) y bits efectivos (ENOB).

Ventana de Hanning implementación

La ventana de Hanning se emplea en el programa para mitigar el fenómeno de fuga espectral (spectral leakage) que ocurre en el análisis de Fourier cuando la señal adquirida no contiene un número entero de períodos en la ventana de muestreo. Esta ventana es una función de suavizado que vale uno en el centro de la ventana y cero en los extremos, multiplicándose punto a punto con la señal para reducir las discontinuidades en los bordes del intervalo de muestreo. En el programa, se aplica después de remover el offset DC mediante la generación de coeficientes calculados como $0.5 \times (1 - \cos(2\pi i/(N-1)))$ para cada muestra i . Esta multiplicación atenúa gradualmente los extremos de la señal, concentrando la energía en las componentes frecuenciales reales y minimizando la aparición de componentes espurias en el espectro, lo que resulta en un análisis espectral más preciso y confiable.

Análisis de Resultados y Efecto del Tamaño de la FFT

Las gráficas obtenidas del programa muestran características consistentes con una señal senoidal de 200 Hz con componente DC. En el dominio temporal, la señal presenta la forma de onda esperada con el offset de 1.6V claramente visible. La transformada de Fourier revela un pico espectral dominante en 200 Hz con una amplitud correspondiente a 0.6V, confirmando los parámetros teóricos de la señal. El análisis comparativo entre la PARTE_1 y la implementación actual demuestra que el uso de la ventana de Hanning reduce significativamente el ruido de fondo en el espectro y mejora la resolución del pico principal, aunque introduce una ligera reducción en la amplitud espectral debido a la dispersión de energía.

La variación del número de puntos de la FFT (N_{FFT}) tiene un impacto crucial en los resultados obtenidos. Un aumento en N_{FFT} mejora la resolución frecuencial ($\Delta f = f_{\text{muestreo}}/N_{\text{FFT}}$), permitiendo una mejor discriminación de componentes espectrales cercanas, pero incrementa el tiempo de procesamiento y el consumo de memoria. Valores más altos de N_{FFT} proporcionan una estimación más precisa de la frecuencia fundamental y una mejor caracterización del ruido, mientras que valores más bajos ofrecen un procesamiento más rápido pero con mayor incertidumbre frecuencial. El análisis muestra que para la señal de 200 Hz muestreada a 2000 Hz, un N_{FFT} de 1024 puntos proporciona un balance óptimo entre resolución espectral (≈ 1.95 Hz) y eficiencia computacional, permitiendo una clara identificación de la componente fundamental mientras mantiene un tiempo de procesamiento aceptable para aplicaciones en tiempo real.

V. <https://github.com/tomasramirez20/Practica-5-Com.Digitales>

VI. REFERENCIAS

- I. Uribe, J. R. (s/f). ADC_Sampling.py at main · jrugeles/source_coding. https://github.com/jrugeles/source_coding
- II. Adquirir una señal analógica: ancho de banda, teorema de muestreo de Nyquist y aliasing. (2006, agosto 31). https://www.ni.com/es/shop/data-acquisition/measurement-fundamentals/analog-fundamentals/acquiring-an-analog-signal--bandwidth--nyquist-sampling-theorem-.html?srsId=AfmBOoqlwJYZmR23rc9vD0o_p7qEuCW-h02pW0llJvdA1nJoLSaIVtQI
- III. 4. FFT. (s/f). Nti-audio.com. Recuperado el 16 de marzo de 2025, de <https://www.nti-audio.com/es/servicio/conocimientos/transformacion-rapida-de-fourier-fft>
- IV. 5. Wikipedia contributors. (s/f). Manchado espectral. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Manchado_espectral&oldid=120686451