Exploración de conceptos de conversión A/D con Raspberry Pi Pico

Luis Carlos Leal Gamboa
est.luis.cleal@unimilitar.edu.co
Docente: José de Jesús Rúgeles

***Resumen*—En este laboratorio se exploró el funcionamiento del conversor análogo-digital (ADC) del microcontrolador Raspberry Pi Pico 2W. A partir de mediciones prácticas de diferentes voltajes DC y señales senoidales, se analizaron las diferencias entre los valores teóricos y experimentales mediante la media, la desviación estándar y el porcentaje de error. Se implementaron programas en MicroPython y MATLAB para la adquisición, procesamiento y representación de datos. Finalmente, se graficaron histogramas y se compararon las señales teóricas y experimentales, evidenciando las limitaciones del ADC frente a factores como ruido, precisión de la fuente y tasa de muestreo.**

***Abstract*— This laboratory explored the operation of the analog-to-digital converter (ADC) of the Raspberry Pi Pico 2W microcontroller. Practical measurements of different DC voltages and sinusoidal signals were carried out, analyzing the differences between theoretical and experimental values through mean, standard deviation, and error percentage. Programs in MicroPython and MATLAB were implemented for data acquisition, processing, and visualization. Finally, histograms were plotted, and theoretical versus experimental signals were compared, highlighting the limitations of the ADC due to factors such as noise, power source precision, and sampling rate.**

I. Introducción

La conversión análogo-digital es un proceso fundamental en los sistemas embebidos y de adquisición de datos, ya que permite que las señales continuas del mundo real sean interpretadas por dispositivos digitales. El Raspberry Pi Pico 2W incorpora conversores ADC de 12 bits que, aunque poseen limitaciones en precisión y velocidad, resultan útiles para aplicaciones académicas y de prototipado.
En este laboratorio se buscó comprender el funcionamiento del ADC del Pico 2W, relacionando las mediciones prácticas con los fundamentos teóricos de la cuantización, la discretización temporal y el muestreo de señales. Además, se emplearon herramientas computacionales como MicroPython y MATLAB para procesar los datos, calcular parámetros estadísticos y graficar los resultados, con el fin de evaluar la confiabilidad y las limitaciones del ADC en diferentes escenarios.

II. Uso del conversor A/D

El rasperry pi pico 2W posee diferentes pines que corresponden a los conversores A/D en este caso, se utilizara el establecido en el GP26, el cual corresponde al ADC0, así mismo, cuenta con el voltaje de referencia máximo el cual corresponde a 3,3V aproximadamente, en este caso el pin 36 cuenta con la salida de este, para corroborar esto, se utiliza el multímetro para comprobar el valor exacto de voltaje presente.

Ilustración 1 Voltaje de referencia real.

A partir de esta ilustración, se establece el valor exacto en el código, el cual nos permitirá obtener el valor del voltaje lo más exacto posible, a partir del cálculo.

$$V = \frac{code12 \times VREF}{4095}$$

Siendo que.

- Code12: Corresponde al código binario que obtiene el conversor análogo digital.
- VREF: voltaje máximo del microcontrolador (3,317)
- 4095 máxima cantidad de bits admitida por el rasperry pi pico 2W (1111 1111 1111)

A partir de esta ecuación, se logra obtener el voltaje medido por el conversor análogo de la rasperry pi pico 2W, por lo cual se establece el siguiente código.

```
import machine
import utime

ADC_PIN = 26      # GP26 -> ADC0 (cambia a 27 o 28 si usas ADC1/ADC2)
VREF    = 3.316   # mide tu 3V3 real y ajústalo aquí para mejor exactitud
PERIOD  = 0.02    # segundos entre lecturas (0.05–0.5 )

adc = machine.ADC(ADC_PIN)

while True:
```

```
    raw16  = adc.read_u16()      # 0..65535 (12 bits alineado a la izq.)
    code12 = raw16 >> 4          # 0..4095 (12 bits reales)
    volts  = (code12 * VREF) / 4095.0
    print(f"Voltaje:{volts:.4f}")        # <-- SOLO un número por línea
    utime.sleep(PERIOD)
```

En este caso, como se observa, se hace uso del comando "raw16", esto es debido a que el programa de microphyton lee únicamente números binarios de 16bits, en este caso, se hace uso del comando.

Code12=raw16>>4

Esto con el fin de eliminar los cuatro primeros bits menos significativos del mensaje, esto es necesario, ya que entra en conflicto el programa con el micro, por ende, es fundamental realizar este paso, con el fin de que los datos leídos por el micro sean correctos y concuerden con la realidad.
Por ende, se establece un voltaje de 2V DC haciendo uso de una fuente dual conectando el voltaje de entrada al pin GP26 y tierra a alguna tierra del rasperry pi pico 2W, por lo cual se observa en el programa.

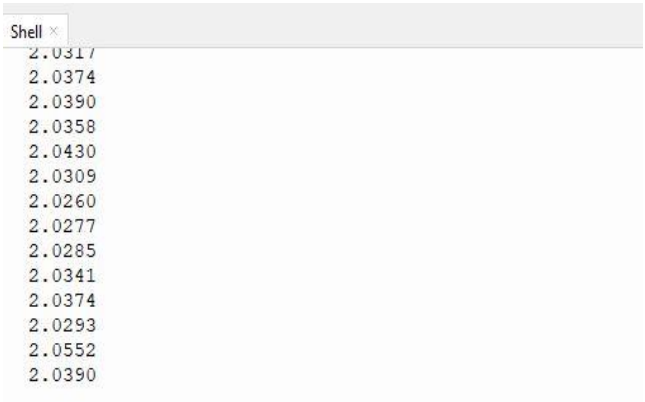

Ilustración 2 Voltaje obtenido por el conversor ADC

Como se observa en la ilustración número dos, el voltaje obtenido varia con respecto a cada una de las tomas que realiza el programa, esto debido a que la fuente no es completamente precisa, por lo cual llega a variar esto.
Así mismo se activo el PLOTTER del programa Thonny, en este caso, se observará el comportamiento del voltaje con respecto al tiempo de cada una de la toma de datos, así mismo se establece un valor de periodo de 0.5 segundos, lo cual es el tiempo que se tomara entre cada una de las lecturas, así mismo se varía el voltaje a 2,4V.

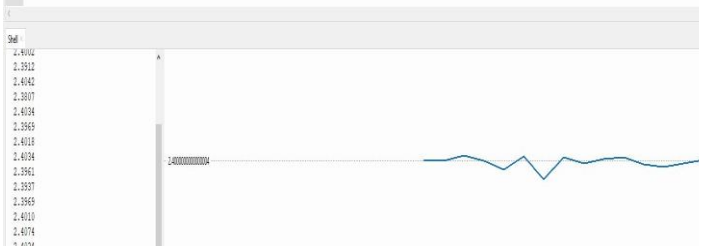

Ilustración 3 Plotter para periodo de 0,5.

Como se observa, la toma de los datos es mas lenta, así mismo se determina a partir de la grafica dada, que los datos obtenidos no son todos iguales, variando en el tiempo, por lo cual se refuta nuevamente que la precisión del voltaje suministrado por la fuente es 100% confiable.
Así mismo, se realiza el mismo proceso, pero estableciendo un valor de periodo de 0,05 como se observa a continuación.

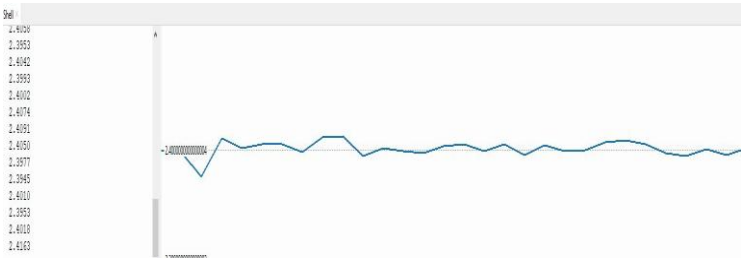

Ilustración 4 Plotter para periodo de 0,05

Como se observa nuevamente, en este caso la toma de datos es mayor, esto debido a que el tiempo de la medida y análisis de datos es mucho mayor con respecto al periodo de 0,5.
Por otro lado, se realizan los cálculos correspondientes para los ejemplos dados por el maestro, así mismo se realizar ciertas modificaciones en el programa con el fin de observar el proceso de RAW16 y el obtener los 12 bits necesarios para el ADC del rasperry, como se observa a continuación.

```
import machine
import utime

ADC_PIN = 26      # GP26 -> ADC0 (cambia a 27 o 28 si usas ADC1/ADC2)
VREF    = 3.316   # mide tu 3V3 real y ajústalo aquí para mejor exactitud
PERIOD  = 0.9     # segundos entre lecturas (0.05–0.5 )

adc = machine.ADC(ADC_PIN)
while True:
    raw16  = adc.read_u16()      # 0..65535 (12 bits alineado a la izq.)
    code12 = raw16 >> 4          # 0..4095 (12 bits reales)
    volts  = (code12 * VREF) / 4095.0
    print(f"Voltaje:{volts:.4f}")        # <-- SOLO un número por línea
    print("---------")
    print(raw16)
    numero_binario1 = bin(raw16)[2:]  # Eliminar el prefijo '0b'
    print(f"El número binario 16 bits es: {numero_binario1}")
    print("---------")
    print(code12)
    numero_binario = bin(code12)[2:]  # Eliminar el prefijo '0b'
    print(f"El número binario 12 bits es: {numero_binario}")
    print("---------")
    utime.sleep(PERIOD)
```

A partir de este programa se realizan las mediciones de los valores dados por el maestro como se observa a continuación.

- 2048(100000000000)

$$V = \frac{2048 \times 3{,}317}{4095}$$
$$V = 1{,}65V$$

Por lo cual, se establece este valor del voltaje en la fuente y se observa en el programa.

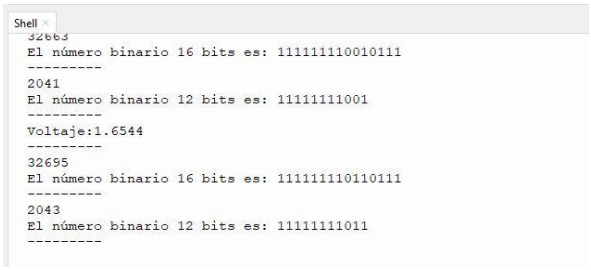

Ilustración 5 Voltaje 1,65V.

Como se observa, los valores dados a partir de establecer aproximadamente 1,65V en la fuente no es completamente preciso, mas si llega a acercarse al valor previamente definido (2048), así mismo se comprueba como es el proceso de "raw16>>4" eliminando los 4 bits menos significativos con el fin de que sean leídos y analizados por el ADC de la rasperry.

- 1024(010000000000)

$$V = \frac{1024 \times 3{,}317}{4095}$$
$$V = 0{,}82V$$

Así mismo, se comprueba que el voltaje obtenido corresponda con lo observado en el programa.

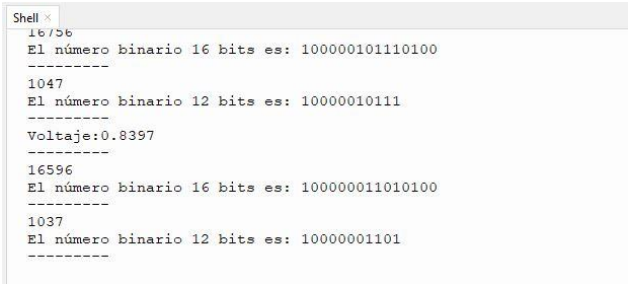

Ilustración 6 Voltaje 0,82V

- 4095(111111111111)

$$V = \frac{4095 \times 3{,}317}{4095}$$
$$V = 3{,}317V$$

Ilustración 7 Voltaje 3,317 V

En este caso, se puede determinar que directamente por la baja precisión de la fuente al suministrar el voltaje, presenta un déficit con respecto a lo calculado, siendo que en ningún momento se mantiene constante.

Por otro lado, se implementaron 3 diferentes números decimales, con el fin de observar así mismo el comportamiento del ADC.

- 512(001000000000)

$$V = \frac{512 \times 3{,}317}{4095}$$
$$V = 0{,}41V$$

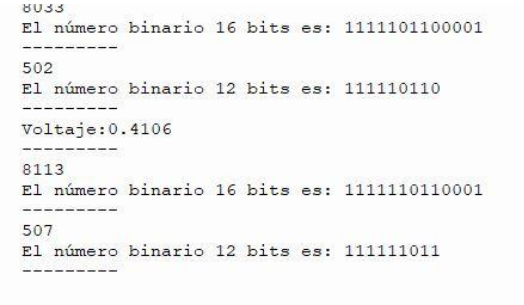

Ilustración 8 Voltaje de 0,41V

- 1900(011101101100)

$$V = \frac{1900 \times 3{,}317}{4095}$$
$$V = 1{,}53V$$

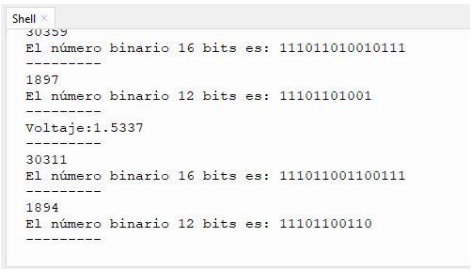

Ilustración 9 Voltaje de 1,53V

- 1500(010111011100)

$$V = \frac{1500 \times 3{,}317}{4095}$$
$$V = 1{,}21V$$

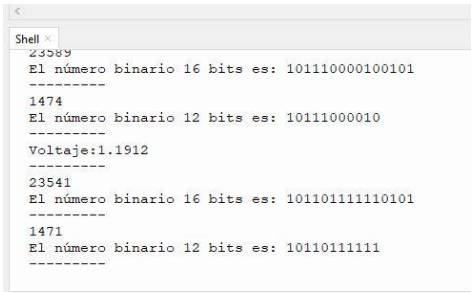

Ilustración 10 Voltaje de 1,21V

Como se observa en cada uno de los diferentes números decimales empleados, el valor no será exacto con respecto a la teoría, esto de forma que las fuentes empleadas no son de alta precisión, por lo cual el voltaje suministrado no será exacto, por otro lado podemos ver que se mantiene de alguna u otra forma concordancia con los datos expuestos de manera teórica, con respecto a lo analizado, por lo cual se puede inferir que el conversor ADC aunque no 100% preciso, cumple con su función de manera concisa.

III. Muestreo de señales

Inicialmente, se hace uso del software de Matlab, esto con el fin de observar el código proporcionado por el docente "adc.m", del cual se obtiene la siguiente grafica.

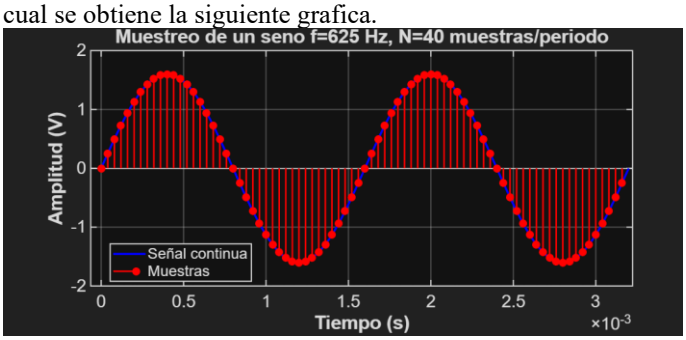

Ilustración 11 Grafica obtenida a partir del código del docente.

Como se observa, la gráfica presenta una señal senoidal con frecuencia de 625Hz, así mismo esta presenta una cantidad de 40 muestras por periodo, por ende, al hacer uso de estos valores conocidos, se calcula la frecuencia de muestreo como se observa continuación.

$$fs = f \times N = 625Hz \times 40 = 25{,}000Hz$$

La frecuencia de muestreo de la señal en cuestión corresponde a 25kHz, lo que significa que la frecuencia de muestreo es mucho mayor con respecto a la frecuencia de la señal por lo cual la exactitud de los datos obtenidos es más precisa.

Así mismo se calcula el periodo de muestreo, como se observa a continuación.

$$T_s = \frac{1}{25000Hz} = 0{,}000004s = 4\mu s$$

El periodo de muestreo para la señal es 4microsegundos.

Así mismo, en el programa de Thonny, se observa el código "ADC_sampling.py" suministrador por el docente. El código captura muestras de una señal analógica a través del convertidor ADC de la Raspberry Pi Pico, tomando una cantidad específica de lecturas por periodo de la señal (configurable mediante el parámetro SAMPLES_PER_PERIOD). Estas mediciones se realizan de forma sincronizada para garantizar una frecuencia de muestreo constante y precisa, ideal para analizar señales periódicas como la corriente alterna o sensores analógicos.

Los datos capturados se guardan en un archivo CSV llamado "adc_capture_2.csv" directamente en la memoria de la Raspberry Pi Pico. Este archivo contiene para cada muestra: el número de muestra, el tiempo transcurrido en microsegundos, el valor digital en formato decimal, binario y hexadecimal, así como el voltaje calculado. Estos datos exportados permiten realizar un análisis detallado posterior en herramientas como Python, MATLAB u hojas de cálculo para estudiar las características de la señal analógica recibida.

Por ende, para hacer análisis del código, se establece una señal con una frecuencia de 625Hz así mismo con 3Vpp y con componente DC=1.6 Voltios, esta señal se comprueba inicialmente en el osciloscopio, con el fin de garantizar una adecuada implementación, con el fin de prevenir algún daño en el microcontrolador.

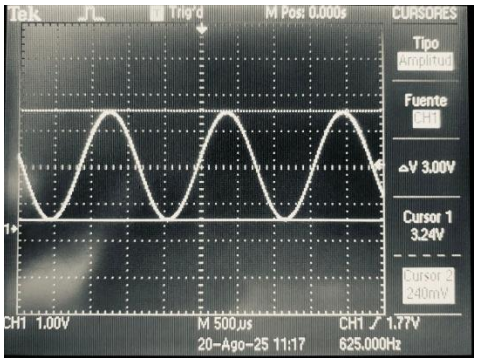

Ilustración 12 Señal generada.

Como se observa, la señal presenta un voltaje correspondiente de 3Vpp, así mismo una frecuencia de 625Hz, adicionalmente se observa a partir de los cursores, el nivel DC de 1,6V establecido, esto se hace con el fin de que todos los valores sean positivos, ya que el micro no tiene la capacidad de leer voltajes negativos, por eso mismo es necesario modificar el código suministrado por el docente "adc.m" con el fin de agregar un nivel DC de 1,6V, esto con el fin de comparar las gráficas obtenidas de manera experimental con lo observado en la teoría.

Por ende, se realiza la conexión del generador de señales, suministrando la señal observada en el osciloscopio, de esta manera se observa el funcionamiento del programa y se obtienen los datos almacenados.

n	t_us	dec	bin	hex	volts
0	564	3902	111100111110	0xF3E	3.144469
1	1676	3917	111101001101	0xF4D	3.156557
2	2434	3824	111011110000	0xEF0	3.081612
3	3159	3624	111000101000	0xE28	2.92044
4	4007	3313	110011110001	0xCF1	2.669817
5	5008	2827	101100001011	0xB0B	2.278168
6	6014	2275	100011100011	0x8E3	1.833333
7	7012	1732	11011000100	0x6C4	1.395751
8	8006	1207	10010110111	0x4B7	0.972674
9	9007	776	1100001000	0x308	0.625348
10	10011	488	111101000	0x1E8	0.39326

Ilustración 13 Datos almacenados.

El programa tiene la función de alamacenar 40 muestras (0;39), para cada una de estas muestras, se obtiene el tiempo de esta, el valor obtenido por en ADC del micro, en notación decimal, binaria y hexadecimal, así mismo realizando el calculo previamente visto, se obtiene el voltaje para cada uno de los puntos.

Estos datos son dispuestos en Matlab, donde se grafican tiempo vs voltaje, por lo cual se obtiene la siguiente señal.

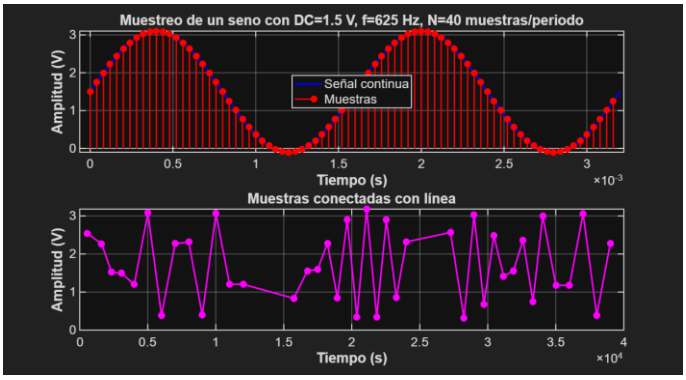

Ilustración 14 Señal obtenida 625Hz.

En este caso, como se observa la señal no tiene mucha similitud con lo planteado teóricamente, esto puede deberse a la alta frecuencia empleada en la señal, por lo cual hay una distorsión al momento de tomar los datos, para corroborar esto, se hace uso de una frecuencia menor, en este caso de 50Hz.

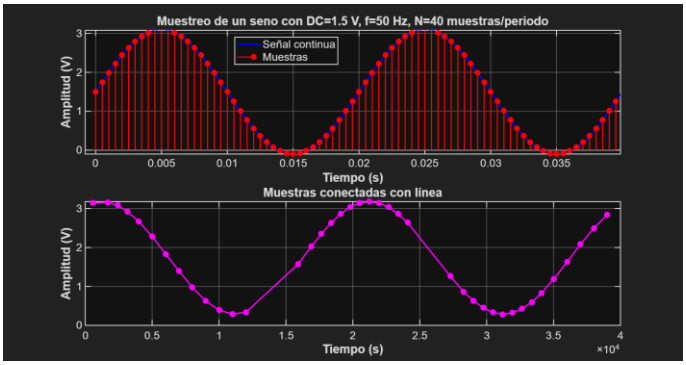

Ilustración 15 Señal obtenida 50Hz.

Como se observa, la señal correspondiente presenta una forma más simétrica, pero a su vez la tasa de muestreo no es uniforme para todos los puntos, esto debido a que teóricamente se usa un reloj de muy alta exactitud con el fin de garantizar adecuadamente las medidas, mientras que, en la realidad, la rasperry no cuenta con este, siendo que la toma de datos no es exacta, por lo cual presenta una variación tan alta con respecto a los datos obtenidos teóricamente.

A partir de estos datos, se observa la tasa de muestreo, para ello se da clic en los puntos más uniformes de la señal experimental, así mismo se establecen puntos similares en la señal teórica, evidentemente la tasa de muestreo será exacta en esta debido al muestreo exacto empleado en Matlab.

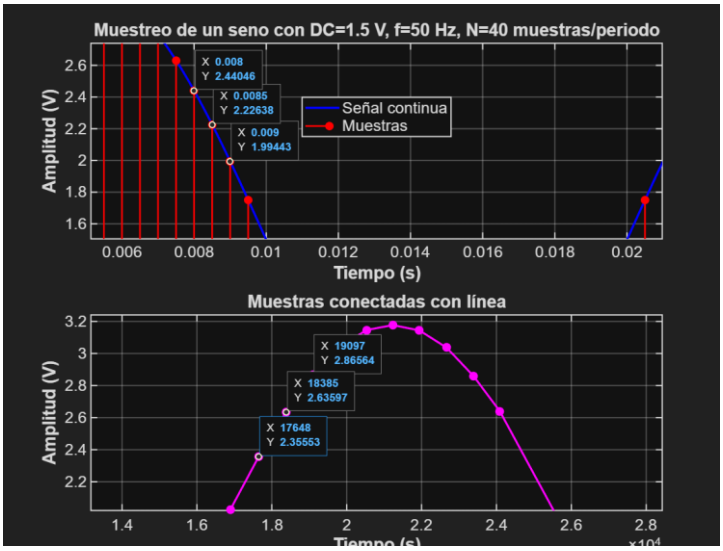

Ilustración 16 Tasa de muestreo teórica y experimental.

Como se observa se toman los datos para X de los puntos seleccionados, en este caso se compara el punto actual con el anterior en el caso teórico.

$$0{,}009 - 0{,}0085 = 0{,}0005$$

Así mismo con el punto anterior.

$$0{,}0085 - 0{,}008 = 0{,}0005$$

Como se observa, cada muestra se mantiene cada 0,0005 segundos, lo que indica una uniformidad con respecto a la toma de estas, a diferencia de la señal experimental, donde se realiza la misma comparación.

$$(1.9097 - 1.8385)x10^4 = 0.712x10^4$$

Así mismo con el siguiente punto.

$$(1.8385 - 1.7648)x10^4 = 0.737x10^4$$

Como se observa, los tiempos no son constantes, llegan a variar, siendo que de estos son los puntos con mas uniformidad de la gráfica, aun así presentado variaciones de tiempo de acuerdo a la toma de estos, por lo cual se puede inferir que el hecho de la toma de datos por parte de la rasperry varia considerablemente debido a que esta no cuenta con un reloj de alta precisión que determine exactamente la toma de datos, así mismo para rectificar esto, se realiza la toma de datos para una frecuencia de 20Hz, con una cantidad de 100 muestras por periodo, lo cual se obtuvo.

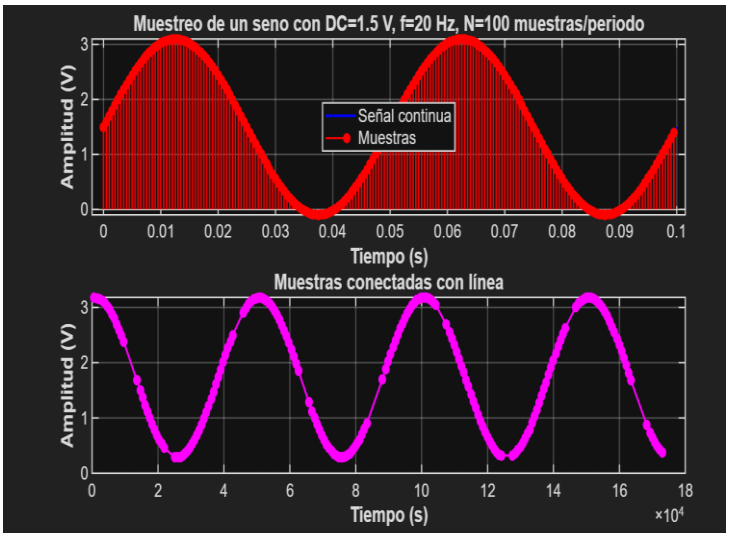

Ilustración 17 Señal 20Hz y 100 muestras.

Como se observa, a pesar del aumento de muestras y menor frecuencia, la señal experimental obtenida presenta aun un déficit con respecto a su uniformidad a la toma de los datos, aunque evidentemente tiene una forma más uniforme, la captura de muestras no es completamente uniforme.

Así mismo, se realizó la toma de datos para una señal de 50Hz pero a menor cantidad de muestras, 20 muestras por periodo, lo cual se obtiene la siguiente señal.

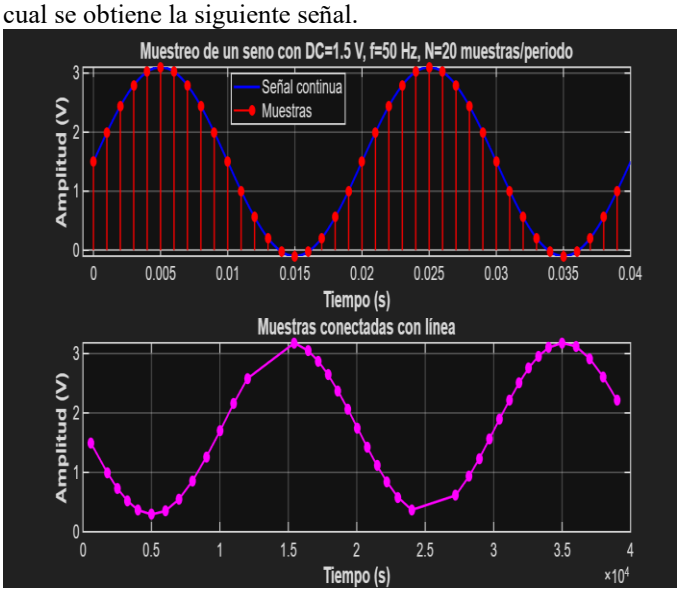

Ilustración 18 Señal 50Hz y 20 muestras.

A partir de lo obtenido, se observa que la señal a diferencia de dos puntos presenta una mejor uniformidad en la toma de datos, nuevamente se observa la tasa de muestreo para comparar los datos obtenidos.

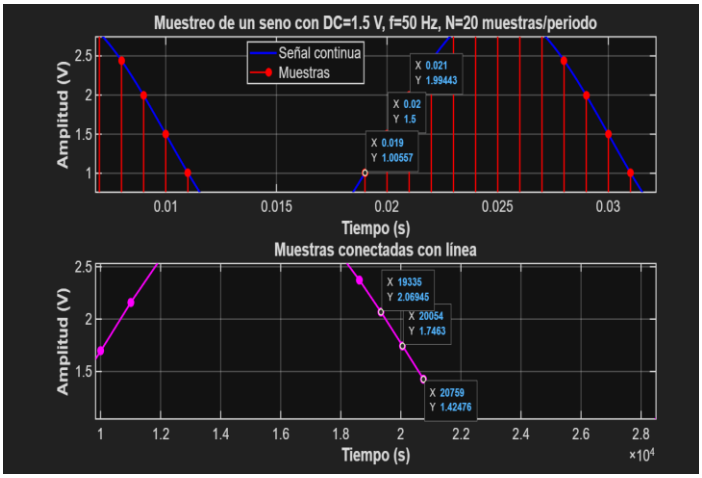

Ilustración 19 Tasa de muestreo teórica y experimental.

Para el caso teórico, la tasa de muestreo es uniforme, en este caso cada 0,001 segundos toma una muestra, a diferencia de la señal experimental.

$$(2.0759 - 2.0054)x10^4 = 0.705x10^4$$

Así mismo, se obtiene la tasa de muestreo para la siguiente muestra

$$(2.0054 - 1.9335)x10^4 = 0.719x10^4$$

Se puede determinar que la tasa de muestreo ronda los 0.719x10^4 segundos, lo cual indica que la uniformidad de la toma de muestras ronda los 0,700 x10^4 segundos, sin tomar en cuenta los datos de dispersión que el cronometro de la rasperry omite tomar.

VI. Análisis estadístico

Se hace uso de la fuente DC del laboratorio conectando esta al ADC de la rasperry, se establecen 5 voltajes diferentes, y se procede a ejecutar el código "sampling_2.py" suministrado por el docente, este código nos permite obtener los datos obtenidos por el ADC, en este caso se obtendrán dos archivos, uno que contiene la información del voltaje con respecto a la cantidad de muestras y otro que presenta el histograma, los datos obtenidos por el programa son.

```
Ciclo de muestreo completado con 10000 muestras.
Estadísticas del ciclo:
  Total de muestras: 10000
  Media: 1.69987 V
  Desviación Estándar: 0.23936 V
Histograma de lecturas (res. 12 bits):
{1627: 5, 1628: 3, 1629: 8, 1630: 9, 1631: 4, 1632: 8, 1633: 6, 1634: …
Datos del histograma guardados en '3'
Programa finalizado.
```

Ilustración 20 Datos obtenidos en el programa.

Como se observa, el programa da como resultado el total de muestras obtenidas, la media de los datos obtenidos, desviación estándar de estos y así mismo el histograma de los datos obtenidos, así mismo indica la ubicación de los datos almacenados.

Este proceso se realiza para los 5 voltajes, dando como resultado la siguiente tabla.

Test	V in (DC)	Media	Desviación estándar	Nombre de los archivos
1	0.8	0,86379	0,23576	"0,8.txt" "h0,8.txt"
2	1,4	1,47668	0,24125	"1,4.txt" "h1,4.txt"
3	1,6	1,69987	0,23936	"1,6.txt" "h1,6.txt"
4	2	2,14733	0,23315	"2.txt" "h2.txt"
5	2,6	2,74544	0,23420	"2,6.txt" "h2,6.txt"

Tabla 1 Datos obtenidos por el programa "sampling_2"

A partir del primer archivo obtenido por parte de las mediciones, este contiene el voltaje con respecto al tiempo en milisegundos del voltaje obtenido y medido por el microcontrolador, a partir de este se realiza el siguiente código, el cual permite calcular la media y la desviación estándar para cada uno de los voltajes.

```
archivo = '0,8.csv';

opts = detectImportOptions(archivo, 'Delimiter', '\t');
datos = readmatrix(archivo, opts);
voltaje = datos(:,2);
media = mean(voltaje);
desviacion = std(voltaje);
fprintf('Media: %.4f\n', media);
fprintf('Desviacion estandar: %.4f\n', desviacion);
```

Este codigo tiene la función de leer apropiadamente el voltaje presente en la segunda columna y a partir de obtener los datos, utiliza la función de mean y std para calcular la media y la

desviación respectivamente, posteriormente imprime estos valores, en el caso del voltaje de 0,8V se obtiene.

```
Media: 0.9672
Desviacion estandar: 0.2510
```

Ilustración 21 Media y desviación estándar 0,8V.

De esta manera se realiza para los otros 4 voltajes establecidos, por lo cual se observan los siguientes resultados.

Test	V in (DC)	Media experimental	Desviación estándar experimental
1	0.8	0,9672	0,2510
2	1,4	1.5133	0.2535
3	1,6	1.7835	0.2540
4	2	2.1460	0.2488
5	2,6	2.7839	0.2503

Tabla 2 Datos obtenidos a partir del programa creado.

Como se observa, se puede calcular el porcentaje de error a partir de los datos obtenidos por el programa en microphyton y los datos obtenidos a partir del código creado.

Test	V in (DC)	Media	Media experimental	% de error
1	0.8	0,86379	0,9672	11%
2	1,4	1,47668	1.5133	2,4%
3	1,6	1,69987	1.7835	4,9%
4	2	2,14733	2.1460	0,06%
5	2,6	2,74544	2.7839	1,4%

Tabla 3 Porcentaje de error con respecto a la media.

Test	V in (DC)	Desviación estándar	Desviación estándar experimental	% de error
1	0.8	0,23576	0,2510	6,4%
2	1,4	0,24125	0.2535	5%
3	1,6	0,23936	0.2540	6%
4	2	0,23315	0.2488	6,7%
5	2,6	0,23420	0.2503	6,8%

Tabla 4 Porcentaje de error con respecto a la desviación estándar

Como se observa, el porcentaje de error con respecto al cálculo de la media varia bastante, esto debido al proceso de análisis del mismo programa, siendo que por el mismo procesamiento de los datos a partir de la rasperry no es 100% exacto y puede afectar como se observa.

Así mismo, a partir del segundo archivo obtenido se grafica el histograma de bits con respecto a la frecuencia haciendo uso del siguiente código.

```
datos = readmatrix('h2,6.csv');
valores = datos(:,1);
figure;
histogram(valores, 20); % el 20 indica el número de bins, puedes ajustarlo
title('Histograma de Voltajes Medidos');
xlabel('Voltaje [V]');
ylabel('Frecuencia');
grid on;
```

Siendo que el código anterior analiza las columnas de los datos obtenidos y genera la gráfica.

- 0,8V

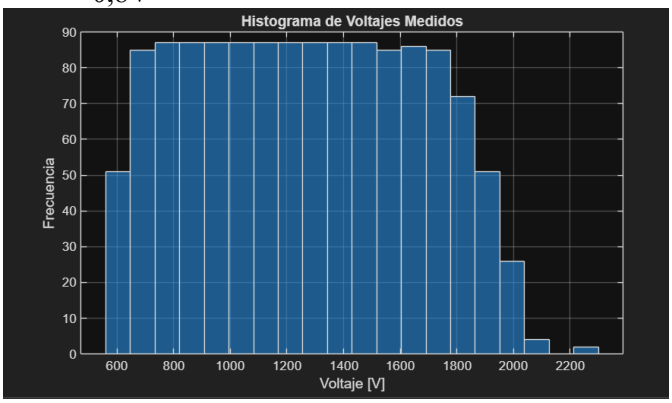

Ilustración 22 Histograma 0,8V

- 1,4V

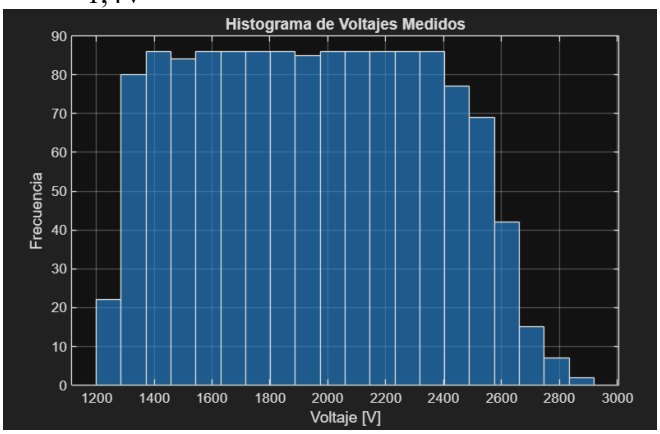

Ilustración 23 histograma 1,4V

- 1,6V

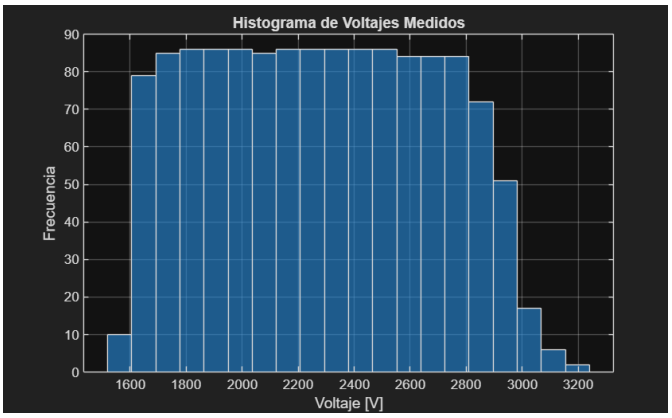

Ilustración 24 Histograma 1,6V

- 2V

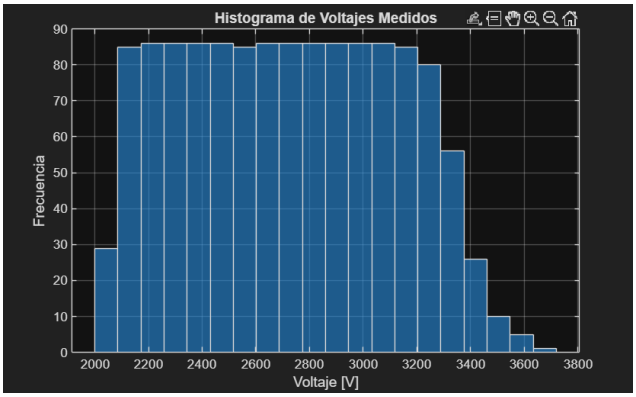

Ilustración 25 Histograma 2V

- 2,6V

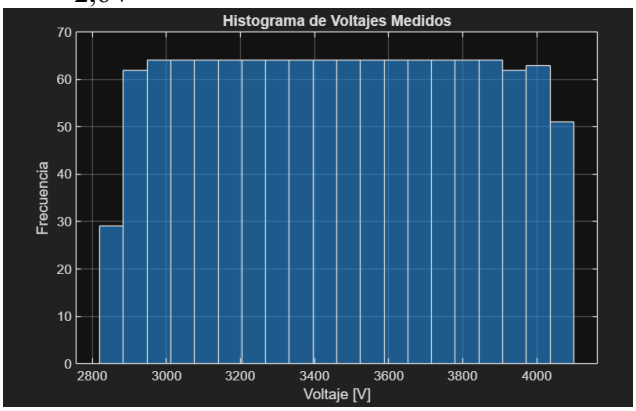

Ilustración 26 Histograma 2,6V

Los histogramas obtenidos para cada uno de los voltajes de entrada permiten observar la distribución de las muestras digitalizadas por el ADC. En todos los casos, los datos se concentran alrededor de un valor medio, pero con una dispersión significativa atribuida al ruido eléctrico y a la variación de la fuente de voltaje. Esta dispersión se refleja en las desviaciones estándar, que aunque son bajas, muestran un error del 5–7% respecto a los valores experimentales de referencia.

Por otro lado, la comparación de las medias calculadas en MATLAB frente a las obtenidas directamente en la Raspberry Pi muestra errores más altos en algunos casos, especialmente a bajo voltaje (11% para 0,8 V). Esto se debe principalmente a la limitada precisión de la fuente de voltaje y a la resolución del ADC, que introduce mayor incertidumbre en rangos bajos. A medida que aumenta el voltaje, el error disminuye y se mantiene en torno al 1–2%, lo cual valida el comportamiento esperado del conversor.

Por ende se puede resumir que los histogramas confirman que el ADC cumple su función de cuantización, pero evidencian que la calidad de las mediciones depende de la estabilidad de la fuente, la resolución del convertidor y la cantidad de muestras adquiridas.

IX. Análisis.

El análisis conjunto de los datos muestra que el ADC del Raspberry Pi Pico 2W es capaz de digitalizar señales analógicas de forma consistente, aunque no con la exactitud de equipos especializados. Las principales fuentes de error detectadas fueron:

- La variabilidad de la fuente DC empleada en las pruebas.
- La resolución de 12 bits, que introduce cuantización y limita la precisión a escalones discretos.
- La falta de un reloj de muestreo de alta precisión, lo cual genera variaciones en la adquisición temporal.

A pesar de estas limitaciones, los resultados experimentales se aproximaron en gran medida a los valores teóricos, validando el uso del ADC para aplicaciones de bajo costo y fines académicos.

X. Conclusiones.

1. El conversor ADC del Raspberry Pi Pico 2W cumple adecuadamente con su función de digitalización, aunque presenta errores derivados del ruido, la fuente de alimentación y la resolución de 12 bits.
2. El error relativo en la media disminuye con voltajes más altos, lo que indica que el ADC es más confiable en rangos intermedios y altos.
3. La desviación estándar se mantuvo con un error entre el 5% y 7%, evidenciando una dispersión constante en las mediciones.
4. Los histogramas reflejaron la naturaleza estadística de las mediciones, mostrando la distribución de los valores digitalizados.
5. El uso combinado de MicroPython y MATLAB permitió no solo adquirir datos, sino también procesarlos y analizarlos de forma estadística, facilitando la interpretación de resultados.
6. En general, el laboratorio permitió comprender el proceso de conversión A/D y sus limitaciones prácticas, reforzando los conceptos teóricos de muestreo y cuantización.

Referencias

[1] Raspberry Pi Foundation. (2023). *Raspberry Pi Pico Datasheet*. Retrieved from https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[2] Alam, M., & Alam, M. (2024, 29 noviembre). How to use ADC in Raspberry Pi Pico | ADC Example Code. How To Electronics. https://how2electronics.com/how-to-use-adc-in-raspberry-pi-pico-adc-example-code

[3] Giraldo, S. A. C. (2021b, marzo 18). Entradas analógicas. Control Automático Educación. https://controlautomaticoeducacion.com/sistemas-embebidos/micropython/adc-pico-esp

[4] Instructables. (2024, 1 octubre). Measuring Small Voltages With Pi Pico ADC and Comparison With Microchip MCP3208 Using Cytron EDU PICO. Instructables. https://www-instructables-com.translate.goog/Measuring-Small-Voltages-With-Pi-Pico-ADC-and-Comp/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc