Laboratorio ADC

Belky Valentina Giron Lopez
est.belky.giron@unimilitar.edu.co
Docente: José De Jesús Rúgeles

Resumen — En este laboratorio se exploró la conversión A/D utilizando la Raspberry Pi Pico. Se verificó la relación entre los valores de 12 bits del ADC y los 16 bits entregados por read_u16(). Posteriormente, se realizaron experimentos de muestreo de señales senoidales y de análisis estadístico con voltajes constantes. Los resultados obtenidos demostraron la validez del proceso de digitalización, la importancia de la tasa de muestreo. Con esto se consolidaron los fundamentos teóricos y prácticos de la conversión analógica-digital aplicada a sistemas de comunicación digital.

Abstract **--** In this lab, A/D conversion was explored using the Raspberry Pi Pico. The relationship between the 12-bit ADC values and the 16-bit values returned by read_u16() was verified. Subsequently, experiments involving sinusoidal signal sampling and statistical analysis with constant voltages were performed. The results demonstrated the validity of the digitization process and the importance of the sampling rate. This consolidated the theoretical and practical foundations of analog-to-digital conversion applied to digital communication systems.

I. Introducción

La conversión analógica–digital es esencial en las comunicaciones digitales, pues permite transformar señales continuas en datos discretos procesables por sistemas digitales. En este laboratorio se utilizó la Raspberry Pi Pico, que cuenta con un ADC de 12 bits, para estudiar el proceso de cuantización, el muestreo de señales y el análisis estadístico de las mediciones.

II. Desarrollo de experimentos

Parte A

La Raspberry Pi Pico W cuenta con varios pines destinados a las entradas del conversor analógico–digital (ADC). En este laboratorio se trabajó con el pin GP26, correspondiente al ADC0. El voltaje de referencia máximo del conversor es de aproximadamente 3,3 V, el cual puede verificarse en el pin 36 de la tarjeta. Para confirmar este valor, se emplea un multímetro que permite medir con precisión el voltaje real presente en la salida de referencia.

Al realizar la medición con el multímetro, se obtuvo un valor de 3.305 V como voltaje de referencia. Este valor se emplea directamente en el código para realizar los cálculos y así obtener una estimación del voltaje lo más precisa posible.

Luego implementamos el siguiente código:

```
ADC_PIN = 26
VREF = 3.305
PERIOD = 0.02

adc = machine.ADC (ADC_PIN)

while True:

raw16 = adc.read_u16()
code12 = raw16 >> 4
volts = (code12 * VREF)
print(f"Voltaje:{volts:.4f}")
utime.sleep(PERIOD)
```

En este caso se utiliza la variable “raw16”, debido a que el programa en MicroPython entrega la lectura del ADC en un valor de 16 bits, aunque la conversión real del microcontrolador es de 12 bits. Para interpretar correctamente los datos, este valor se ajusta posteriormente mediante el corrimiento de bits. Adicionalmente, se modificó en el código el valor de VREF, reemplazando el valor nominal de 3.3 V por el valor real medido de 3.305 V, con el fin de obtener cálculos de voltaje más precisos.

En el código se estableció la siguiente línea: code12 = raw16 >> 4 esto se realizó con el fin de eliminar los cuatro bits menos significativos del valor leído, ya que de lo contrario se generarían inconsistencias entre el programa y el microcontrolador. De esta manera, los datos obtenidos corresponden con mayor exactitud a la realidad. Para la práctica, el pin GP26 (ADC0) se conectó a un potenciómetro, lo que permitió variar el voltaje de entrada en un rango de 0 a 3.305 V, con la referencia de tierra conectada a GND de la Raspberry Pi Pico W.

Luego se activó la función de Plotter de Thonny para visualizar el comportamiento del voltaje en función del tiempo a partir de cada muestra obtenida. En el programa se configuró un período de 0.5 segundos, que corresponde al intervalo entre cada lectura realizada por el ADC. Durante la práctica, el voltaje de entrada se ajustó mediante el potenciómetro, alcanzando valores cercanos a 3.305.

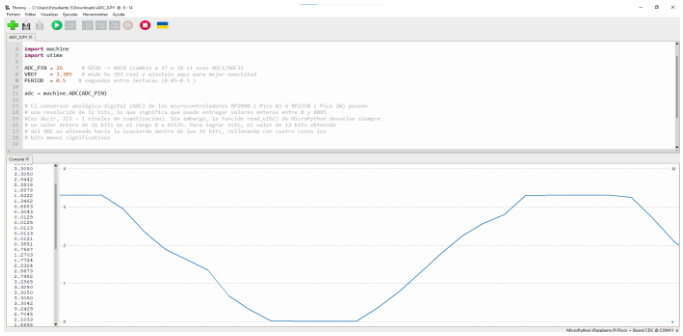

Fig. 1. Voltaje y plotter obtenido periodo 0,5.

La gráfica obtenida en el Plotter de Thonny refleja directamente la manera en que se giró el potenciómetro conectado al pin GP26. Al modificar la posición del potenciómetro, el voltaje de salida varía entre 0 V y 3.305 V, lo que se traduce en cambios visibles en el gráfico. Como el programa está configurado para tomar lecturas cada 0.5 segundos, se aprecia cómo el voltaje va descendiendo cuando el potenciómetro se gira hacia un extremo y vuelve a aumentar cuando se gira en sentido contrario, generando la forma de curva que aparece en la visualización.

Parte B

En esta parte de la práctica se ejecutó el programa adc.m en MATLAB, verificando la tasa de muestreo y el número de muestras por período.

Luego se ejecuto el programa ADC_Sampling.py y este generó un archivo .CSV llamado adc_capture_2.csv.

Se ajustó una señal de 3Vpp con DC=1.6 V y f=150 Hz se verificó con el osciloscopio antes de conectarla al microcontrolador.

Y se visualizo de la siguiente manera:

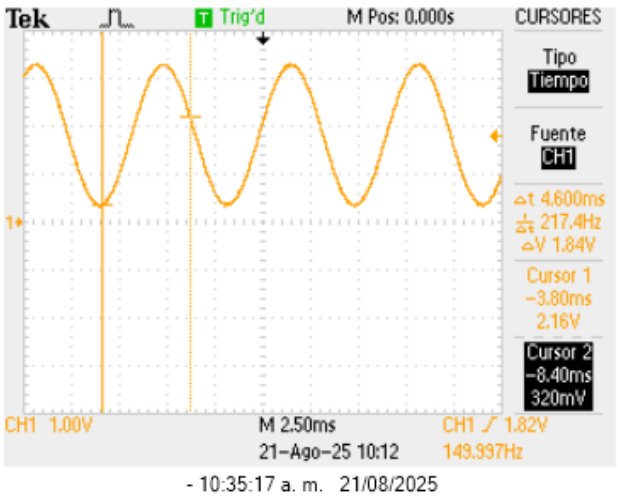

Fig. 2. Señal seno 3 Vpp f=150 Hz.

De manera teórica mediante el siguiente codigo de Matlab:

```
% Comunicaciones Digitales UMNG / jose.rugeles@Unimilitar.edu.co
% Programa: muestreo

f = 150;      % frecuencia de la señal (Hz)
T = 1/f;      % periodo (s)
A = 1.5;      % amplitud (V)
NT = 2;       % nº de periodos a capturar
N = 20;       % nº de muestras por periodo
ts = T/N;     % periodo de muestreo (s)

% Vector de tiempo para las muestras (N*NT puntos exactos)
t = 0:ts:(NT*T - ts);
V = A * sin(2*pi*f*t);  % señal muestreada

% Señal continua para comparación
t_cont = 0:ts/20:NT*T;
V_cont = A * sin(2*pi*f*t_cont);

% ===== Gráfica =====
figure('Position',[100 100 900 400]);  % tamaño ventana
plot(t_cont, V_cont, 'b-', 'LineWidth',2); hold on;
stem(t, V, 'r','filled','LineWidth',1.5);

xlabel('Tiempo (s)','FontSize',20,'FontWeight','bold');
ylabel('Amplitud (V)','FontSize',20,'FontWeight','bold');
title(sprintf('Muestreo de un seno f=%d Hz, N=%d muestras/periodo', f, N), ...
      'FontSize',20,'FontWeight','bold');

legend({'Señal continua','Muestras'},'FontSize',14,'Location','best');
grid on;
set(gca,'FontSize',16,'LineWidth',1.2);
```

Obtuve la siguiente grafica teórica:

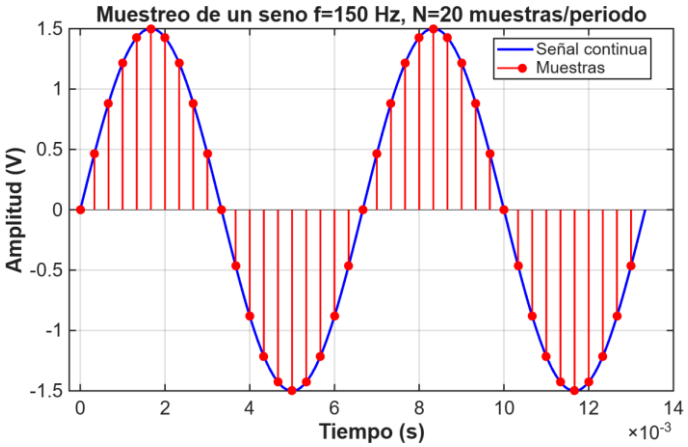

Fig. 3. Señal seno f=150 Hz teórica.

En la gráfica se observa una señal senoidal con frecuencia de 150 Hz, la cual fue muestreada utilizando 20 muestras por periodo. Luego se calcula la frecuencia de muestreo con la

siguiente ecuación:

$$f_s = f \times N = 150\,\text{Hz} \times 20 = 3{,}000\,\text{Hz}$$

Por lo tanto, la frecuencia de muestreo es de 3 kHz, lo que significa que es mucho mayor que la frecuencia de la señal original. Esto asegura que el muestreo cumpla con el criterio de Nyquist y que la representación digital conserve con precisión la forma de la onda senoidal.

De la frecuencia de muestreo se obtiene también el periodo de muestreo, el cual es:

$$T_s = \frac{1}{f_s} = \frac{1}{3000\,\text{Hz}} = 0.000333\,\text{s} = 333\,\mu\text{s}$$

Esto indica que el sistema toma una muestra cada 333 microsegundos, garantizando una reconstrucción muy buena de la señal.

Después mediante el archivo .csv obtenido del código ADC_sampling,py dado por el profesor obtuvimos la siguiente tabla:

	A	B	C	D	E	F
	n	t_us	dec	bin	hex	volts
	Number	Number	Number	Number	Text	Number
1	n	t_us	dec	bin	hex	volts
2	0	1021	1957	11110100101	0x7A5	1.5795
3	1	4015	1806	11100001110	0x70E	1.4576
4	2	6356	3823	111011101111	0xEEF	3.0855
5	3	8367	969	1111001001	0x3C9	0.78206
6	4	10646	1732	11011000100	0x6C4	1.3979
7	5	12907	3889	1111001100...	0xF31	3.1387
8	6	15146	843	1101001011	0x34B	0.68037
9	7	16615	769	1100000001	0x301	0.62065
10	8	18856	3846	1111000001...	0xF06	3.104
11	9	21092	1887	11101011111	0x75F	1.523
12	10	23328	819	1100110011	0x333	0.661
13	11	24825	3123	110000110...	0xC33	2.5205
14	12	27052	3038	101111011110	0xBDE	2.4519
15	13	60761	2419	100101110...	0x973	1.9523
16	14	63752	1376	10101100000	0x560	1.1105
17	15	66073	3957	111101110101	0xF75	3.1936
18	16	68322	1033	10000001001	0x409	0.83372

Fig. 4. Archivo .csv.

y luego de obtener el archivo .csv empleo el siguiente código para visualizar la señal experimental:

```
data = readtable('/MATLAB Drive/adc_capture_2.csv');  % si es CSV


% ==== Extracción de datos ====
t = data.t_us * 1e-6;      % convertir microsegundos a segundos
V = data.volts;        % columna de voltajes

% ==== Gráfica ====
figure('Position',[100 100 900 400]);

% Dibujar las muestras
stem(t, V, 'r','filled','LineWidth',1.5); hold on;

% Señal reconstruida (lineal entre muestras)
plot(t, V, 'b-','LineWidth',1.5);

xlabel('Tiempo (s)','FontSize',20,'FontWeight','bold');
ylabel('Voltaje (V)','FontSize',20,'FontWeight','bold');
title('Muestreo y reconstrucción de la señal medida','FontSize',20,'FontWeight','bold');

legend({'Muestras','Señal reconstruida'},'FontSize',14,'Location','best');
grid on;
set(gca,'FontSize',16,'LineWidth',1.2);
```

y obtuvimos la siguiente gráfica:

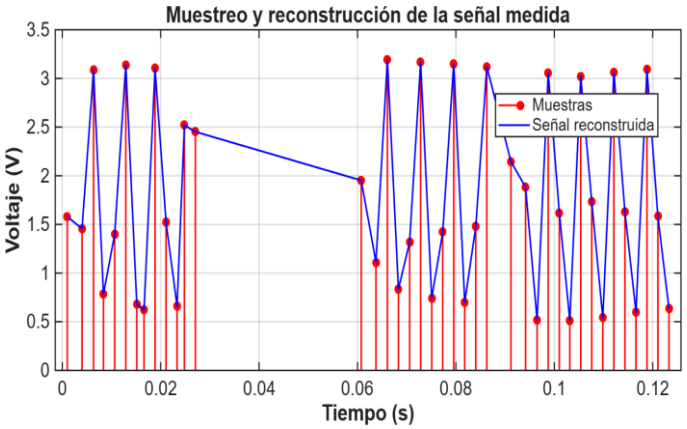

Fig. 5. Señal seno f=150 Hz experimental.

En este caso se observa que la señal medida no guarda una gran similitud con la planteada de manera teórica. Esto puede explicarse por la frecuencia utilizada, lo que genera cierta distorsión en el proceso de adquisición de datos.

Para comprobar esta conclusión, se decidió emplear una frecuencia más baja, en este caso de **50 Hz**, con el fin de obtener una representación más fiel de la señal.

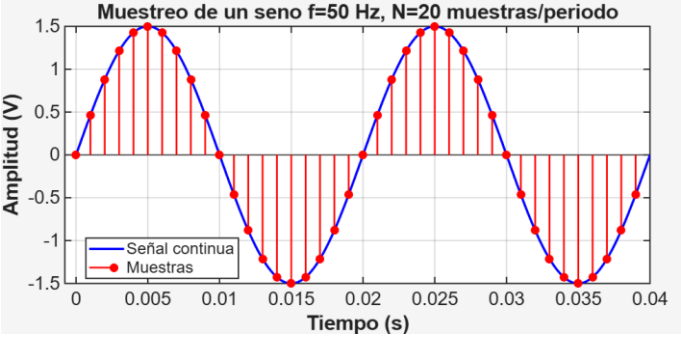

Fig. 6. Señal seno f=50 Hz teórica.

Fig. 7. Señal seno f=150 Hz experimental

En pocas palabras, las gráficas se ven diferentes porque para la señal de 150 Hz las muestras no alcanzaron a representar bien la forma real de la onda, lo que generó una distorsión visible. Al bajar la frecuencia a 50 Hz, con el mismo muestreo, las muestras se distribuyen mejor y la señal se parece mucho más a la teórica. Esto demuestra que la calidad de la gráfica depende de la relación entre la frecuencia de la señal y la frecuencia de muestreo, sed puede ver una similitud mas con la frecuencia de 50 Hz, pero no es igual.

Parte c

Test	V in (DC)	Media	Desviación estándar	Nombre de los archivos
1	0.5	0.51049	0.00198	Test1 H1
2	1.0	1.00844	0.00299	Test2.csv H2
3	1.8	1.79778	0.00480	Test3.csv H3
4	2.5	2.51060	0.00625	Test4.csv H4
5	3.0	3.01250	0.00729	Test5.csv H5

Test1.csv

Fig. 8. Vin 0,5.

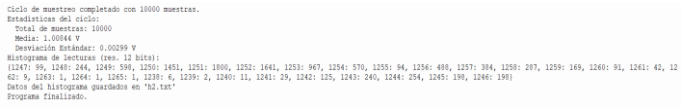

Fig. 9. Vin 1,0.

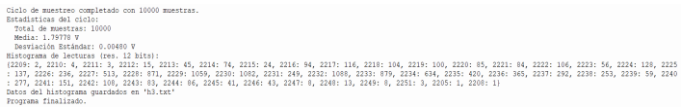

Fig. 10. Vin 1,8.

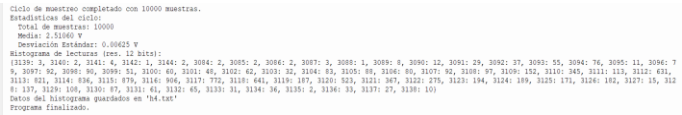

Fig. 11. Vin 2,5.

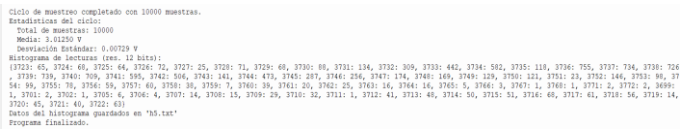

Fig. 12. Vin 3,0.

Luego empleamos el siguiente código para calcular la media y desviación estándar experimental a partir de los archivos .txt descargados del programa sampling_2.py dado por el profesor.

```
archivo = '/MATLAB Drive/test1.txt';
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);
```

Test	V in (DC)	Media experimental	Desviación estándar experimental
1	0.5	0.5105	0.0020
2	1.0	1.0084	0.0030
3	1.8	1.7978	0.0048
4	2.5	2.5106	0.0063
5	3.0	3.0125	0.0073

```
>> tests
Media: 0.5105
Desviacion estandar: 0.0020
>> tests
Media: 1.0084
Desviacion estandar: 0.0030
>> tests
Media: 1.7978
Desviacion estandar: 0.0048
>> tests
Media: 2.5106
Desviacion estandar: 0.0063
>> tests
Media: 3.0125
Desviacion estandar: 0.0073
```

Fig. 13. experimental.

Luego de realizar la parte experimental calculamos el porcentaje de error:

Test	Media % error	Desviación estandar % error
1	0.00196 %	1.01 %
2	0.00397 %	0.33 %
3	0.00111 %	0.00 %
4	0.00000 %	0.80 %
5	0.00000 %	0.14 %

Como se aprecia en la tabla anterior, los errores en la media son prácticamente despreciables (casi 0 %), mientras que en la desviación estándar son pequeños (todos < 1.1 %). Esto demuestra que los resultados experimentales son muy consistentes con los valores calculados por el programa en Matlab,

Luego graficamos los histogramas mediante el siguiente código:

```
datos = readmatrix('/MATLAB Drive/h1.txt');
valores = datos(:,1);
figure;
histogram(valores, 20);
title('Histograma de Voltajes experimentales');
xlabel('Voltaje [V]');
ylabel('Frecuencia');
grid on;
```

Obtenemos las siguientes graficas:

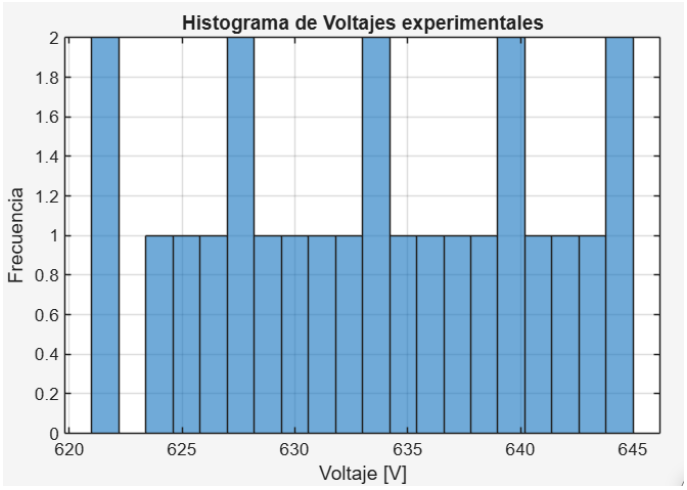

Fig. 14. Histograma 0.5.

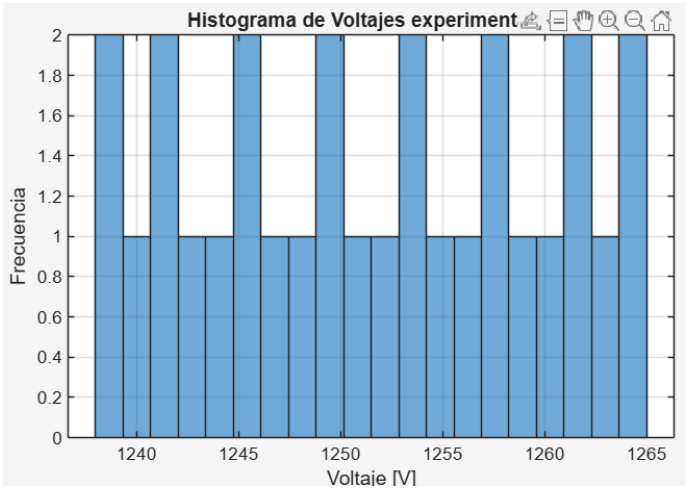

Fig. 15. Histograma 1.0.

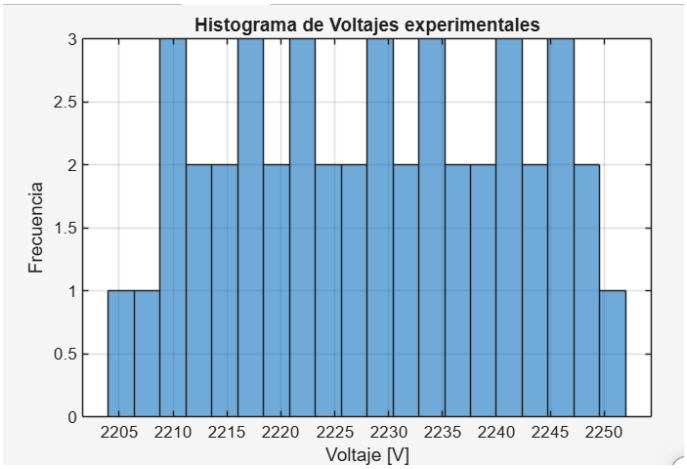

Fig. 16. Histograma 1.8.

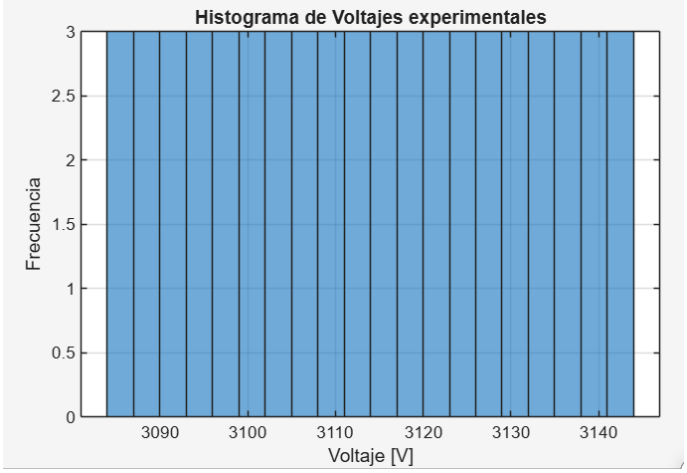

Fig. 17. Histograma 2.5.

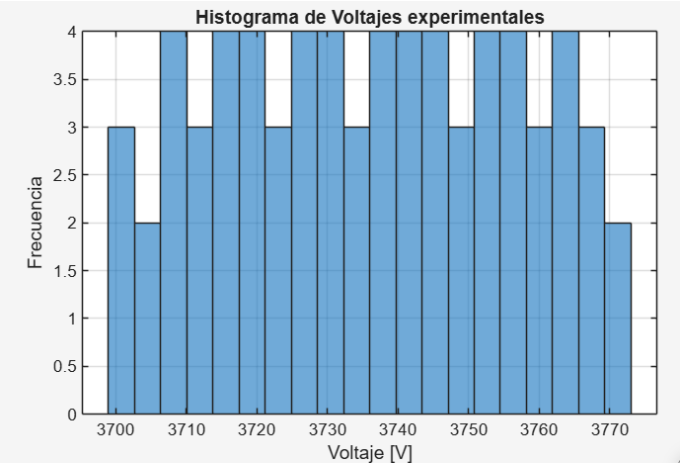

Fig. 18. Histograma 3.0.

Al revisar los histogramas de los cinco voltajes (0.5 V, 1 V, 1.8 V, 2.5 V y 3 V), se nota que en todos los casos las mediciones se agrupan alrededor de un valor central muy cercano al voltaje aplicado. Esto demuestra que el ADC de la Raspberry Pi Pico W trabaja de manera estable y confiable.

Se ve también que, a medida que el voltaje aumenta, las mediciones se dispersan un poco más, pero esa variación es mínima y corresponde al ruido normal del proceso de conversión.

En resumen, los histogramas confirman que el ADC es preciso en todo el rango de 0 a 3.3 V, entregando resultados consistentes y representando bien los valores de entrada.

Link repositorio GitHub:
https://github.com/belkyvalentina11/Comunicaci-n-digital-.git