



# LABORATORIO DE CONVERSOR A/D CON RASPBERRY PI PICO 2W

Vallentina Diaz Valbuena

[est.vallentina.diaz@unimilitar.edu.co](mailto:est.vallentina.diaz@unimilitar.edu.co)

Docente: José de Jesús Rugeles

## I. RESUMEN

Este artículo documenta la metodología y los hallazgos principales de una práctica de laboratorio orientada a caracterizar el funcionamiento del conversor analógico-digital (ADC) integrado en un microcontrolador de arquitectura moderna. El procedimiento experimental consistió en la aplicación controlada de señales de voltaje DC conocidas y señales alternas variables para evaluar parámetros clave como la linealidad, la resolución, la precisión y las limitaciones dinámicas del sistema de conversión. El proceso de adquisición de datos se implementó mediante un script en MicroPython, mientras que el análisis posterior y la visualización de los datos se llevaron a cabo utilizando herramientas de software especializado. Los resultados obtenidos permitieron validar el modelo teórico de conversión, cuantificar el ruido de cuantización y determinar el rango operativo óptimo del ADC. Asimismo, se identificaron y analizaron las principales fuentes de error y las limitaciones inherentes a la plataforma hardware bajo prueba. Los hallazgos de este estudio proporcionan una base práctica para la selección y uso efectivo de este tipo de convertidores en aplicaciones de instrumentación, adquisición de datos y sistemas embebidos, destacando tanto sus capacidades como sus restricciones de diseño.

## II. ABSTRACT

This paper documents the methodology and main findings of a laboratory practice aimed at characterizing the operation of the analog-to-digital converter (ADC) integrated into a modern architecture microcontroller. The experimental procedure involved the controlled application of known DC voltage signals and variable alternating signals to evaluate key parameters such as linearity, resolution, accuracy, and the dynamic limitations of the conversion system. The data acquisition process was implemented using a MicroPython script, while subsequent data analysis and visualization were performed using specialized software tools. The obtained results allowed for the validation of the theoretical conversion model, quantification of quantization noise, and determination

of the ADC's optimal operational range. Furthermore, the main sources of error and the inherent limitations of the hardware platform under test were identified and analyzed. The findings of this study provide a practical foundation for the selection and effective use of this type of converter in instrumentation, data acquisition, and embedded systems applications, highlighting both its capabilities and its design constraints.

## III. DESARROLLO EXPERIMENTAL

### Datasheet de la Raspberry Pi Pico 2W

El datasheet destaca sus 26 GPIO multifunción (con ADC de 12 bits, UART, SPI e I2C), capacidad de operar entre 1.8V y 5.5V, y modos de bajo consumo energético. Además, su diseño mantiene compatibilidad con las librerías estándar de MicroPython. Esta combinación de potencia, conectividad y eficiencia la posiciona como una solución versátil para prototipado rápido y sistemas integrados avanzados. A continuación se muestra el datasheet descrito:

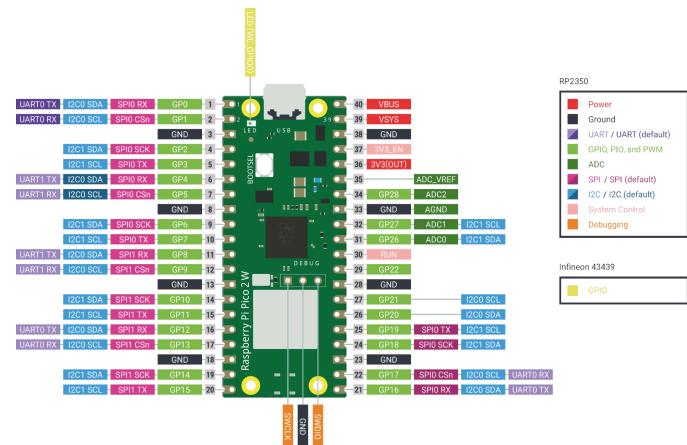


Imagen 1. Datasheet Raspberry Pi Pico 2W

### Parte A-Uso del conversor A/D

Ahora bien, con toda la información técnica, se procede a identificar los terminales ADC en las GPIO del dispositivo, ya determinados se conecta una fuente de tensión DC DF1731SL5A del laboratorio, entre la terminal ADC0 y tierra del dispositivo. En la siguiente imagen se observa que el terminal GP26 es el pin 31 y el terminal GND es el pin 33.

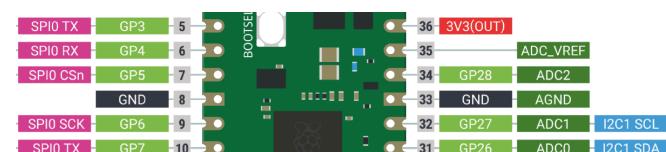


Imagen 2. ADC y GND

A continuación, se muestra cómo sería esa conexión física:



*Imagen 3. ADC y GND a la fuente*

Normalmente, el voltaje de referencia de la Raspberry es de 3,30V, para rectificar este dato, con ayuda de un multímetro, la pinza positiva se coloca en el terminal 3V3(OUT) el cual es el pin 36 y la pinza negativa en el terminal GND el cual es el pin 33, como lo indica la imagen 2. Ahora bien, se configura el multímetro en modo DC con el rango de 20V, como se muestra en la siguiente imagen:



*Imagen 4. Voltaje de referencia*

Posteriormente, con el voltaje de referencia rectificado, se debe tener en cuenta que la fuente se debe mantener entre 0 y 3,3V, ya que si se excede este rango, se podría dañar el microcontrolador. Se implementa el siguiente programa en el entorno de desarrollo integrado Thonny con Python, para poder implementar una conversión analógica-digital.

```
import machine
import utime

ADC_PIN = 26
VREF    = 3.300
PERIOD  = 0.5

adc = machine.ADC(ADC_PIN)

while True:
    raw16 = adc.read_u16()
```

```
    code12 = raw16 >> 4
    volts  = (code12 * VREF) /
4095.0
    print(f"volts:{.4f}"")
    utime.sleep(PERIOD)
```

El conversor analógico-digital (ADC) de los microcontroladores RP2350 (Pico 2W) poseen una resolución de 12 bits, lo que significa que puede entregar valores enteros entre 0 y 4095 (es decir,  $2^{12} - 1$  niveles de cuantización). Sin embargo, la función `read_ul6()` de MicroPython devuelve siempre un valor entero de 16 bits en el rango 0 a 65535. Para lograr esto el valor de 12 bits obtenido del ADC es alineado hacia la izquierda dentro de los 16 bits, llenando con cuatro ceros los bits menos significativos. En otras palabras:

`raw16 = code12 × 16 ≡ code12 << 4`

Donde:

*raw16*: Valor leído directamente con *read u160*

*code12*: Valor real del ADC de 12 bits.

`<<4: Desplazamiento de 4 posiciones a la izquierda en binario`

Para recuperar el valor de 12 bits, se aplica un desplazamiento hacia la derecha:

```
code12 = raw16 << 4
```

donde <<4 equivale a dividir entre  $2^4 = 16$ . Por otro lado, para interpretar correctamente la conversión, se emplea la siguiente ecuación:

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

Para poder entender la conversion analogo-digital se realizaron 6 ejemplos, los cuales se explican a continuación:

### Ejemplo 1

Se escoge el *code12* como el valor medio de la escala 2048 y se convierte en un binario de 12 bits de la siguiente manera:

$$3^{11} = 2049$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces el bit 11 está en 1 y los demás en 0, como se observa a continuación:



Ahora bien, el valor devuelto por `read_ul6()` es:

$$1000\ 0000\ 0000\ 0000 = 2^{15} = 32768$$

Ya que el `code12` es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$\text{code 12} = 1000\ 0000\ 0000$$

Entonces, aplicando la ecuación mencionada anteriormente, el valor ingresado desde la fuente de voltaje sería el siguiente:

$$V = \frac{2048 \times 3,3V}{4095} = 1,6V$$

Teniendo en cuenta, que solo se puede ingresar un decimal en la fuente, se visualiza el resultado de la conversión, activando el *Plotter* en Thonny.

1.5884  
1.5827  
1.5940  
1.5650  
1.5884  
1.6004

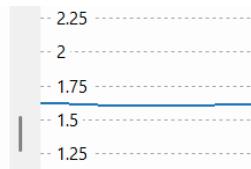


Imagen 5. Voltaje del code12 2048

Ejemplo 2.

Se escoge el `code12` como el valor cuarto de la escala 1024 y se convierte en un binario de 12 bits, de la siguiente manera:

$$2^{10} = 1024$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces el bit 10 está en 1 y los demás en 0, como se observa a continuación:

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	0	0	0

Ahora bien, el valor devuelto por `read_ul6()` es:

$$0100\ 0000\ 0000\ 0000 = 2^{14} = 16384$$

Ya que el `code12` es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$\text{code 12} = 0100\ 0000\ 0000$$

Entonces, aplicando la ecuación, el valor ingresado desde la fuente de voltaje sería el siguiente:

$$V = \frac{1024 \times 3,3V}{4095} = 0,8V$$

Se visualiza el resultado de la conversión:

0.8012  
0.8075  
0.7946  
0.8034  
0.8316  
0.8067  
0.7873

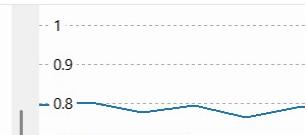


Imagen 6. Voltaje del code12 1024

Ejemplo 3.

Se escoge el `code12` como el valor máximo de la escala 4095 y se convierte en un binario de 12 bits, de la siguiente manera:

$$2^{12} - 1 = 4095$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces todos los bits están en 1, como se observa a continuación:

11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1

Ahora bien, el valor devuelto por `read_ul6()` es:

$$1111\ 1111\ 1111\ 0000 = 2^{15} = 65520$$

Ya que el `code12` es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$\text{code 12} = 1111\ 1111\ 1111$$

Entonces, aplicando la ecuación, el valor ingresado desde la fuente de voltaje sería el siguiente:

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

Se visualiza el resultado de la conversión:

3.1900  
3.1880  
3.1920  
3.1944  
3.2057  
3.1944  
3.2041

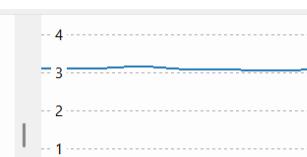


Imagen 7. Voltaje del code12 4095

Ejemplo 4.

Para los siguientes ejemplos, se añaden las siguientes líneas en el código, para poder analizar mejor los datos:

```
print(f"raw16")
```



```
print(f"{code12}")
```

Se escoge el voltaje de la fuente 1,1V y aplicando la ecuación, el valor del *code12* sería el siguiente:

$$code12 = \frac{1,1V \times 4095}{3,3V} = 1365$$

Al convertirlo en un binario de 12 bits, quedaría de la siguiente manera:

$$2^{10} + 2^8 + 2^6 + 2^4 + 2^2 + 2^0 = 1365$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces los bits utilizados como exponentes están en 1 y los demás en 0, como se observa a continuación:

11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1	0	1	0	1

Ahora bien, el valor devuelto por *read\_ul6()* es:

$$0101\ 0101\ 0101\ 0000 = 21840$$

Ya que el *code12* es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$code\ 12 = 0101\ 0101\ 0101$$

Se visualiza el resultado de la conversión:

1.1032  
21909  
1369  
1.1089  
22021  
1376



Imagen 8. Voltaje de la fuente 1,1V

### Ejemplo 5.

Se escoge el voltaje de la fuente 0,4V y aplicando la ecuación, el valor del *code12* sería el siguiente:

$$code12 = \frac{0,4V \times 4095}{3,3V} = 496$$

Al convertirlo en un binario de 12 bits, quedaría de la siguiente manera:

$$2^8 + 2^7 + 2^6 + 2^5 + 2^4 = 496$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces los bits utilizados como exponentes están en 1 y los demás en 0, como se observa a continuación:

11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	1	0	0	0	0

Ahora bien, el valor devuelto por *read\_ul6()* es:

$$1100\ 0001\ 1110\ 0000 = 49632$$

Ya que el *code12* es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$code\ 12 = 1100\ 0001\ 1110$$

Se visualiza el resultado de la conversión:



Imagen 9. Voltaje de la fuente 0,4V

### Ejemplo 6.

Se escoge el voltaje de la fuente 2,5V y aplicando la ecuación, el valor del *code12* sería el siguiente:

$$code12 = \frac{2,5V \times 4095}{3,3V} = 3102$$

Al convertirlo en un binario de 12 bits, quedaría de la siguiente manera:

$$2^{11} + 2^{10} + 2^4 + 2^3 + 2^2 + 2^1 = 3102$$

Los bits se numeran desde 0 (LSB) hasta 11 (MSB), entonces los bits utilizados como exponentes están en 1 y los demás en 0, como se observa a continuación:

11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	1	1	1	1	0

Ahora bien, el valor devuelto por *read\_ul6()* es:

$$1100\ 0001\ 1110\ 0000 = 49632$$

Ya que el *code12* es rellenado con cuatro ceros en los LSB, al aplicar el corrimiento a la derecha quedaría:

$$code\ 12 = 1100\ 0001\ 1110$$

Se visualiza el resultado de la conversión:



2.5127  
49900  
3118  
2.4772  
49196  
3074



Imagen 10. Voltaje de la fuente 2,5V

### Muestreo de señales

El muestreo de señales es una etapa fundamental en el procesamiento de señales analógicas, ya que permite convertir una señal continua en el tiempo en una representación discreta que puede ser procesada por sistemas digitales. En esta fase del laboratorio, se evaluó la capacidad del conversor analógico-digital (ADC) de la Raspberry Pi Pico 2W para muestrear una señal sinusoidal con precisión. Se utilizó un generador de señales para producir una onda senoidal con un componente de DC offset, asegurando que la señal se mantuviera dentro del rango de voltaje admitido por el ADC (0–3.3V). El objetivo principal fue capturar múltiples ciclos de la señal, almacenar los datos en formato .CSV y posteriormente analizar la reconstrucción de la señal muestreada, así como observar el efecto de la frecuencia de la señal en la calidad del muestreo.

En primer lugar se ejecuta el programa *adc.m* dado por el docente en MatLab:

```
f = 625;
T = 1/f;
A = 1.6;
NT = 2;
N = 40;
ts = T/N;

t = 0:ts:(NT*T - ts);
V = A * sin(2*pi*f*t);

t_cont = 0:ts/20:NT*T;
V_cont = A * sin(2*pi*f*t_cont);

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

xlabel('Tiempo
(s)', 'FontSize',20,'FontWeight','bol
d');
ylabel('Amplitud
(V)', 'FontSize',20,'FontWeight','bol
d');
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);
```

¿Qué nombre tiene el archivo .CSV generado por el programa?

La aplicación que captura los datos del conversor analógico a digital (ADC) le asigna de forma automática el nombre "*adc\_capture1*" al archivo .CSV, siguiendo una convención estándar para nombrar que ayuda a identificar y organizar las mediciones. Este formato de nombre facilita la distinción rápida de cada conjunto de datos obtenido, en particular cuando se llevan a cabo varias pruebas o sesiones de medición. Adicionalmente, se garantiza la compatibilidad con diversos sistemas de procesamiento, análisis y almacenamiento al guardar los archivos utilizando nombres sencillos y secuenciales, lo que también previene errores en la gestión de datos. Este método es también provechoso en ambientes de laboratorio o desarrollo, donde es crucial tener un registro claro y organizado de las adquisiciones para que puedan ser analizadas, trazadas y reproducidas después.

Se analiza el resultado y se verifica el número de muestras por periodo, el cual es de 40 y la tasa de muestreo para la señal, es cual es de 40 $\mu$ s, como se muestra a continuación:

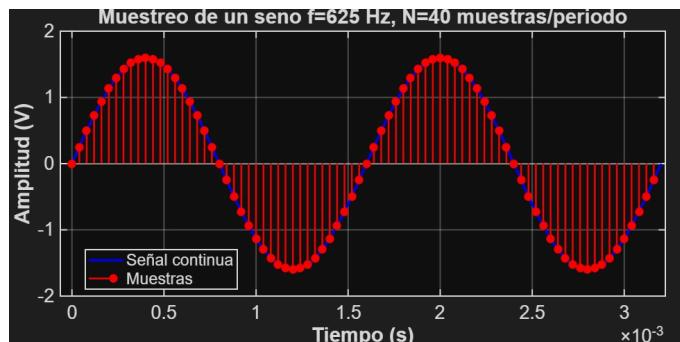


Imagen 11. Muestreo de señal seno 625 Hz

Luego, se carga el programa *ADC\_Sampling.py* en el micrcontrolador.

Posteriormente se ajusta una señal de 3Vpp en el generador de señales, una frecuencia de 625 Hz y con un componente DC=1,6V, ya que por el rango de cuantización, solo pueden emplearse valores positivos. Se verifica con el osciloscopio antes de conectarla al microcontrolador, como se observa en las siguiente imagen:

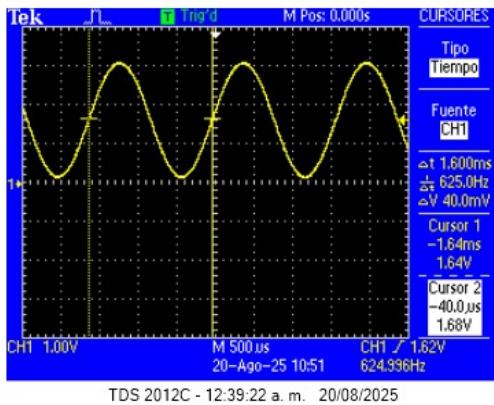


Imagen 12. Señal seno en osciloscopio

Una vez observada la señal en el osciloscopio se procedió a almacenar la información de la señal en formato .CSV, como se mencionó anteriormente, para su correspondiente reconstrucción, cabe resaltar que a mayor frecuencia es más complicado para el código captar los datos desde el Raspberry, ya que generan una pequeña interferencia, para lograr apreciar esto, se tomaron 3 pruebas diferentes cada una con frecuencias diferentes desde 625 bajando hasta 20 Hz:

#### Frecuencia de 20 Hz

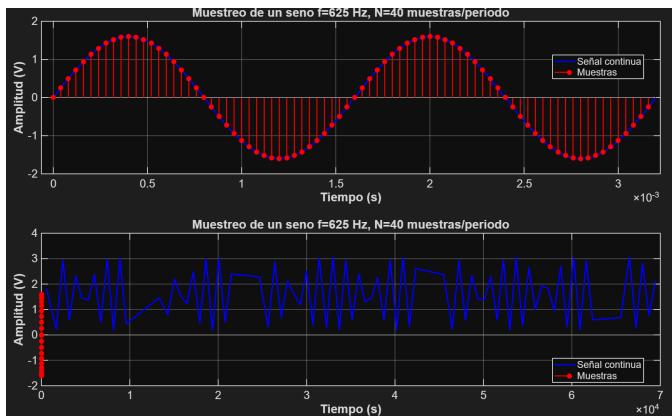


Imagen 13. Muestreo con 625 Hz

#### Frecuencia de 50 Hz

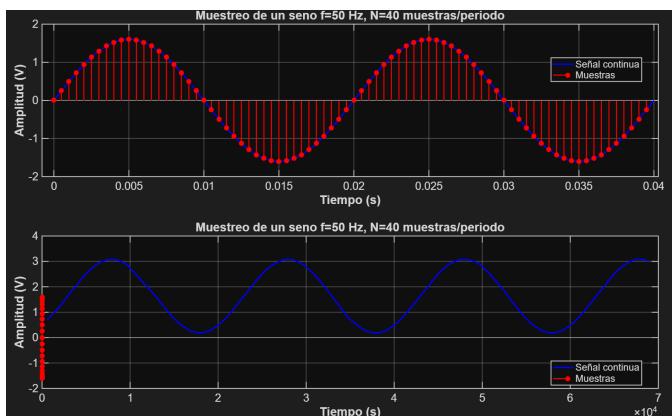


Imagen 14. Muestreo con 50 Hz

#### Frecuencia de 20 Hz

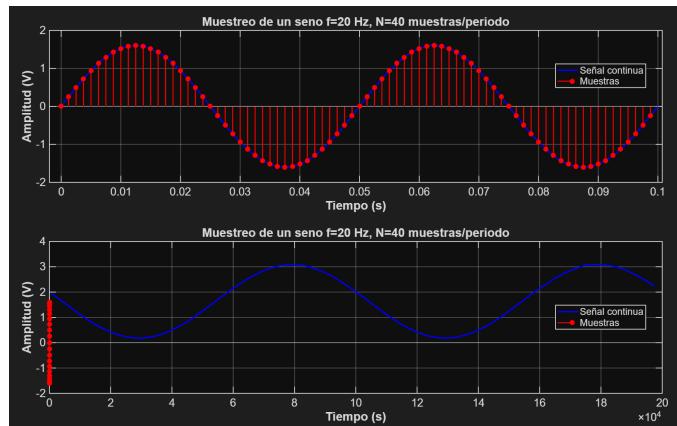


Imagen 15. Muestreo con 20 Hz

#### Análisis estadístico

Se procedió a correr el programa suministrado por el docente llamado *sampling\_2* el cual se puede consultar en el repositorio, este código realiza un proceso de muestreo y análisis estadístico de señales analógicas utilizando el conversor ADC de la Raspberry Pi Pico 2W. Primero solicita al usuario los nombres de los archivos donde se almacenarán las muestras y el histograma. Posteriormente, toma un número definido de muestras desde el pin ADC, convierte cada lectura cruda en voltaje y registra el tiempo asociado. De manera simultánea, el código calcula de forma incremental la media y la desviación estándar mediante el algoritmo de *Welford*, además de generar un histograma de las lecturas con una resolución de 12 bits. Finalmente, los resultados se guardan en archivos de texto y se muestran en consola, permitiendo tanto el análisis numérico como la visualización de la distribución de las señales medidas.

Además, el programa se ajustó para tomar alrededor de 10.000 muestras en los archivos almacenados. A continuación se seleccionaron 5 valores de voltaje distintos, los cuales se procesan con el código y generan las siguientes mediciones:

#### Voltaje 0.8V

---

Ciclo de muestreo completado con 10000 muestras.  
Estadísticas del ciclo:  
Total de muestras: 10000  
Media: 0.83689 V  
Desviación Estándar: 0.01290 V  
Histograma de lecturas (res. 12 bits):  
{1116: 1, 1118: 3, 1119: 1, 930: 1, 932: 1, 933: 1}  
Datos del histograma guardados en 'h0.8'  
Programa finalizado.

Imagen 16. Valores de muestreo con 0.8V



### Voltaje 1.4 V

```
Ciclo de muestreo completado con 10000 muestras.  
Estadísticas del ciclo:  
    Total de muestras: 10000  
    Media: 1.39377 V  
    Desviación Estándar: 0.01243 V  
Histograma de lecturas (res. 12 bits):  
{1670: 3, 1671: 1, 1672: 7, 1673: 1, 1674: 3, 1675:  
Datos del histograma guardados en 'h1.4'  
Programa finalizado.
```

Imagen 17. Valores de muestreo con 1.4 V

### Voltaje 1.8V

```
Ciclo de muestreo completado con 10000 muestras.  
Estadísticas del ciclo:  
    Total de muestras: 10000  
    Media: 1.78472 V  
    Desviación Estándar: 0.01036 V  
Histograma de lecturas (res. 12 bits):  
{2171: 10, 2172: 4, 2173: 8, 2174: 8, 2176: 5, 2177:  
Datos del histograma guardados en 'h1.8'  
Programa finalizado.
```

Imagen 18. Valores de muestreo con 1.8V

### Voltaje 2.4V

```
Ciclo de muestreo completado con 10000 muestras.  
Estadísticas del ciclo:  
    Total de muestras: 10000  
    Media: 2.41007 V  
    Desviación Estándar: 0.01010 V  
Histograma de lecturas (res. 12 bits):  
{3006: 27, 3007: 29, 3008: 24, 3009: 17, 3010: 14,  
Datos del histograma guardados en 'h2.4'  
Programa finalizado.
```

Imagen 19. Valores de muestreo con 2.4V

### Voltaje 3.0 V

```
Ciclo de muestreo completado con 10000 muestras.  
Estadísticas del ciclo:  
    Total de muestras: 10000  
    Media: 2.89400 V  
    Desviación Estándar: 0.01138 V  
Histograma de lecturas (res. 12 bits):  
{3568: 19, 3569: 23, 3570: 26, 3571: 30, 3572: 43  
Datos del histograma guardados en 'h3.0'  
Programa finalizado.
```

Imagen 20. Valores de muestreo con 3.0V

Ahora bien, se procede a realizar el siguiente código en MatLab, el cual a partir de los datos del voltaje permita hallar tanto la media como la desviación estándar:

```
voltajes = x0_8.Voltaje_V;
```

```
% Calcular media  
media = mean(voltajes);
```

```
% Calcular desviación estándar  
(poblacional)  
desv_std = std(voltajes, 1); % el  
"1" indica varianza poblacional  
  
fprintf("Resultados calculados en  
MATLAB:\n");  
fprintf("Media: %.5f V\n", media);  
fprintf("Desviación Estándar: %.5f  
V\n", desv_std);
```

El código anterior generó los siguientes valores para cada voltaje, recordando que son los experimentales ejecutados con ayuda del csv:

### Voltaje 0.8V

```
Resultados calculados en MATLAB:  
Media: 0.83689 V  
Desviación Estándar: 0.01290 V
```

Imagen 21. Valores de muestreo con 0.8V en Matlab

### Voltaje 1.4 V

```
Resultados calculados en MATLAB:  
Media: 1.39377 V  
Desviación Estándar: 0.01242 V
```

Imagen 22. Valores de muestreo con 1.4V en Matlab

### Voltaje 1.8V

```
Resultados calculados en MATLAB:  
Media: 1.78472 V  
Desviación Estándar: 0.01036 V
```

Imagen 23. Valores de muestreo con 1.8V en Matlab

### Voltaje 2.4V

```
Resultados calculados en MATLAB:  
Media: 2.41007 V  
Desviación Estándar: 0.01010 V
```

Imagen 24. Valores de muestreo con 2.4V en Matlab

*Voltaje 3.0 V*

### Resultados calculados en MATLAB:

Media: 2.89399 V

Desviación Estándar: 0.01138 V

Imagen 25. Valores de muestreo con 3.0V en Matlab

Finalmente la información capturada de las pruebas realizadas tanto por el programa en Thonny como por el programa en Matlab se muestran en la siguiente tabla:

Test	Vin (DC)	Media	Desviación Estándar	Nombre de los archivos
1	0.8v	0.83	0.0129	0.8dc, h0.8
2	1.4v	1.39	0.0124	1.4dc, h1.4
3	1.8v	1.78	0.0103	1.8dc, h1.8
4	2.4v	2.41	0.0101	2.4dc, h2.4
5	3v	2.89	0.0113	3.0dc, h3.0

Tabla 1. Valores de programa en Thonny y MatLab

Como puede verse en la tabla anterior, los resultados son idénticos, lo que demuestra la coherencia de las mediciones efectuadas. Además, como se había señalado, se creó un archivo de frecuencias a partir de los valores adquiridos. Este archivo hace posible el análisis de la distribución de las lecturas del ADC (12 Bits) y la observación precisa del comportamiento estadístico de la señal muestreada. Para cada voltaje, se utilizó el siguiente código en MatLab:

```

x = h3_0.Valor_ADC_12bits;
y = h3_0.Frecuencia;
figure;
bar(x, y, 'FaceColor', [0 0.447 0.741]);
xlabel('Valor ADC (12 bits)');
ylabel('Frecuencia');
title('Histograma de valores ADC');
grid on;

```

El código anterior genera los siguientes histogramas en frecuencia de cada valor de voltaje:

*Voltaje 0.8V*

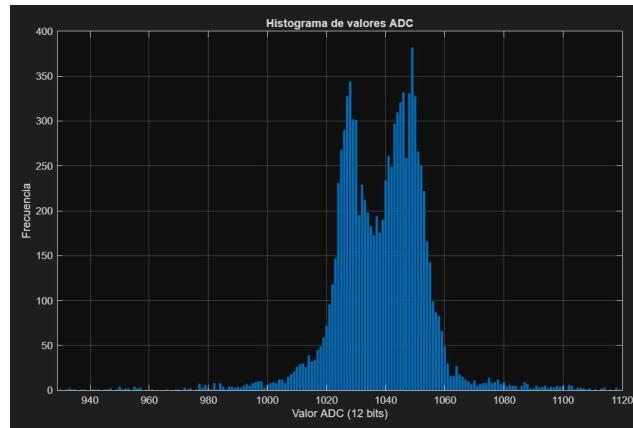


Imagen 26. Histograma de voltaje 0.8V

*Voltaje 1.4 V*

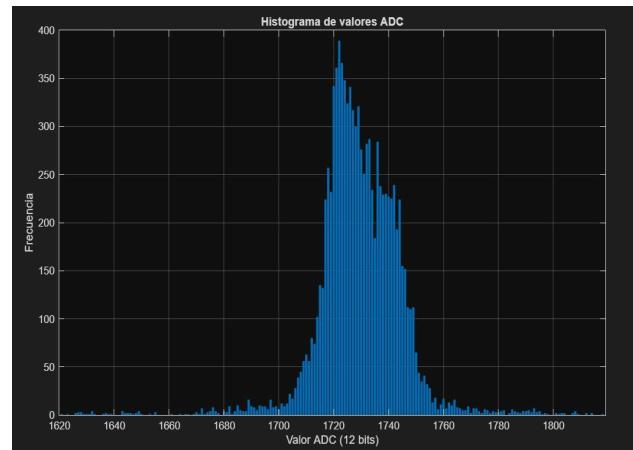


Imagen 27. Histograma de voltaje 1.4V

*Voltaje 1.8V*

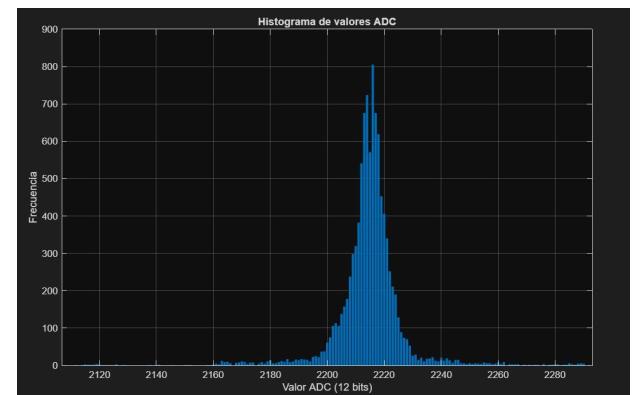


Imagen 28. Histograma de voltaje 1.8V

*Voltaje 2.4V*

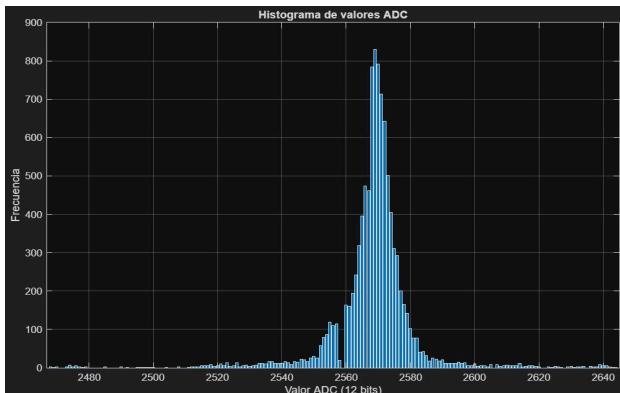


Imagen 29. Histograma de voltaje 2.4V

*Voltaje 3.0 V*

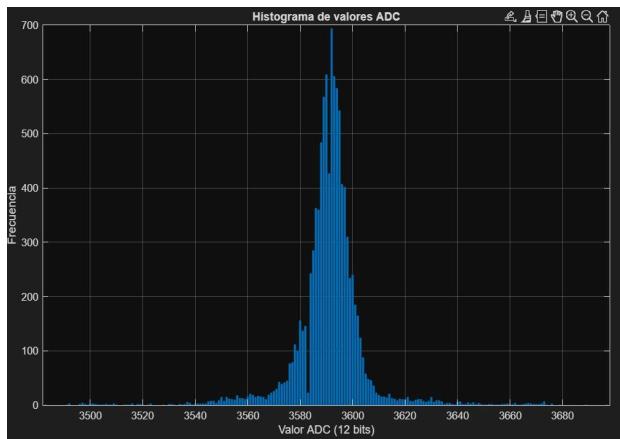


Imagen 30. Histograma de voltaje 3.0V

Los histogramas previos exhiben cómo se distribuyen las frecuencias de los valores digitales que el convertidor analógico-digital (ADC) de 12 bits obtiene al muestrear señales con un rango de entre 0 y 3,3V. El valor medio del ADC muestra un crecimiento notable conforme se incrementa el voltaje de entrada, desde 1040 (0.77 V) hasta 3590 (2.66 V), pasando por los siguientes valores: 1725 (1.28 V), 2220 (1.64 V) y 2570 (1.90 V). Por lo general, las distribuciones son simétricas en los rangos altos y medios, lo cual señala una estabilidad adecuada y una escasa variabilidad en la señal.

En el primer caso, por otro lado (cerca de 0.77 V), se puede observar una distribución con dos picos pronunciados, lo que podría indicar que la señal de entrada es inestable o tiene un ruido considerable, probablemente a causa de interferencias o restricciones en la resolución para tensiones bajas. La frecuencia máxima de cada pico se eleva a medida que el voltaje sube, lo que muestra una concentración más alta de los datos y una dispersión menor. En el último gráfico, a pesar de que el punto máximo continúa siendo alto, se percibe una expansión de la distribución. Esto podría señalar efectos de saturación o no linealidad del ADC. En general, estos resultados muestran que el sistema opera con precisión en la

mayoría del rango, sin embargo, es necesario poner especial atención en zonas de baja tensión para asegurar que las mediciones sean confiables.

#### IV. ANÁLISIS DE RESULTADOS

- El comportamiento del conversor analógico-digital (ADC) de 12 bits que viene incorporado en la Raspberry Pi Pico 2W fue caracterizado gracias a un desarrollo experimental. Se obtuvo una conversión de valores digitales a voltaje y se aplicaron tensiones DC en el pin GP 26, utilizando MicroPython. Se usó una tensión de referencia de 3.3V. Los resultados revelaron una gran coherencia entre los valores teóricos y los que se midieron, con desviaciones menores que pueden atribuirse al ruido de la medición y a las restricciones de la cuantización.
- Para mantener la señal en el rango positivo del ADC, se empleó una señal sinusoidal con 3Vpp y un offset de 1.6V durante la fase de muestreo de señales variables. Se notó que, conforme la frecuencia de la señal se reducía, la recolección de datos se tornaba más estable; en cambio, a frecuencias más elevadas aparecen distorsiones e interferencias, lo cual indica que el microcontrolador tiene restricciones en su velocidad de muestreo para señales rápidas.
- Luego, se llevó a cabo un estudio estadístico con 10,000 muestras por cada valor de voltaje DC evaluado (0.8V, 1.4V, 1.8V, 2.4V y 3.0V). Los resultados confirmaron la fiabilidad del sistema, dado que las medias y las desviaciones estándar en Thonny y MatLab son idénticas. Las desviaciones estándar se mantuvieron entre 0.010 y 0.013V, lo que es un indicativo de una buena exactitud en las mediciones.
- Los histogramas creados a partir de los valores ADC de 12 bits mostraron distribuciones unimodales y simétricas en la mayor parte de los voltajes, salvo con 0.8V, que mostró una distribución bimodal, probablemente debido a inestabilidad o ruido en tensiones bajas. Los histogramas presentaron una dispersión más baja a medida que el voltaje se incrementa, lo que significó que los valores se concentraron más alrededor de la media.

En general, los resultados indican que el ADC de la Raspberry Pi Pico 2W proporciona un rendimiento fiable en casi todo su rango operativo; no obstante, se aconseja tener cuidado en los extremos del mismo para prevenir mediciones alteradas por saturación o ruido.



## V. CONCLUSIONES

## VII. REPOSITORIO DE GITHUB

<https://github.com/Vallentincita/Comunicaci-n-digital-TEL-A->

1. El ADC de 12 bits de la Raspberry Pi Pico 2W funciona correctamente y su conversión analógica-digital se ajusta al modelo teórico esperado.
2. El sistema ofrece mayor precisión y estabilidad en el rango medio de voltajes (1.4V – 2.4V), con desviaciones estándar bajas (~ 0.01 V).
3. En voltajes bajos (0.8V) se observó mayor susceptibilidad al ruido, generando distribuciones bimodales e inestabilidad en las mediciones.
4. El microcontrolador presenta limitaciones de muestreo a altas frecuencias, mostrando distorsiones en señales de 625 Hz, por lo que resulta más adecuado para señales de baja frecuencia o de tipo continua.
5. Los resultados estadísticos obtenidos en Thonny/MicroPython y MatLab fueron consistentes, validando la confiabilidad del proceso de adquisición y análisis de datos.
6. El ADC ofrece un desempeño confiable y lineal en la mayor parte de su rango, siendo recomendable operar en la zona central y evitar los extremos, donde aumentan los efectos de ruido, saturación e inestabilidad.

## VI. REFERENCIAS

- [1] Raspberry Pi Foundation, *Raspberry Pi Pico 2 W Datasheet*,
- [2] R. Long, “Measuring Small Voltages With Pi Pico ADC and Comparison With ...”, *Instructables*
- [3] Analog Devices, “Understanding Noise, ENOB, and Effective Resolution in Analog-to-Digital Converter Circuits