COMUNICACIÓN DIGITAL - PROCESAMIENTO DE SEÑALES (agosto de 2025)

Fernando Javier Riaño Rios
Est.fernando.riano@unimilitar.edu.co

Resumen -En este trabajo se analizaron diez señales diferentes, incluyendo dos señales seno, dos triangulares, dos cuadradas y cuatro pulsos con diferentes ciclos útiles. Algunas de las señales, además de los pulsos, contenían componentes de corriente continua (DC). Se utilizó un osciloscopio digital para capturar estas señales, generando archivos en formato CSV con los datos tanto en el dominio del tiempo como en el de la frecuencia. Para cada señal, se obtuvieron imágenes del espectro en ambos dominios y se compararon con los resultados obtenidos mediante programas en Matlab. En Matlab, se reconstruyeron las señales utilizando los primeros armónicos, tanto teóricos como experimentales, y se visualizaron en diferentes dominios para observar sus comportamientos. Este análisis permitió comprender mejor las características y el comportamiento de las diferentes señales en aplicaciones de comunicaciones digitales.

Abstract – This work analyzes ten different signals, including two sinusoidal, two triangular, two square, and four pulse signals with varying duty cycles. Some of the signals, in addition to pulses, contained DC components. Using a digital oscilloscope, the signals were captured, and CSV files were generated containing data in both time and frequency domains. Spectrum images were obtained for each signal and compared with results from MATLAB reconstructions. MATLAB programs were developed to reconstruct the signals based on the first harmonic components, both theoretical and experimental, and to visualize their behavior in different domains. This comprehensive analysis enhances the understanding of signal characteristics in digital communication systems.

Índice de Términos - Atenuación, Decibelios (dB), Frecuencia, muestreo, Potencia, Voltaje.

La comunicación digital es un campo que requiere un entendimiento profundo del comportamiento de las señales que se transmiten y reciben en los sistemas electrónicos. Conocer cómo se comportan las señales en ambos dominio, tiempo y frecuencia, ayuda a diseñar sistemas más eficientes y confiables. En este trabajo, se analizaron varias señales periódicas y pulsos, que son comunes en aplicaciones de comunicaciones digitales. La captura y análisis de estas señales en un osciloscopio digital permitieron obtener datos precisos y visualizaciones del espectro, facilitando su comparación con modelos teóricos en Matlab. Este enfoque combina la medición experimental con herramientas de simulación para profundizar en el estudio de las señales digitales y su procesamiento.

I. DESARROLLO DE LOS EXPERIMENTOS

1. Parámetros generales del instrumento.

Al activar la función FFT en el osciloscopio y ajustar los controles de tiempo, se puede visualizar claramente los primeros armónicos de la señal en el espectro. El archivo CSV generado contiene las amplitudes de cada frecuencia y sus correspondientes valores en el dominio de la frecuencia. Además de los datos del muestreo, el archivo suele incluir información adicional, estos datos corresponden a los parámetros técnicos y metadatos como la resolución en frecuencia, el número de puntos utilizados en la FFT, la escala del eje de frecuencia y parámetros técnicos del análisis.

Se analiza los parámetros técnicos del instrumento encontrando; La tasa de muestreo del osciloscopio empleado en la adquisición de las señales es aproximadamente 500 kHz (500,000 muestras por segundo). el número de puntos almacenados en la adquisición fue de 2500 puntos, que, con el intervalo de muestreo mencionado, da una duración total del análisis de aproximadamente 5 ms.

2. Señal 1: seno

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz						
Vp	1						
SEÑAL	DC	ARMONICA	f (HZ)	A			
				d.B	Vrms	VP(v)	TEORICO
Sen 1	0	1	800	-2,99	0,709	1,002	1

Imagen de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia.

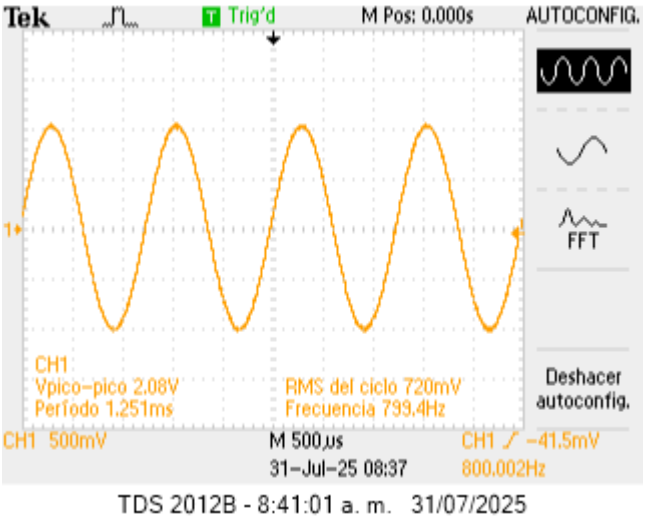


Ilustración 1 señal generada y exportada del osciloscopio - en tiempo

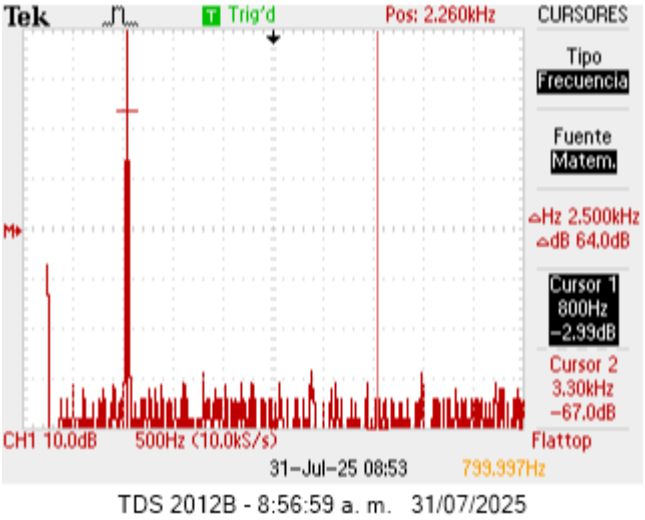


Ilustración 2 señal generada y exportada del osciloscopio - en espectro de frecuencia

De acuerdo a estos datos experimentales tomados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/100000:2*T;
A1=1;
v1=A1*sin(2*pi*f*t);
plot(t,v1);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal sinusoidal');
```

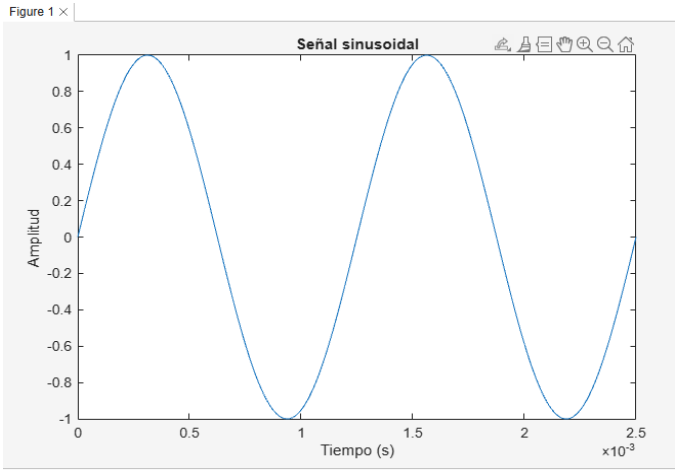


Ilustración 3 grafica generada por Matlab - tiempo.

Graficamos en Python los datos exportados desde el osciloscopio con el siguiente código:

```
import pandas as pd
import matplotlib.pyplot as plt
archivo = 'DOMINIOTIEMPO1.csv'
# Leer archivo desde fila 18, sin encabezado, con codificación Latin-1
datos = pd.read_csv(archivo, skiprows=17, header=None, encoding='latin1')
tiempo = datos.iloc[:, 3]  # Columna 4 en MATLAB (índice 3)
voltaje = datos.iloc[:, 4]  # Columna 5 en MATLAB (índice 4)
plt.plot(tiempo, voltaje, color='#FFA500')
plt.title('Dominio del tiempo SENO2')
plt.xlabel('Tiempo (s)')
plt.ylabel('Voltaje (V)')
plt.grid(True)
plt.show()
```

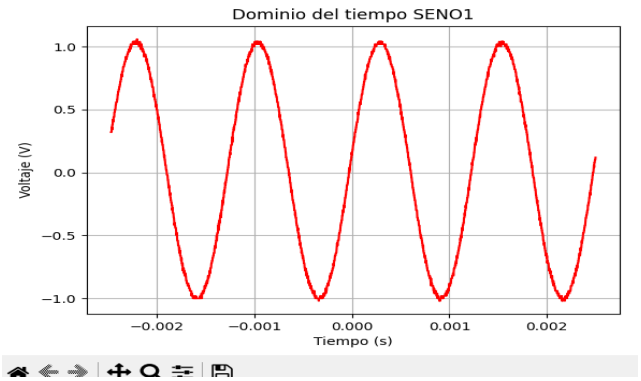


Ilustración 4 grafica generada en python

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DATDOMINIOTIEMPOSEN1.csv');
opts.DataLines = [18 Inf];  % Solo desde la fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres de variables en la fila 18

% Leer los datos
datos = readtable('DATDOMINIOTIEMPOSEN1.csv', opts);

% Convertir las columnas correctas (4 y 5) a vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje

% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

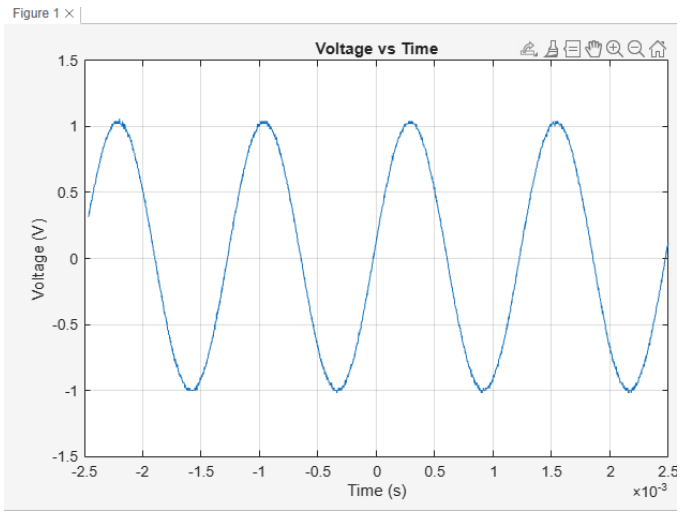


Ilustración 5 grafica generada por Matlab con los datos exportados.

De acuerdo a estos datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('FFTSEN1.csv');
opts.DataLines = [18 Inf];           % Solo desde la fila 18 en adelante
opts.Delimiter = ',';                % Delimitador por coma
opts.VariableNamesLine = 0;          % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('FFTSEN1.csv', opts);
% Convertir las columnas correctas (4 y 5) a vectores
f = datos{:,4};     % Columna 4 = Frecuencia (Hz)
mag = datos{:,5};   % Columna 5 = Magnitud (en dB o voltios)
% Graficar espectro de frecuencia
plot(f, mag, 'b', 'LineWidth', 1.5);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB o V)');
title('Espectro de Frecuencia - Señal Senoidal');
grid on;
xlim([0 max(f)]);
ylim([min(mag)-5, max(mag)+5]);
```

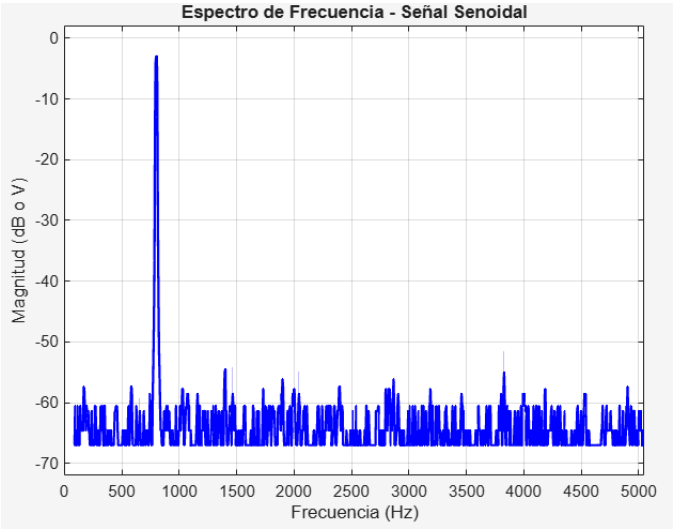


Ilustración 6 grafica generada por Matlab con los datos exportados-frecuencia.

Se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo calcular el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s)
f = 1/T;                  % Frecuencia (Hz) → 800 Hz
Fs = 1e5;                 % Frecuencia de muestreo (100 kHz)
```

```
t = 0:1/Fs:2*T;          % Tiempo total de 2
ciclos (0 a 0.0025 s)
A1 = 1;                  % Amplitud de la
señal
% Generar señal senoidal
v1 = A1 * sin(2*pi*f*t);
% Cálculo de la FFT
N = length(v1);                  % Número
total de muestras
Y = fft(v1);                     %
Transformada rápida de Fourier
Y_mag = abs(Y)/N;                % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1); % Duplicar
componentes (menos DC y Nyquist)
% Escala de frecuencia
f_axis = (0:N-1)*(Fs/N);         % Vector de
frecuencias asociado a la FFT

% Convertir magnitud a decibelios
Y_dB = 20*log10(Y_mag + eps);    % eps evita
log(0) y errores numéricos

% Graficar el espectro (solo la mitad
positiva)
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.5);
xlabel('Frecuencia (Hz)');
ylabel('Amplitud (dBV)');
title('Espectro en Frecuencia en dB - Señal
Senoidal');
grid on;
xlim([0 5000]);                  % Rango de
frecuencias (visualización)
ylim([-80 10]);                  % Rango de
amplitud (escala dBV)
```

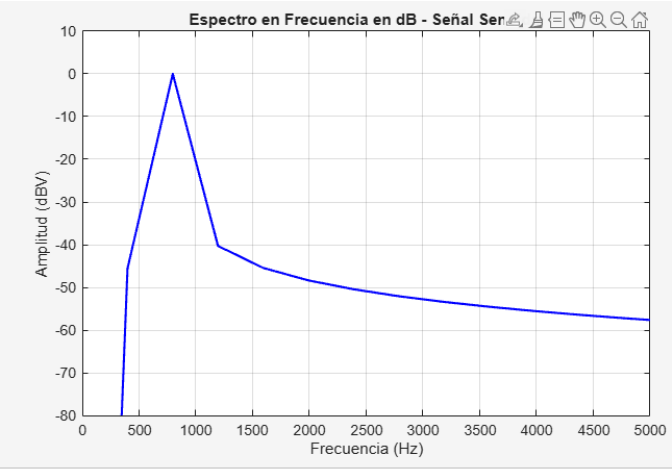


Ilustración 7 grafica generada en Matlab FFT de seno1

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que ambas gráficas son coherentes entre sí ya que representan el mismo espectro de una señal senoidal pura de 800 Hz y 1 V de amplitud. La gráfica en dBV (0 dBV =1V) destaca que solo hay una componente fuerte (la fundamental), estas gráficas confirman que la señal es casi perfectamente senoidal, sin componentes armónicas significativas.

Ahora se crea un programa que permite reconstruir la señal en el tiempo a partir de los datos ya obtenidos al aplicar la FFT, esto aplicando la IFFT.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla.

Con el siguiente código aplicamos la IFFT para reconstruirla.

```
% Parámetros de la señal
T = 0.00125;             % Periodo de la
señal (s)
f = 1/T;                 % Frecuencia de la
señal (Hz) → 800 Hz
Fs = 1e5;                % Frecuencia de
muestreo (100 kHz)
t = 0:1/Fs:2*T;          % Tiempo total de 2
ciclos (0 a 0.0025 s)
A1 = 1;                  % Amplitud de la
señal
% Señal original (onda seno)
v1 = A1 * sin(2*pi*f*t);
% FFT de la señal
N = length(v1);          % Número de muestras
Y = fft(v1);             % FFT de la señal

% Reconstrucción por IFFT
v1_rec = ifft(Y, 'symmetric'); % Reconstruir
señal aplicando IFFT

% Gráfica final de la señal reconstruida
t_ms = t * 1000;         % Convertir a
milisegundos para graficar

figure;
plot(t_ms, v1_rec, 'b', 'LineWidth', 1.5);
xlabel('Tiempo (ms)');
ylabel('Amplitud (V)');
title('Señal reconstruida con IFFT (dominio
del tiempo)');
grid on;
```

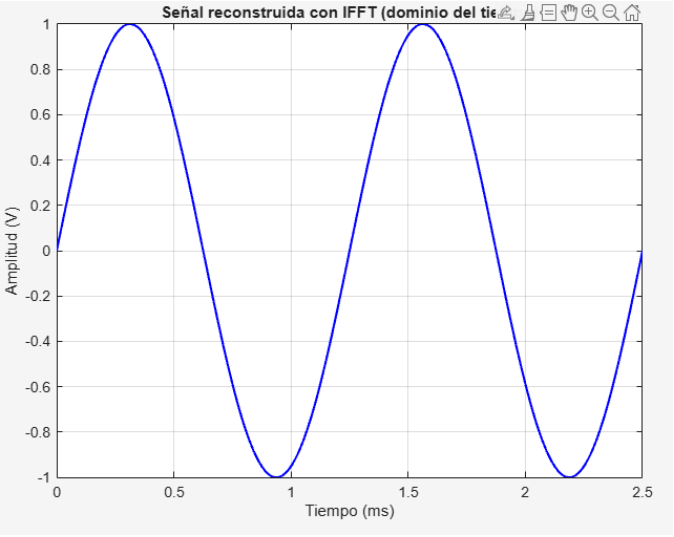


Ilustración 8 Grafica Generada Por Matlab- IFFT seno 1.

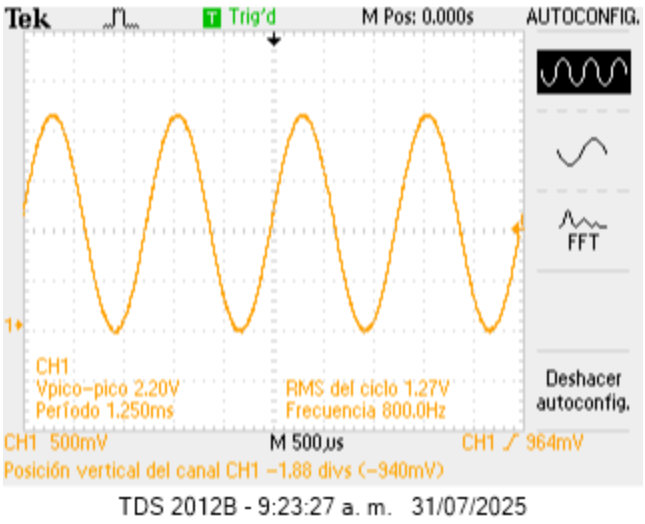


Ilustración 9 señal generada y exportada del osciloscopio - en tiempo

3. Señal 2: seno con DC=1

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz					
Vp	1					
SEÑAL	DC	ARMONICA	f (HZ)	A		
				d.B	Vrms	VP(v)
Sen 1	1	1	800	-2,19	0,777	1,099

Imagen de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia.

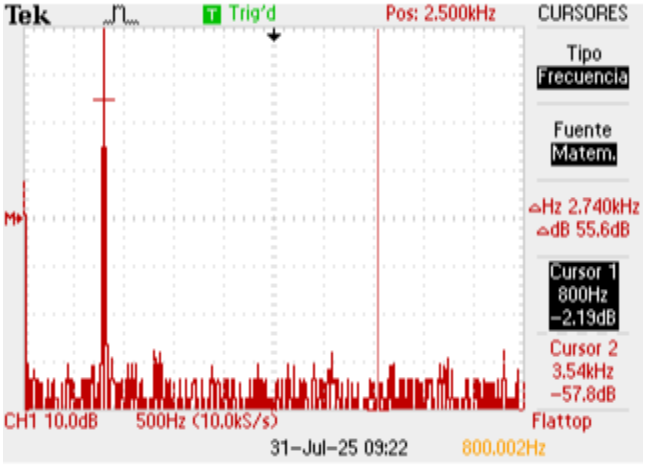


Ilustración 10 señal generada y exportada del osciloscopio - en frecuencia.

De acuerdo a estos datos experimentales tomados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/100000:2*T;
A1=1;
v1=1+A1*sin(2*pi*f*t);
plot(t,v1)
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal sinusoidal con DC=1');
```

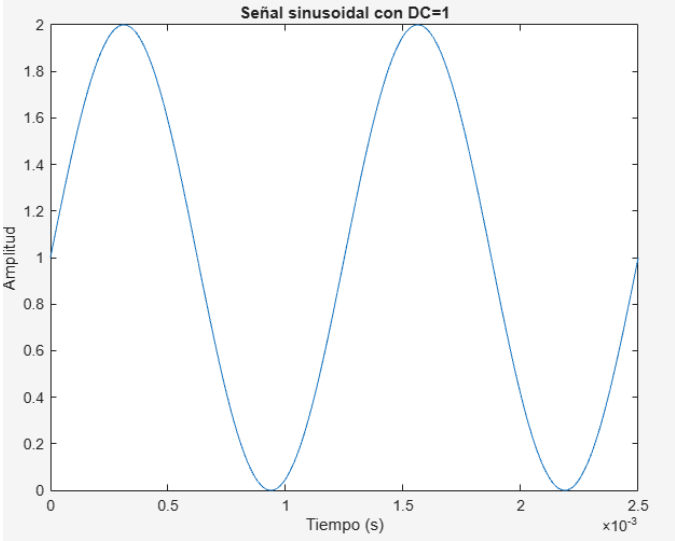


Ilustración 11 grafica generada por Matlab - tiempo.

Graficamos en Python los datos exportados desde el osciloscopio.

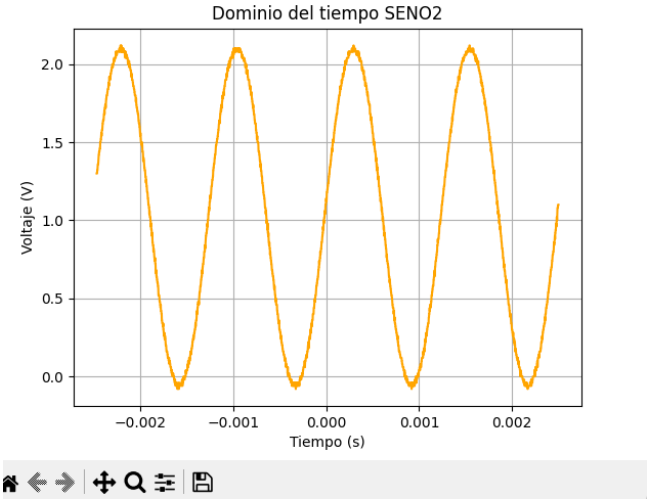


Ilustración 12 grafica generada en Python

Como esta señal tiene un DC=1, Se observa que se desplaza 1v, desplazándola desde el origen e iniciando en 1v hasta Vmax 2v y Vmin en 0v.

De acuerdo a los datos experimentales tomados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOTIEMPOSEN2.csv');
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18

% Leer los datos
datos = readtable('DOMINIOTIEMPOSEN2.csv',
opts);

% Convertir las columnas correctas (4 y 5) a
vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje

% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

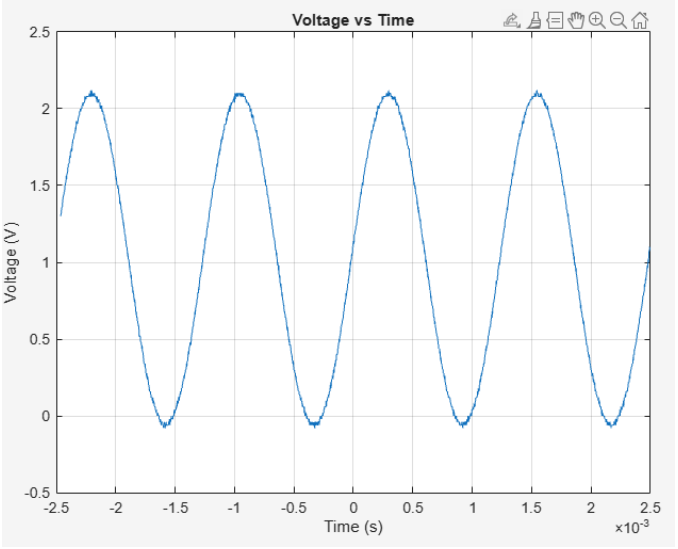


Ilustración 13 grafica generada por Matlab con los datos exportados- tiempo.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOFRECUENCIASEN2.cs
v');
opts.DataLines = [18 Inf];          % Solo
desde la fila 18 en adelante
opts.Delimiter = ',';               %
Delimitador por coma
opts.VariableNamesLine = 0;         % No hay
nombres de variables en la fila 18
% Leer los datos
```

```
datos =
readtable('DOMINIOFRECUENCIASEN2.csv', opts);
% Extraer columnas (frecuencia y magnitud en
dB o amplitud)
f = datos{:,4};       % Columna 4 =
Frecuencia (Hz)
mag = datos{:,5};     % Columna 5 = Magnitud
(dB o V)
% Graficar
plot(f, mag, 'r', 'LineWidth', 1.5);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB o V)');
title('Espectro de Frecuencia - Señal
Senoidal');
grid on;
xlim([0 max(f)]);
ylim([min(mag)-5, max(mag)+5]); % Ajuste
automático del eje Y
```

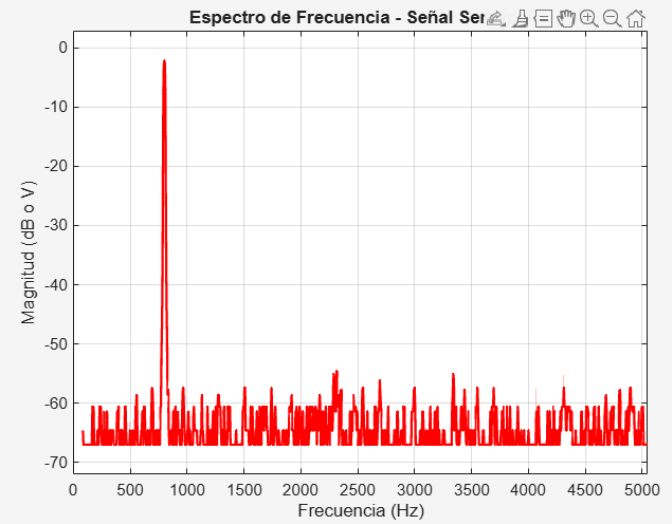


Ilustración 14 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

Se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo calcular el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s) →
1/800 Hz
f = 1/T;                  % Frecuencia de la
señal
Fs = 1e5;                 % Frecuencia de
muestreo (100 kHz)
t = 0:1/Fs:2*T;           % Tiempo (2 ciclos)
A1 = 1;                   % Amplitud de la
señal
v1 = 1 + A1 * sin(2*pi*f*t); % Señal con
componente DC = 1 V
% Aplicar la FFT
N = length(v1);                % Número de
muestras
Y = fft(v1);                   % FFT
Y_mag = abs(Y)/N;              % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1); % Doblar
las componentes (excepto DC y Nyquist)

% Crear eje de frecuencia
f_axis = (0:N-1)*(Fs/N);       % Eje de
frecuencia

% Graficar espectro de magnitudes (solo mitad
positiva)
plot(f_axis(1:N/2), Y_mag(1:N/2), 'b',
'LineWidth', 1.5);
xlabel('Frecuencia (Hz)');
ylabel('Amplitud (V)');
title('Espectro de frecuencia (FFT) - Señal
seno con DC');
grid on;
xlim([0 5000]);                % Límite de
visualización
```

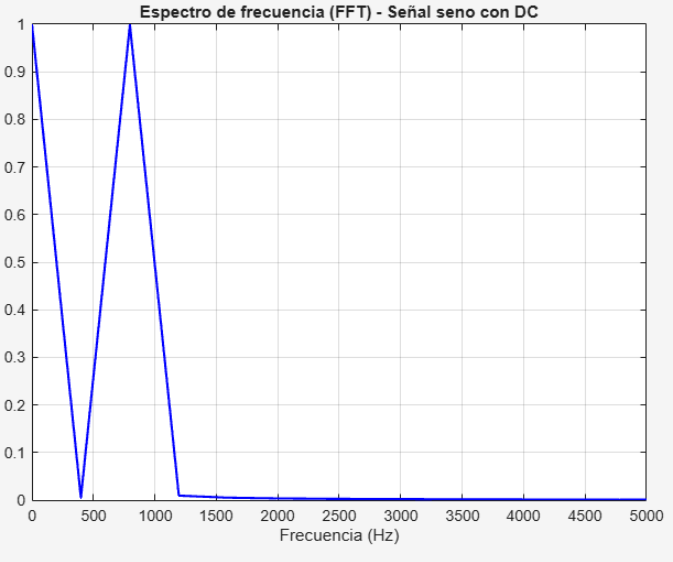


Ilustración 15 grafica generada en Matlab FFT de seno2

La gráfica muestra un pico significativo en aproximadamente 800 Hz. Esto corresponde a la frecuencia fundamental de la señal sinusoidal de entrada, que originalmente tenías en 800 Hz, la gráfica muestra un pico en 0 Hz, al tener una señal con DC (valor constante) también aparece un pico en esa frecuencia.

Ahora se crea un programa que permite reconstruir la señal en el tiempo a partir de los datos ya obtenidos al aplicar la FFT, esto aplicando la IFFT.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .CSV solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla.

Con el siguiente código aplicamos la IFFT para reconstruirla.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s) → 1/800 Hz
f = 1/T;                  % Frecuencia de la señal
Fs = 1e5;                 % Frecuencia de muestreo (100 kHz)
t = 0:1/Fs:2*T;           % Tiempo (2 ciclos)
A1 = 1;
v1 = 1 + A1 * sin(2*pi*f*t);
% Aplicar FFT
N = length(v1);
Y = fft(v1);
Y_mag = abs(Y)/N;
Y_mag(2:end-1) = 2*Y_mag(2:end-1);
% Crear eje de frecuencia
f_axis = (0:N-1)*(Fs/N);
% Graficar espectro de magnitudes (solo mitad positiva)
figure;
plot(f_axis(1:N/2), Y_mag(1:N/2), 'b', 'LineWidth', 1.5);
xlabel('Frecuencia (Hz)');
ylabel('Amplitud (V)');
title('Espectro de frecuencia (FFT) - Señal seno con DC');
grid on;
xlim([0 5000]);
% Reconstrucción usando IFFT
v1_rec = ifft(Y, 'symmetric');  % IFFT de la señal
% Graficar señal reconstruida
figure;
plot(t*1000, v1_rec, 'r', 'LineWidth', 1.5);
xlabel('Tiempo (ms)');
ylabel('Amplitud (V)');
title('Señal reconstruida con IFFT');
grid on;
```

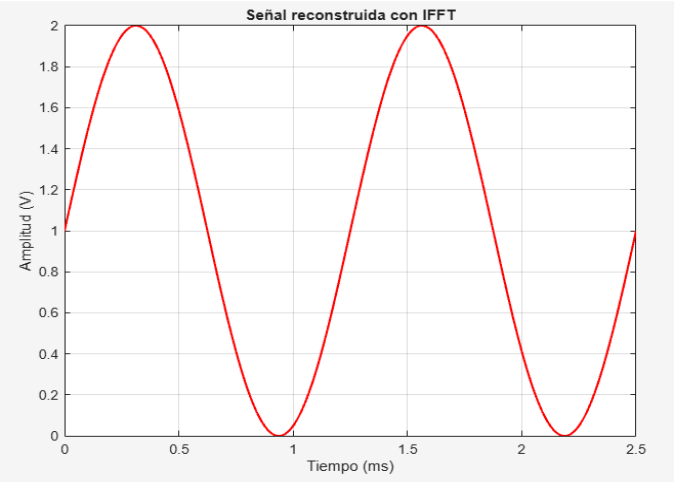


Ilustración 16 Grafica Generada Por Matlab- IFFT seno 2 con DC=1

La gráfica muestra una onda sinusoidal suave, lo que indica que la señal reconstruida mantiene la forma original de una señal armónica simple. La señal oscila alrededor de 0 V, con un pico cercano a 2 V y un valor mínimo cerca de 0 V, formando una onda sinusoidal clásica con DC=1..

4. señal: triangular 1 con DC=0

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz						
Vp	1						
SEÑAL	DC	ARMONICA	f (HZ)	A			
				d.B	Vrms	VP(v)	TEORICO
TRIANGULAR1	0	1	800	-4,59	0,5895	0,8337	0,81057
		3	2400	-23,8	0,0646	0,0913	0,09006
		5	4000	-32,6	0,0234	0,0332	0,03242
		7	5600	-38,6	0,0117	0,0166	0,01654
		9	7200	-43	0,0071	0,0100	0,01001
		11	8800	-47	0,0045	0,0063	0,0067

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia.

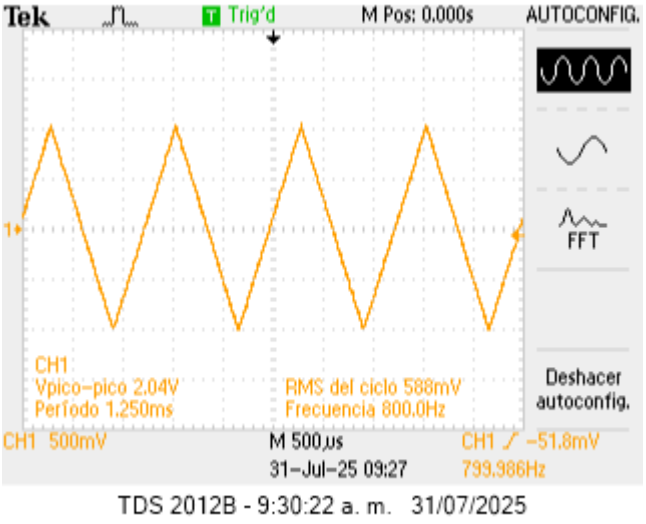


Ilustración 17 señal generada y exportada del osciloscopio - en tiempo

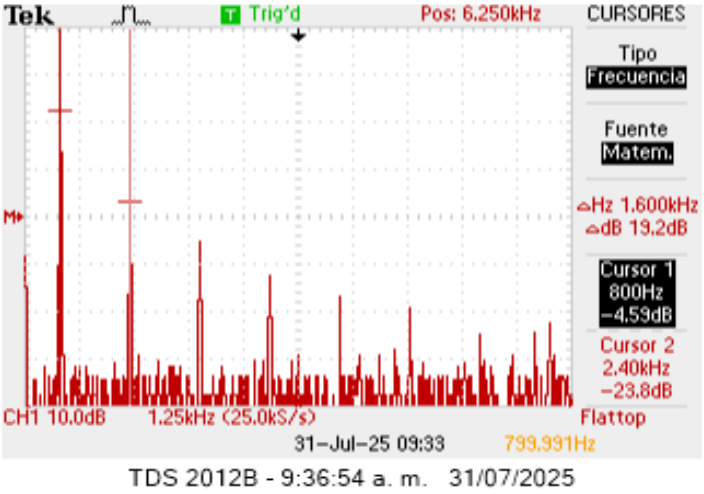


Ilustración 18 señal generada y exportada del osciloscopio - en espectro de frecuencia.

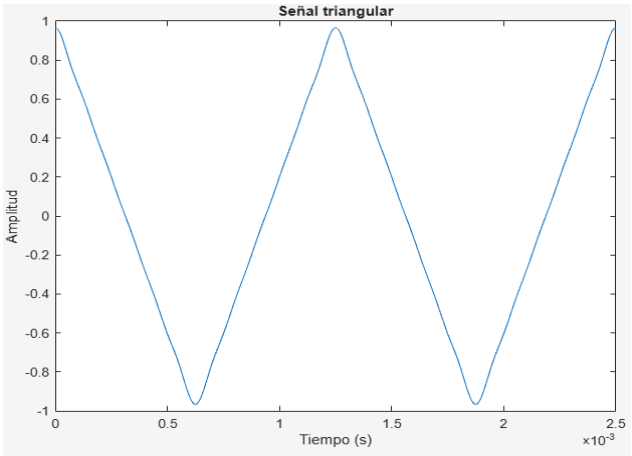


Ilustración 19 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

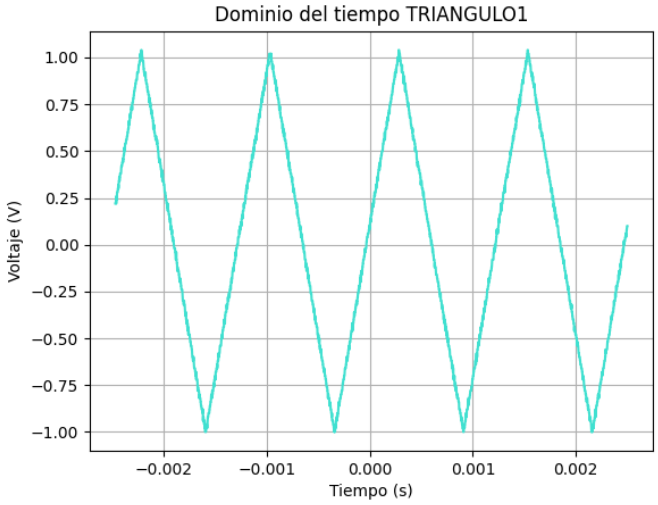
```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
A1=0.810;
A3=0.090;
A5=0.032;
A7=0.016;
A9=0.010;
A11=0.0066;
v1=A1*cos(2*pi*f*t);
v2=A3*cos(2*pi*3*f*t);
v3=A5*cos(2*pi*5*f*t);
v4=A7*cos(2*pi*7*f*t);
v5=A9*cos(2*pi*9*f*t);
v6=A11*cos(2*pi*11*f*t);
VT=v1+v2+v3+v4+v5+v6;
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal triangular');
```

SERIES DE FOURIER

Es una función periódica par (simétrica), Solo contiene armónicos impares (frecuencias múltiplos impares del fundamental).
La amplitud de los armónicos decrece con el cuadrado del orden ($1/n^2$), lo que la hace suave y con pocos armónicos relevantes.
Fórmula general:

$$x(t) = a_0 + \sum_{n=1,3,5,\ldots}^{\infty} a_n \cos\left(\frac{2\pi n t}{T}\right)$$

Graficamos en Python.



De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
A1=0.833;
A3=0.091;
A5=0.033;
A7=0.016;
A9=0.010;
A11=0.0063;
v1=A1*cos(2*pi*f*t);
v2=A3*cos(2*pi*3*f*t);
v3=A5*cos(2*pi*5*f*t);
v4=A7*cos(2*pi*7*f*t);
v5=A9*cos(2*pi*9*f*t);
v6=A11*cos(2*pi*11*f*t);
VT=v1+v2+v3+v4+v5+v6;
```

```
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal triangular');
```

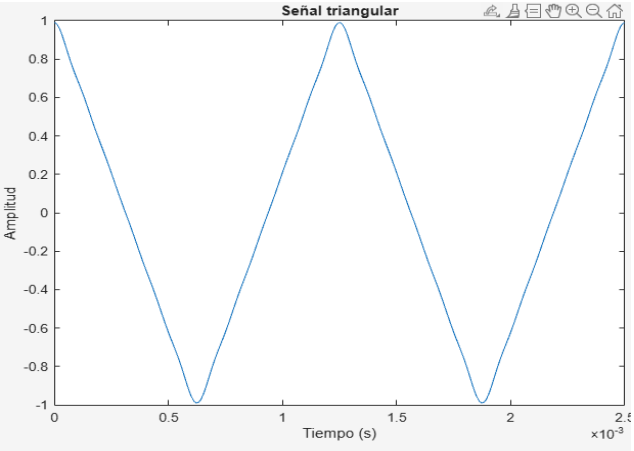


Ilustración 20 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal triangular en el dominio del tiempo, pero hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de generación de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), la mayor diferencia se encuentra en la armónica uno que para el dato teórico es de 0.810v y para el dato experimental es de 0.833v, por lo que esta grafica tiene una leve mayor amplitud.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando fila 18
opts =
detectImportOptions('DOMINIOTIEMPOTRI1.csv');
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0;
datos = readtable('DOMINIOTIEMPOTRI1.csv',
opts);
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

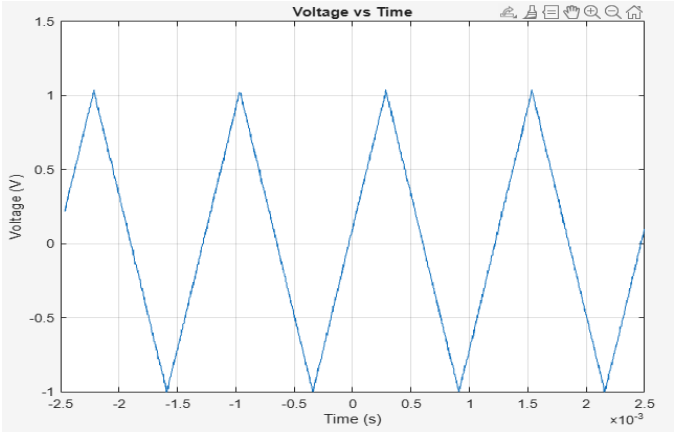


Ilustración 21 grafica generada por Matlab con los datos exportados-tiempo.

De acuerdo a estos datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOFRECUENCIATRI1.cs
v');
opts.DataLines = [18 Inf];
opts.Delimiter = ',';% Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18
datos =
readtable('DOMINIOFRECUENCIATRI1.csv', opts);
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

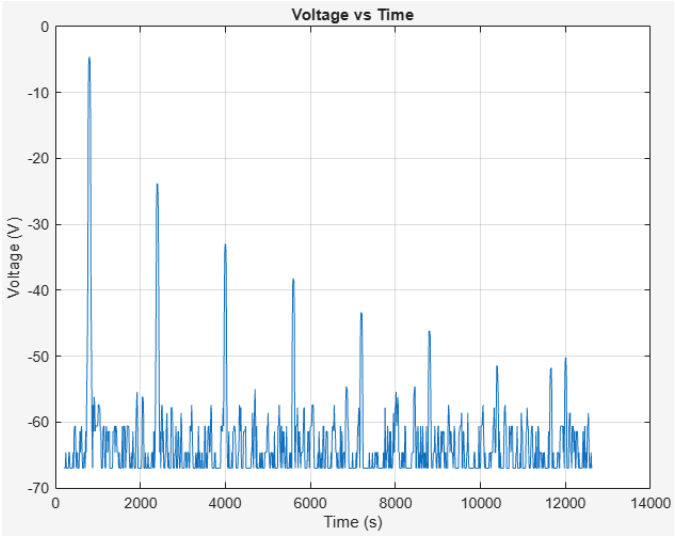


Ilustración 22 grafica generada por Matlab (dB-S) con los datos exportados-frecuencia.

Se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo calcular el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;              % Periodo de la señal (s)
f = 1/T;                  % Frecuencia fundamental
(Hz)
Fs = 1e6;                 % Frecuencia de muestreo
(1 MHz) → suficiente para 11*f
t = 0:1/Fs:2*T;           % Tiempo (dos periodos)
% Armónicos (suma de cosenos impares)
A1 = 0.833;
A3 = 0.091;
A5 = 0.033;
A7 = 0.016;
A9 = 0.010;
A11 = 0.0063;
v1 = A1 * cos(2*pi*f*t);
v2 = A3 * cos(2*pi*3*f*t);
v3 = A5 * cos(2*pi*5*f*t);
v4 = A7 * cos(2*pi*7*f*t);
v5 = A9 * cos(2*pi*9*f*t);
v6 = A11 * cos(2*pi*11*f*t);
% Señal total (triangular aproximada)
VT = v1 + v2 + v3 + v4 + v5 + v6;
% Calcular la FFT
N = length(VT);               % Número de muestras
Y = fft(VT);                  % Transformada rápida
de Fourier
Y_mag = abs(Y)/N;             % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  % Ajustar
amplitud (salvo DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);      % eps para
evitar log(0)
% Crear eje de frecuencia
f_axis = (0:N-1)*(Fs/N);           % Vector de
frecuencias
% Graficar espectro (solo mitad positiva)
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b', 'LineWidth',
1.5);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de frecuencia - Señal triangular
(FFT)');
grid on;
xlim([0 15000]); % Zoom hasta 15 kHz para
visualizar armónicos
ylim([-80 5]);   % Rango típico para amplitudes en
dB
```

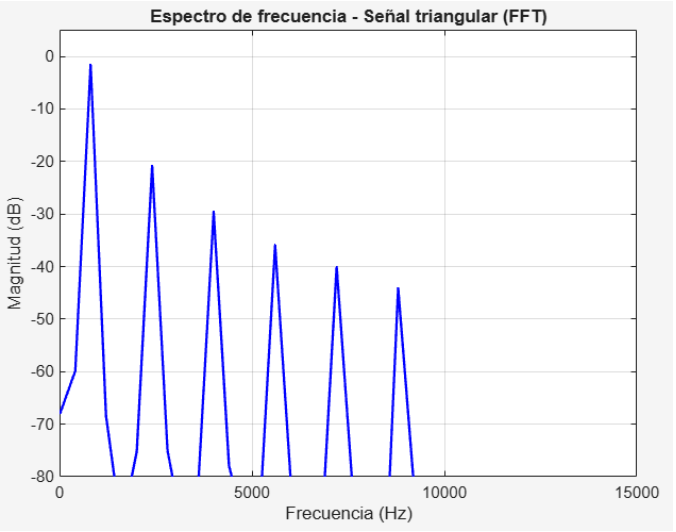


Ilustración 23 grafica generada en Matlab FFT de trian1

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que ambas gráficas son coherentes entre sí ya que los gráficos representan una señal triangular generada como suma de armónicos impares (frecuencias: 1f, 3f, 5f, 7f, 9f, 11f), las componentes espectrales aparecen a frecuencias específicas (múltiplos impares del fundamental), lo cual es típico de una señal triangular. Cuando la señal fue generada con más armónicos, el espectro mostró mayor riqueza, Cuando provino de datos experimentales, aparecieron armónicos menos exactos (ruido) y el espectro fue mínimamente más disperso.

Cuando provino de datos experimentales, aparecieron armónicos adicionales (ruido) y el espectro fue más disperso.

Ahora se crea un programa que permite reconstruir la señal en el tiempo a partir de los datos ya obtenidos al aplicar la FFT, esto aplicando la IFFT.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
% Parámetros de la señal
T = 0.00125;              % Periodo de la
señal (s)
f = 1/T;                  % Frecuencia
fundamental (Hz)
```

```
Fs = 1e6;                  % Frecuencia de
muestreo (1 MHz)
t = 0:1/Fs:2*T;            % Tiempo para dos
periodos
% Componentes armónicas de la señal
triangular
A1 = 0.833;
A3 = 0.091;
A5 = 0.033;
A7 = 0.016;
A9 = 0.010;
A11 = 0.0063;

% Composición de la señal triangular con
armónicos impares
v1 = A1 * cos(2*pi*f*t);
v2 = A3 * cos(2*pi*3*f*t);
v3 = A5 * cos(2*pi*5*f*t);
v4 = A7 * cos(2*pi*7*f*t);
v5 = A9 * cos(2*pi*9*f*t);
v6 = A11 * cos(2*pi*11*f*t);
% Señal total aproximada
VT = v1 + v2 + v3 + v4 + v5 + v6;
% Aplicar FFT e IFFT para reconstruirla
Y = fft(VT);
VT_rec = ifft(Y, 'symmetric');
% Graficar la señal reconstruida
t_ms = t * 1000;  % Convertir tiempo a
milisegundos
figure;
plot(t_ms, VT_rec, 'r', 'LineWidth', 1.5);
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida con IFFT');
grid on;
```

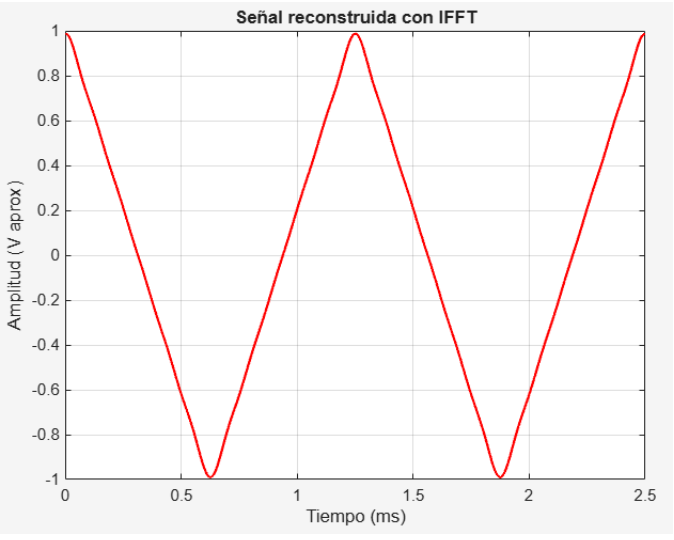


Ilustración 24 Grafica Generada Por Matlab- IFFT triangular1

Se logra recuperar la señal en tiempo original. El resultado es una forma muy parecida a la señal triangular ideal, pero no perfecta, ya que faltan más armónicos para afinar la forma.

5. Señal 4: triangular 2, DC=1.5

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz						
Vp	1						
SEÑAL	DC	ARMONICA	f(HZ)	A			TEORICO
				d.B	Vrms	VP(v)	8*V/(n*pi)^2
T2	1,5	1	800	-4,19	0,6173	0,8730	0,8106
		3	2400	-23,4	0,0676	0,0956	0,0901
		5	4000	-32,2	0,0245	0,0347	0,0324
		7	5600	-38,6	0,0117	0,0166	0,0165
		9	7200	-43,4	0,0068	0,0096	0,0100
		11	8800	-45,08	0,0056	0,0079	0,0067

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia.

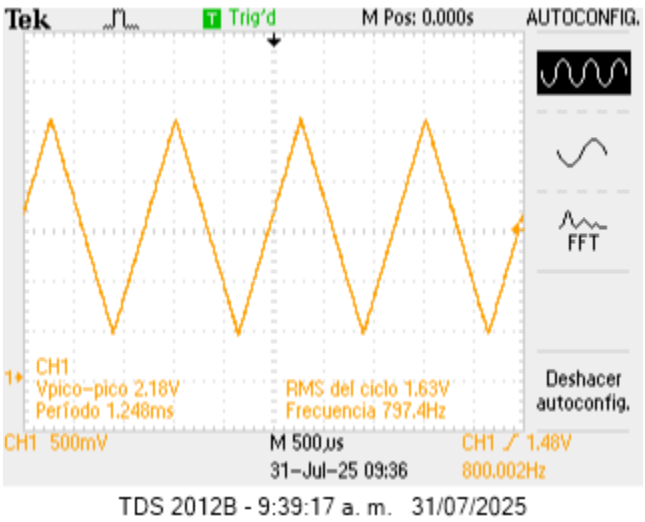


Ilustración 25 señal generada y exportada del osciloscopio - en tiempo

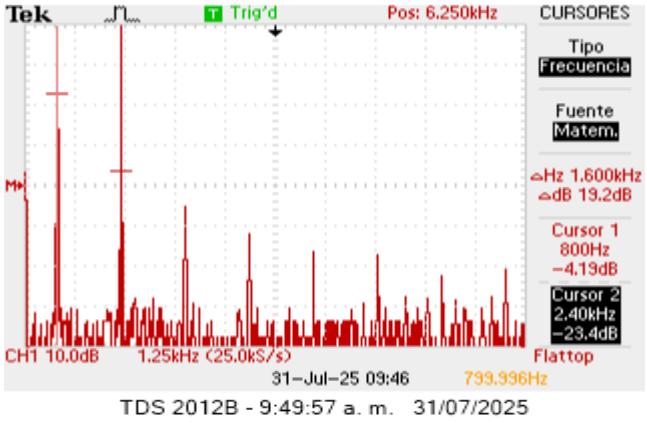


Ilustración 26 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=0.810;
A3=0.090;
A5=0.032;
A7=0.016;
A9=0.010;
A11=0.006;
v1=A1*cos(2*pi*f*t);
v2=A3*cos(2*pi*3*f*t);
v3=A5*cos(2*pi*5*f*t);
v4=A7*cos(2*pi*7*f*t);
v5=A9*cos(2*pi*9*f*t);
v6=A11*cos(2*pi*11*f*t);

VT=1.5+(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal triangular con DC 1.5');
```

SERIES DE FOURIER

Es una función periódica par (simétrica), Solo contiene armónicos impares (frecuencias múltiplos impares del fundamental).
La amplitud de los armónicos decrece con el cuadrado del orden ($1/n^2$), lo que la hace suave y con pocos armónicos relevantes.
Fórmula general:

$$x(t) = a_0 + \sum_{n=1,3,5,\ldots}^{\infty} a_n \cos\left(\frac{2\pi n t}{T}\right)$$

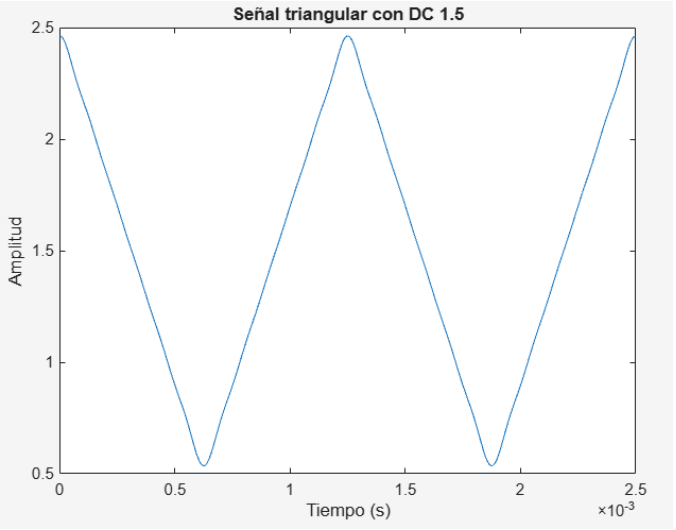


Ilustración 27 grafica generada por Matlab datos teóricos – tiempo.

El valor mínimo está alrededor de 0.5 V y el máximo en 2.5 V, lo que indica una amplitud pico a pico de aproximadamente. La señal está desplazada hacia arriba, centrada en 1.5 V, lo que confirma la presencia de una componente DC de 1.5 V agregada a la señal.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=0.873;
A3=0.095;
A5=0.034;
A7=0.016;
A9=0.009;
A11=0.007;
v1=A1*cos(2*pi*f*t);
v2=A3*cos(2*pi*3*f*t);
v3=A5*cos(2*pi*5*f*t);
v4=A7*cos(2*pi*7*f*t);
v5=A9*cos(2*pi*9*f*t);
v6=A11*cos(2*pi*11*f*t);

VT=1.5+(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal triangular con DC 1.5');
```

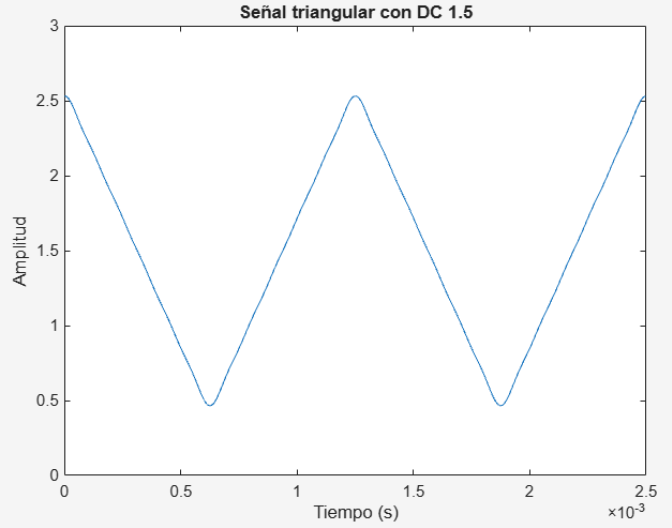


Ilustración 28 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal triangular en el dominio del tiempo, pero hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), la mayor diferencia se encuentra en la armónica uno que para el dato teórico es de 0.810v y para el dato experimental es de 0.873v, por lo que esta grafica tiene una leve mayor amplitud.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOTIEMPOTRI2.csv');
opts.DataLines = [18 Inf];  % Solo desde la fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOTRI2.csv', opts);
% Convertir las columnas correctas (4 y 5) a vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

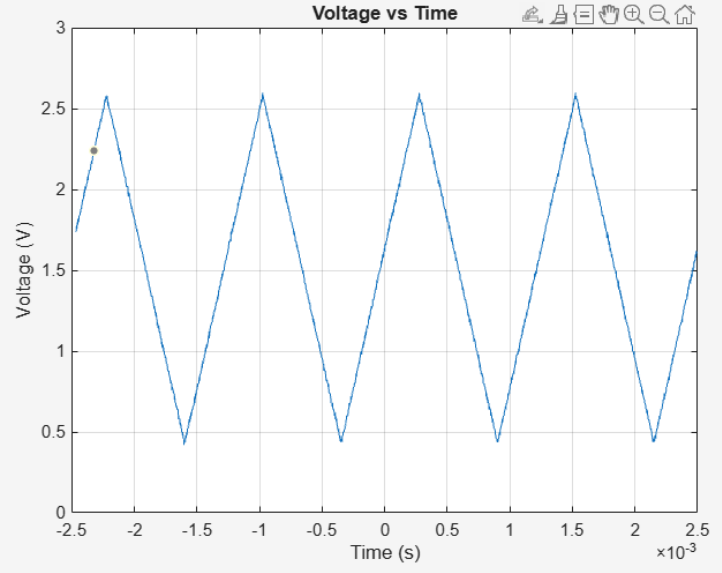


Ilustración 29 grafica generada por Matlab con los datos exportados-tiempo.

Es una señal triangular periódica claramente definida, La señal no está centrada en cero, lo que indica presencia de componente DC positiva.

De acuerdo a estos datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOFRECUENBCIATRI2.csv');
opts.DataLines = [18 Inf];          % Solo desde la fila 18 en adelante
opts.Delimiter = ',';               % Delimitador por coma
opts.VariableNamesLine = 0;         % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOFRECUENBCIATRI2.csv', opts);
% Extraer columnas (frecuencia y magnitud en dB)
f = datos{:,4};   % Columna 4 = Frecuencia (Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud (dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
```

```
title('Espectro de Frecuencia - Señal
Triangular (FFT)');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); %
Ajuste automático del eje Y
```

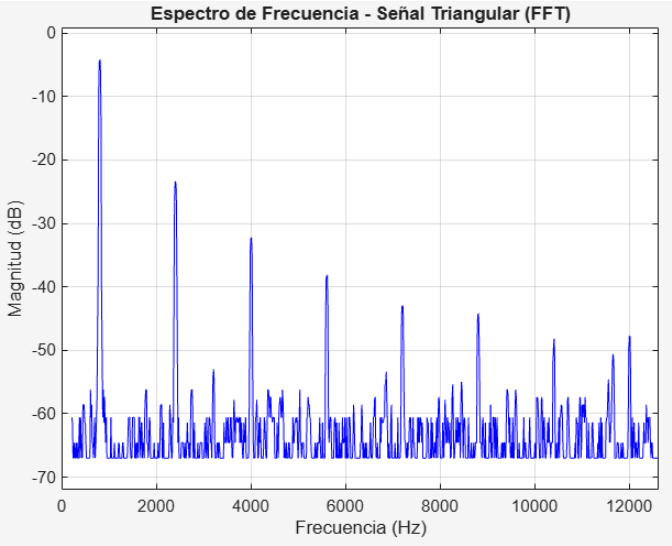


Ilustración 30 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

Ahora se crea un programa que permite reconstruir la señal en el tiempo a partir de los datos ya obtenidos al aplicar la FFT, esto aplicando la IFFT.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
% Parámetros de la señal
T = 0.00125;              % Periodo de la
señal (s)
f = 1/T;                  % Frecuencia
fundamental (Hz)
Fs = 1e6;                 % Frecuencia de
muestreo (1 MHz) → suficiente para 11*f
t = 0:1/Fs:2*T;           % Tiempo (dos
periodos)
% Armónicos (suma de cosenos impares)
A1 = 0.833;
A3 = 0.091;
A5 = 0.033;
A7 = 0.016;
A9 = 0.010;
A11 = 0.0063;
v1 = A1 * cos(2*pi*f*t);
v2 = A3 * cos(2*pi*3*f*t);
v3 = A5 * cos(2*pi*5*f*t);
v4 = A7 * cos(2*pi*7*f*t);
v5 = A9 * cos(2*pi*9*f*t);
v6 = A11 * cos(2*pi*11*f*t);
% Señal total (triangular aproximada)
VT=1.5+(v1+v2+v3+v4+v5+v6);
% Calcular la FFT
N = length(VT);               % Número de
muestras
Y = fft(VT);                  % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;             % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  % Ajustar
amplitud (salvo DC y Nyquist)
Y_dB = 20*log10(Y_mag + eps);      % eps para
evitar log(0)

% Reconstrucción de la señal con IFFT
VT_rec = ifft(Y, 'symmetric');     %
Reconstrucción de la señal
% Graficar sólo la señal reconstruida
figure;
plot(t*1000, VT_rec, 'r', 'LineWidth', 1.5);
% t en milisegundos
xlabel('Tiempo (ms)');
ylabel('Amplitud (V)');
title('Señal triangular reconstruida
(IFFT)');
grid on;
```

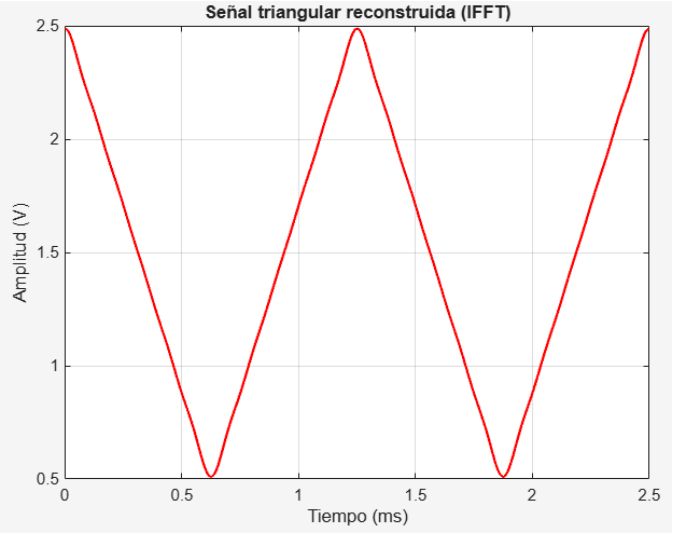


Ilustración 31 Grafica Generada Por Matlab- triangular2

Se logra recuperar la señal en tiempo original. El resultado es una triangular con DC=1.5.

6. señal: cuadrada 1 con DC=0

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

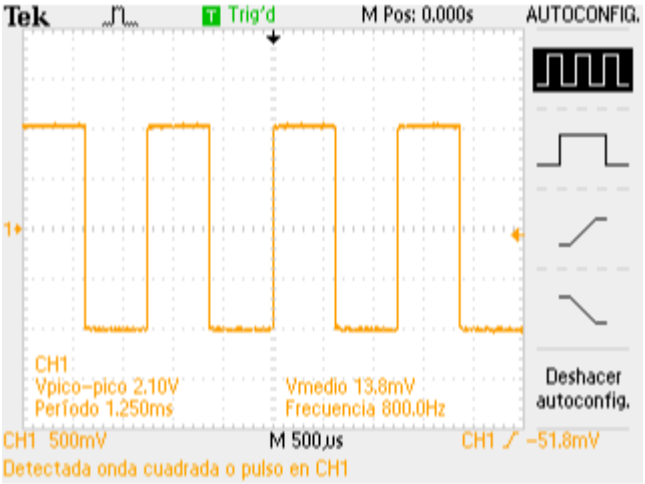
posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

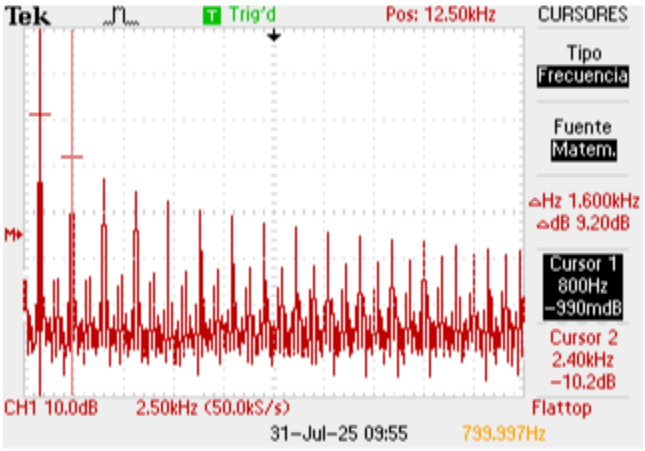
f	800 Hz						
Vp	1						
SEÑAL	DC	ARMONICA	f (HZ)	A			TEORICO
				d.B	Vrms	VP(v)	4*V/n*pi
CUAD1	0	1	800	-0,999	0,8914	1,2606	1,27324
		3	2400	-10,2	0,3090	0,4370	0,42441
		5	4000	-15	0,1778	0,2515	0,25465
		7	5600	-17,8	0,1288	0,1822	0,18189
		9	7200	-19,8	0,1023	0,1447	0,14147
		11	8800	-21,8	0,0813	0,1150	0,11575

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:



TDS 2012B - 9:54:25 a. m. 31/07/2025

Ilustración 32 señal generada y exportada del osciloscopio - en tiempo



TDS 2012B - 9:58:41 a. m. 31/07/2025

Ilustración 33 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=1.273;
A3=0.424;
A5=0.254;
A7=0.181;
A9=0.141;
A11=0.115;
v1=A1*sin(2*pi*f*t);
v2=A3*sin(2*pi*3*f*t);
v3=A5*sin(2*pi*5*f*t);
v4=A7*sin(2*pi*7*f*t);
v5=A9*sin(2*pi*9*f*t);
v6=A11*sin(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal cuadrada');
```

SERIES DE FOURIER SEÑAL CUADRADA

Es una función periódica impar (antisimétrica). solo tiene armónicos impares, pero su amplitud decrece con
1/n, por lo tanto, es menos suave y con más contenido espectral.

Fórmula general:

$$x(t) = \sum_{n=1,3,5,\ldots}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right)$$

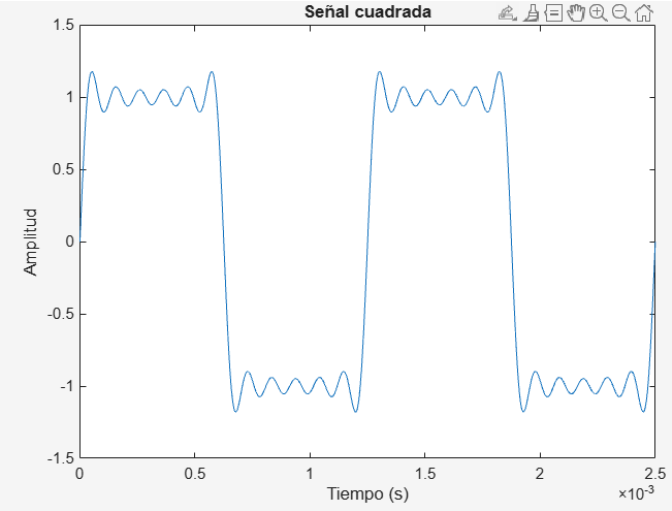


Ilustración 34 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=1.260;
A3=0.437;
A5=0.251;
A7=0.182;
A9=0.144;
A11=0.115;
v1=A1*sin(2*pi*f*t);
v2=A3*sin(2*pi*3*f*t);
v3=A5*sin(2*pi*5*f*t);
v4=A7*sin(2*pi*7*f*t);
v5=A9*sin(2*pi*9*f*t);
v6=A11*sin(2*pi*11*f*t);

VT=(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal cuadrada');
```

$$x(t) = \sum_{n=1,3,5,\ldots}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right)$$

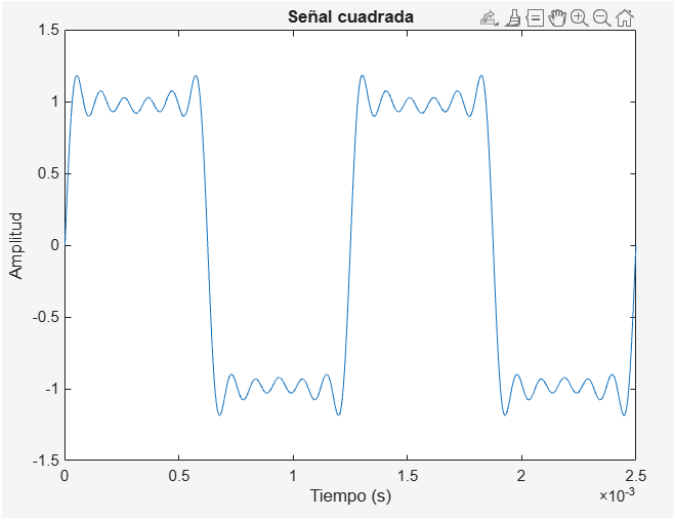


Ilustración 35 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal cuadrada en el dominio del tiempo, pero hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de generación de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran una buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOTIEMPOCUA1.csv');
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOCUA1.csv',
opts);
% Convertir las columnas correctas (4 y 5) a
vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

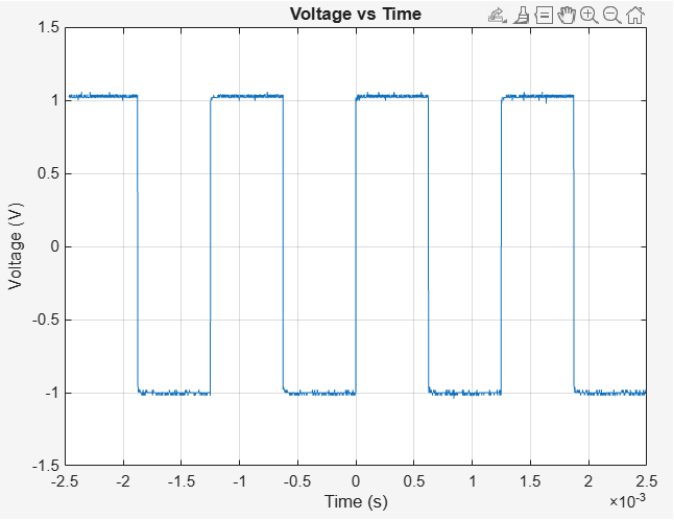


Ilustración 36 grafica generada por Matlab con los datos exportados-tiempo.

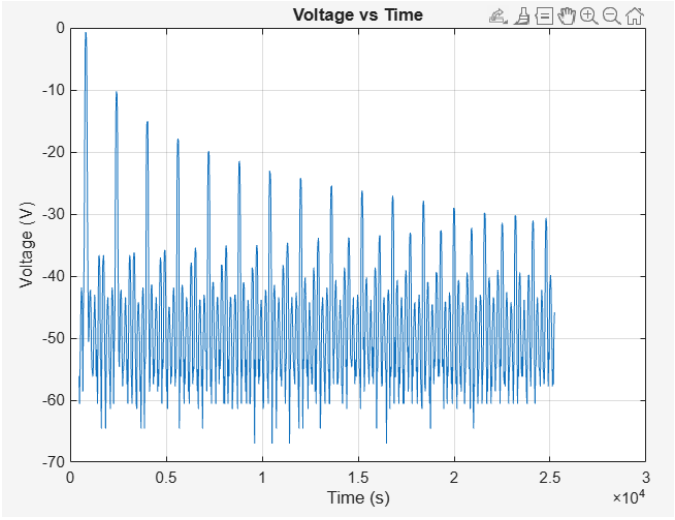


Ilustración 37 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

Es una señal cuadrada periódica claramente definida, La señal está centrada en cero, lo que indica que no hay presencia de componente DC positiva.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOFRECUENCIACUA1.csv');
opts.DataLines = [18 Inf];  % Solo desde la fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres de variables en la fila 18

% Leer los datos
datos = readtable('DOMINIOFRECUENCIACUA1.csv', opts);

% Convertir las columnas correctas (4 y 5) a vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje

% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s)
f = 1/T;                  % Frecuencia fundamental (Hz)
Fs = 1e6;                 % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T;           % Tiempo (dos periodos)
% Coeficientes de armónicos impares
A1 = 1.273;
A3 = 0.424;
A5 = 0.254;
A7 = 0.181;
A9 = 0.141;
A11 = 0.115;
% Construir señal cuadrada por suma de senos impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT = v1 + v2 + v3 + v4 + v5 + v6;
% ---------- FFT ----------
N = length(VT);             % Número de muestras
Y = fft(VT);                % Transformada rápida de Fourier
Y_mag = abs(Y)/N;           % Magnitud normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  % Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
```

```
Y_dB = 20*log10(Y_mag + eps);      % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);          % Zoom para ver
primeros armónicos
ylim([-80 5]);            % Escala típica en
dB
xticks(0:1000:15000);     % Etiquetas del eje
X cada 1000 Hz
```

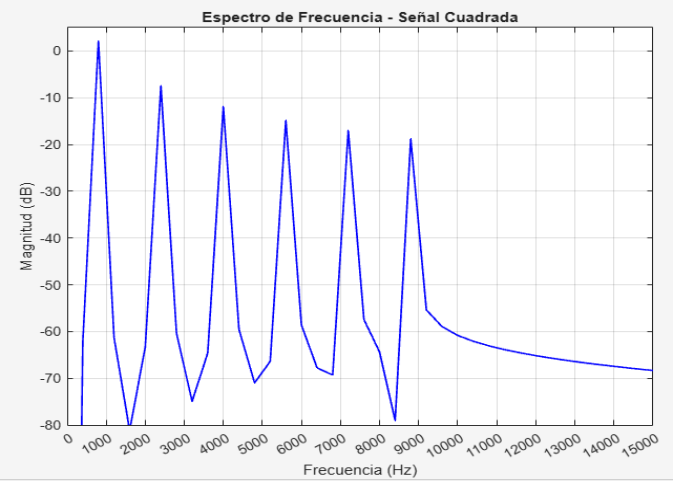


Ilustración 38 grafica generada en Matlab FFT de cuadrada1

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que ambas gráficas son coherentes entre sí ya que los gráficos representan una señal cuadrada generada como suma de armónicos impares (frecuencias: 1f, 3f, 5f, 7f, 9f, 11f), las componentes espectrales aparecen a frecuencias específicas (múltiplos impares del fundamental), lo cual es típico de una señal cuadrada. Cuando la señal fue generada con más armónicos, el espectro mostró mayor riqueza, Cuando provino de datos experimentales, aparecieron armónicos menos exactos (ruido) y el espectro fue mínimamente más disperso.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s)
f = 1/T;                  % Frecuencia
fundamental (Hz)
Fs = 1e6;                 % Frecuencia de
muestreo (Hz)
t = 0:1/Fs:2*T;           % Tiempo (dos
periodos)
% Coeficientes de armónicos impares
A1 = 1.273;
A3 = 0.424;
A5 = 0.254;
A7 = 0.181;
A9 = 0.141;
A11 = 0.115;
% Construir señal cuadrada por suma de senos
impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT = v1 + v2 + v3 + v4 + v5 + v6;
% ---------- FFT ----------
N = length(VT);               % Número de
muestras
Y = fft(VT);                  % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;             % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  %
Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);     % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
% (opcional, se puede comentar si no quieres
mostrarla)
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);
ylim([-80 5]);
xticks(0:1000:15000);
% -------- IFFT: Reconstruir señal en el
tiempo --------
```

```
v_rec = ifft(Y, 'symmetric');  %
Reconstrucción de la señal original
% -------- Gráfica de la señal reconstruida
(IFFT) --------
figure;
plot(t*1000, v_rec, 'r', 'LineWidth', 1.5);
% Tiempo en milisegundos
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

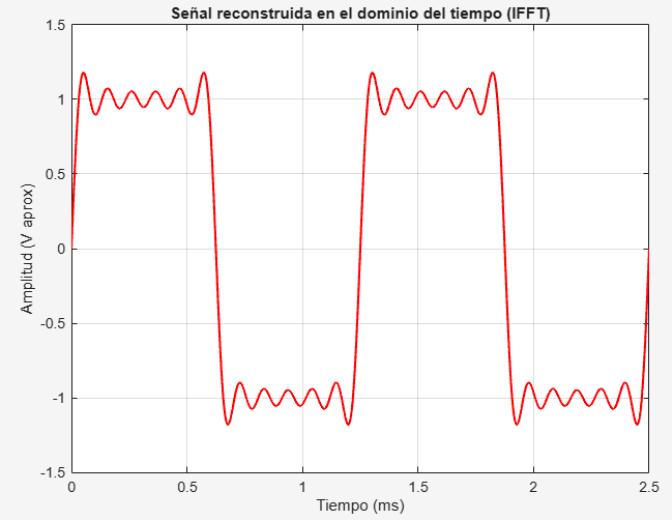


Ilustración 39 Grafica Generada Por Matlab IFFT

Visualmente, la señal es simétrica, periódica y respeta los bordes definidos de una onda cuadrada, Esta señal es una versión reconstruida desde el dominio de la frecuencia, lo que demuestra que la información de la señal se conserva tras el paso por la FFT y la IFFT.

7. señal: cuadrada 2 con DC=1.25

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz						
Vp	1						
SEÑAL	DC	ARMONICA	f (HZ)	A			TEORICO
				d.B	Vrms	VP(v) exp	4*V/n*pi
CUAD2	1,25	1	800	-0,19	0,9784	1,3836	1,27324
		3	2400	-9,79	0,3240	0,4582	0,42441
		5	4000	-14,6	0,1862	0,2633	0,25465
		7	5600	-17,4	0,1349	0,1908	0,18189
		9	7200	-19,4	0,1072	0,1515	0,14147
		11	8800	-21	0,0891	0,1260	0,11575

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:

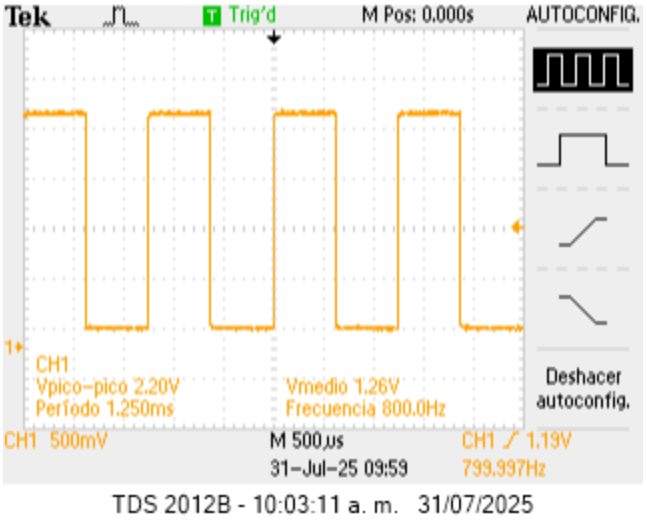


Ilustración 38 señal generada y exportada del osciloscopio - en tiempo

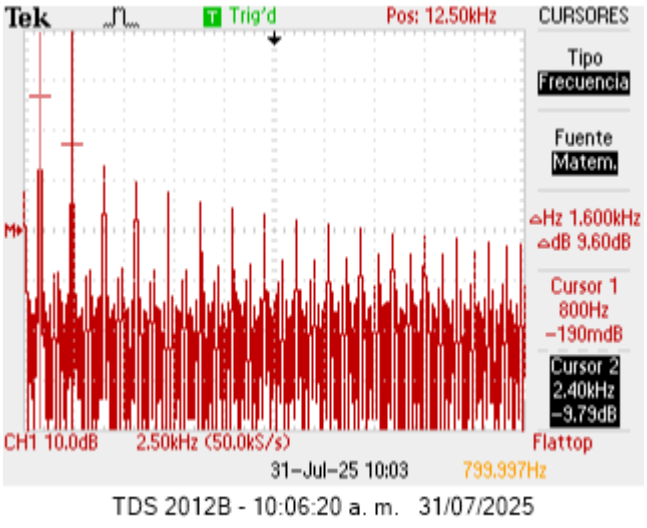


Ilustración 39 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del

tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=1.273;
A3=0.424;
A5=0.254;
A7=0.181;
A9=0.141;
A11=0.115;
v1=A1*sin(2*pi*f*t);
v2=A3*sin(2*pi*3*f*t);
v3=A5*sin(2*pi*5*f*t);
v4=A7*sin(2*pi*7*f*t);
v5=A9*sin(2*pi*9*f*t);
v6=A11*sin(2*pi*11*f*t);

VT=1.25+(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal cuadrada');
```

SERIES DE FOURIER SEÑAL CUADRADA

Es una función periódica impar (antisimétrica). solo tiene armónicos impares, pero su amplitud decrece con 1/n, por lo tanto, es menos suave y con más contenido espectral.

Fórmula general:

$$x(t) = \sum_{n=1,3,5,\ldots}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right)$$

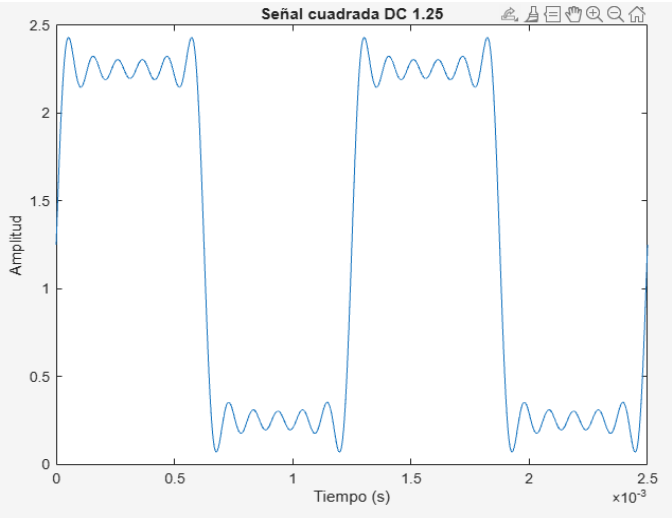


Ilustración 40 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab Cambiando los anteriores datos por los siguientes experimentales:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=1.383;
A3=0.458;
A5=0.263;
A7=0.190;
A9=0.151;
A11=0.126;
```

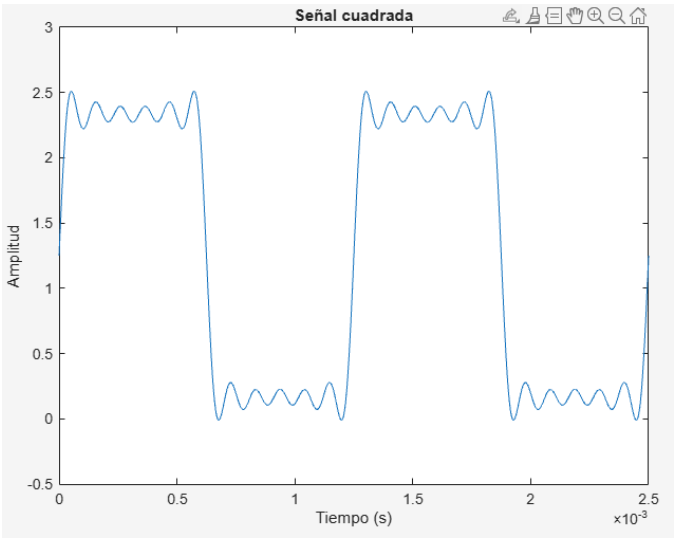


Ilustración 41 grafica generada por Matlab datos experimentales – tiempo.

Graficamos en Python con la tabla de datos en dominio del tiempo que exportamos del osciloscopio.

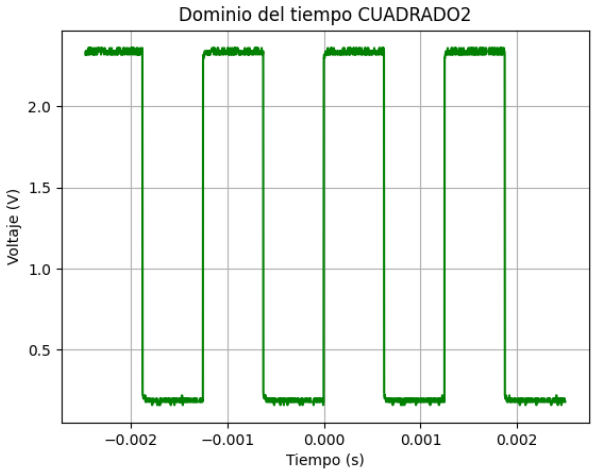


Ilustración 40 tabla Python, dominio tiempo cuadrada 2

Al observar las dos gráficas proporcionadas, ambas representan una señal cuadrada en el dominio del tiempo, pero

hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de generación de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran una buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

La señal oscila entre un mínimo cercano a 0.25 V y un máximo de 2.25 V, es decir, tiene una componente DC (desplazamiento vertical)

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
opts =
detectImportOptions('DOMINIOTIEMPOCUA2.csv');
opts.DataLines = [18 Inf];% fila 18 en elante
opts.Delimiter = ',';  % Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOCUA2.csv',
opts);
% Convertir las columnas correctas (4 y 5) a
vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time');
grid on;
```

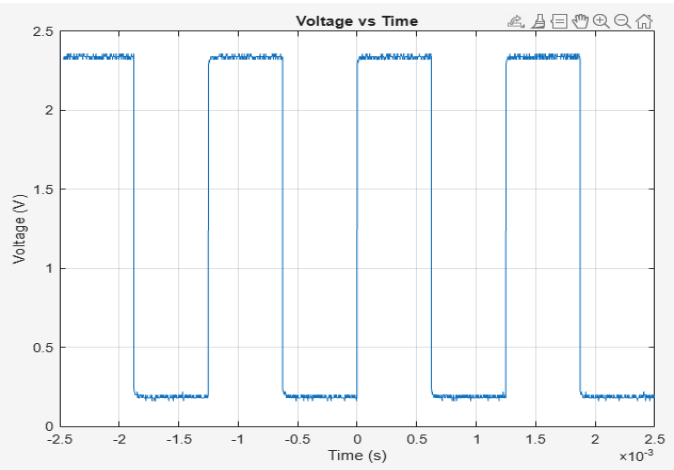


Ilustración 42 grafica generada por Matlab con los datos exportados-tiempo.

Es una señal cuadrada periódica claramente definida, La señal no está centrada en cero, lo que indica que hay presencia de componente DC positiva.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOFRECUENCIACUA2.cs
v');
opts.DataLines = [18 Inf];          % Solo
desde la fila 18 en adelante
opts.Delimiter = ',';               %
Delimitador por coma
opts.VariableNamesLine = 0;         % No hay
nombres de variables en la fila 18
% Leer los datos
datos =
readtable('DOMINIOFRECUENCIACUA2.csv', opts);
% Extraer columnas (frecuencia y magnitud en
dB)
f = datos{:,4};   % Columna 4 = Frecuencia
(Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud
(dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Triangular (FFT)');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); %
Ajuste automático del eje Y
```

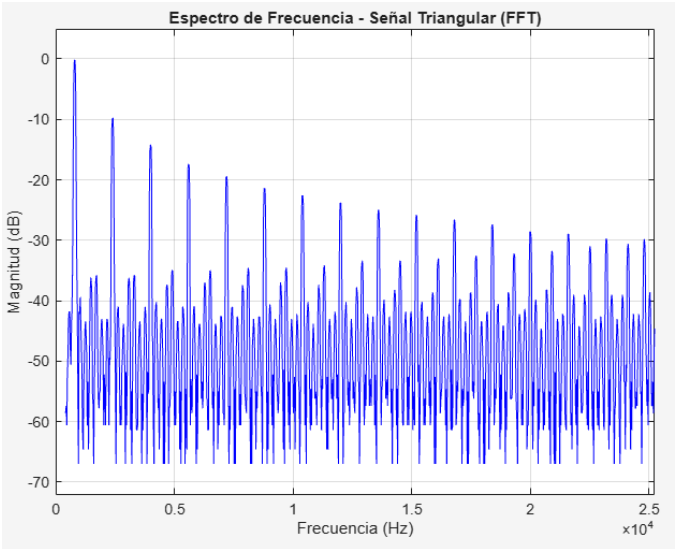


Ilustración 43 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;            % Periodo (s)
f = 1/T;                % Frecuencia
fundamental (Hz)
Fs = 1e6;               % Frecuencia de
muestreo (Hz)
t = 0:1/Fs:2*T;         % Tiempo (dos
periodos)
% Coeficientes de armónicos impares
A1 = 1.273;
A3 = 0.424;
A5 = 0.254;
A7 = 0.181;
A9 = 0.141;
A11 = 0.115;
% Construir señal cuadrada por suma de senos
impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT =1.25+( v1 + v2 + v3 + v4 + v5 + v6);
% ---------- FFT ----------
N = length(VT);             % Número de
muestras
Y = fft(VT);                % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;           % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  %
Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);     % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);          % Zoom para ver
primeros armónicos
ylim([-80 5]);            % Escala típica en
dB
xticks(0:1000:15000);     % Etiquetas del eje
X cada 1000 Hz
```

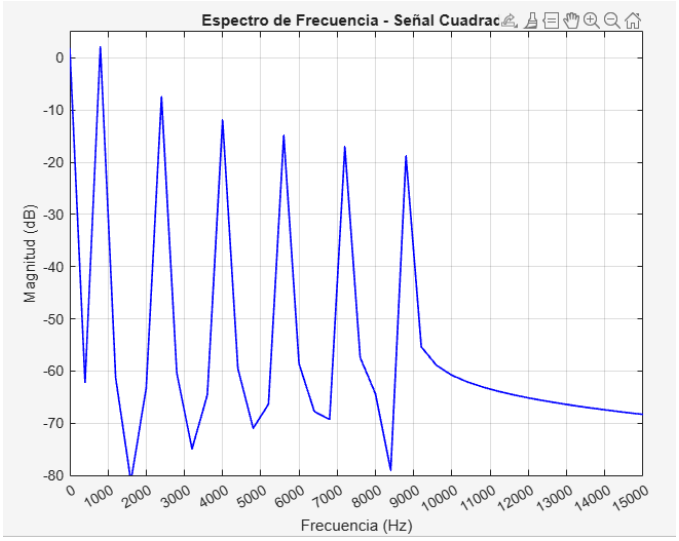


Ilustración 44 grafica generada en Matlab FFT

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que ambas gráficas son coherentes entre sí ya que los gráficos representan una señal cuadrada generada como suma de armónicos impares (frecuencias: 1f, 3f, 5f, 7f, 9f, 11f), las componentes espectrales aparecen a frecuencias específicas (múltiplos impares del fundamental), lo cual es típico de una señal cuadrada. Cuando la señal fue generada con más armónicos, el espectro mostró mayor riqueza, Cuando provino de datos experimentales, aparecieron armónicos menos exactos (ruido) y el espectro fue mínimamente más disperso.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
% Parámetros de la señal
T = 0.00125;            % Periodo (s)
f = 1/T;                % Frecuencia
fundamental (Hz)
Fs = 1e6;               % Frecuencia de
muestreo (Hz)
t = 0:1/Fs:2*T;         % Tiempo (dos
periodos)
% Coeficientes de armónicos impares
A1 = 1.273;
A3 = 0.424;
A5 = 0.254;
A7 = 0.181;
A9 = 0.141;
A11 = 0.115;
```

```
% Construir señal cuadrada por suma de senos
impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT = 1.25+(v1 + v2 + v3 + v4 + v5 + v6);
% ---------- FFT ----------
N = length(VT);              % Número de
muestras
Y = fft(VT);                 % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;            % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  %
Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);     % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
% (opcional, se puede comentar si no quieres
mostrarla)
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);
ylim([-80 5]);
xticks(0:1000:15000);
% -------- IFFT: Reconstruir señal en el
tiempo --------
v_rec = ifft(Y, 'symmetric');  %
Reconstrucción de la señal original
% -------- Gráfica de la señal reconstruida
(IFFT) --------
figure;
plot(t*1000, v_rec, 'r', 'LineWidth', 1.5);
% Tiempo en milisegundos
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

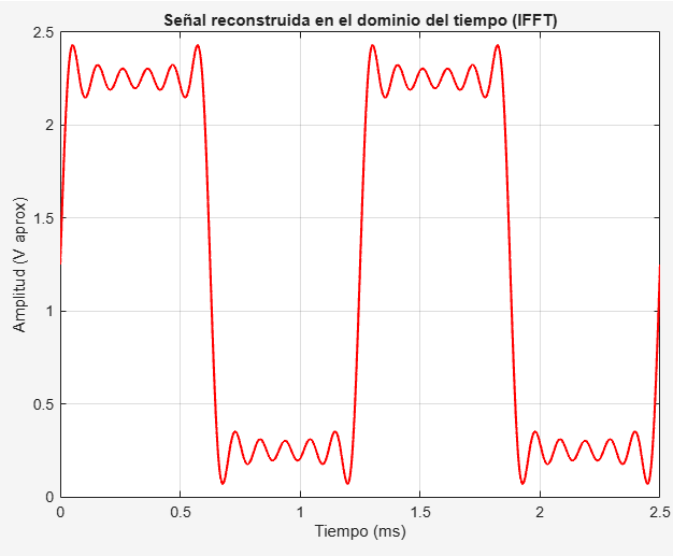


Ilustración 45 Grafica Generada Por Matlab IFFT

Es una señal cuadrada aproximada generada por una suma de senos impares (Fourier). El componente DC de 1.25 V desplaza la señal hacia arriba en el eje Y.

8. señal: PULSO 1 ciclo útil 50%

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Encontrando los siguientes datos:

f	800 Hz							
Vp	1							
DC	0							
SEÑAL	CICLO UTIL	ARMONICA	f (HZ)	A			TEORICO	
				d.B	Vrms	VP(v)	(2*A/n*π)	%ERROR
PUL1	50%	1	800	-0,19	0,9784	1,3836	1,27323954	8,6688
		3	2400	-10,2	0,3090	0,4370	0,42441318	2,9737
		5	4000	-15	0,1778	0,2515	0,25464791	1,2414
		7	5600	-17,8	0,1288	0,1822	0,18189136	0,1620
		9	7200	-19,8	0,1023	0,1447	0,14147106	2,2933
		11	8800	-21,8	0,0813	0,1150	0,05787452	98,6221

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:

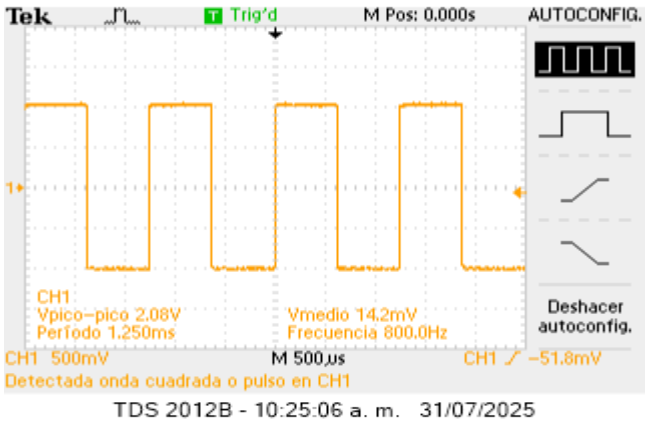


Ilustración 46 señal generada y exportada del osciloscopio - en tiempo

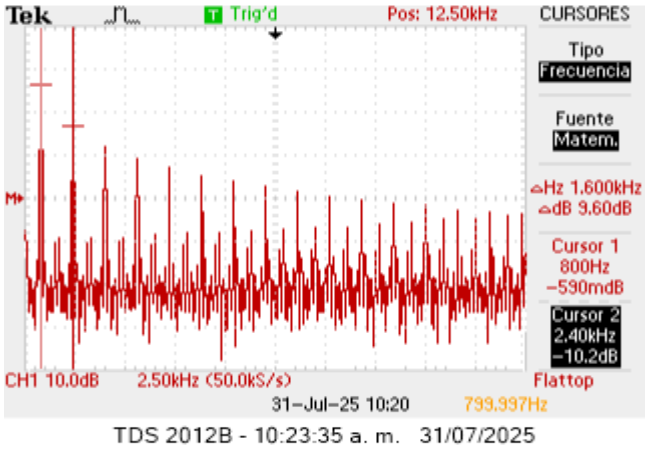


Ilustración 47 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
A1=1.273;
A3=0.424;
A5=0.254;
A7=0.181;
A9=0.141;
A11=0.115;
v1=A1*sin(2*pi*f*t);
v2=A3*sin(2*pi*3*f*t);
v3=A5*sin(2*pi*5*f*t);
v4=A7*sin(2*pi*7*f*t);
v5=A9*sin(2*pi*9*f*t);
v6=A11*sin(2*pi*11*f*t);

VT=(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 50 %');
```

$$x(t) = \sum_{n=1,3,5,\ldots}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right)$$

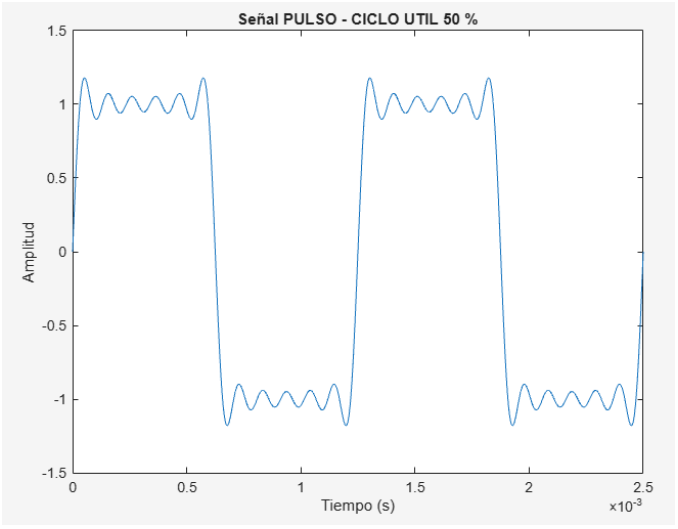


Ilustración 48 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000:2*T;
DC=0.5;
A1=1.383;
A3=0.437;
A5=0.251;
A7=0.182;
A9=0.144;
A11=0.115;
DC=0.5;
v1=A1*sin(2*pi*f*t);
v2=A3*sin(2*pi*3*f*t);
v3=A5*sin(2*pi*5*f*t);
v4=A7*sin(2*pi*7*f*t);
v5=A9*sin(2*pi*9*f*t);
v6=A11*sin(2*pi*11*f*t);

VT=(v1+v2+v3+v4+v5+v6);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 50 %');
```

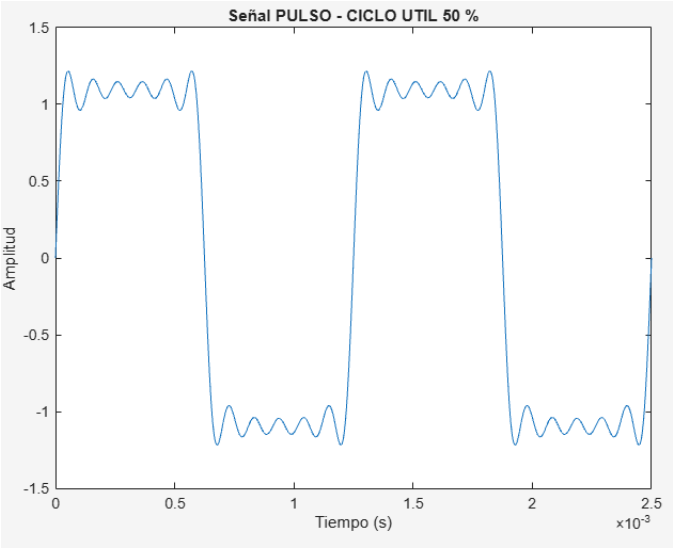


Ilustración 49 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal cuadrada en el dominio del tiempo ya que su ciclo útil es de 50%, pero hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de generación de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran una buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOTIEMPUL1.csv');
opts.DataLines = [18 Inf];  % Solo desde la fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por coma
opts.VariableNamesLine = 0; % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOPUL1.csv', opts);
% Convertir las columnas correctas (4 y 5) a vectores
t = datos{:,4};    % Columna 4 = Tiempo
v = datos{:,5};    % Columna 5 = Voltaje

% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time ciclo útil 50 %');
grid on;
```

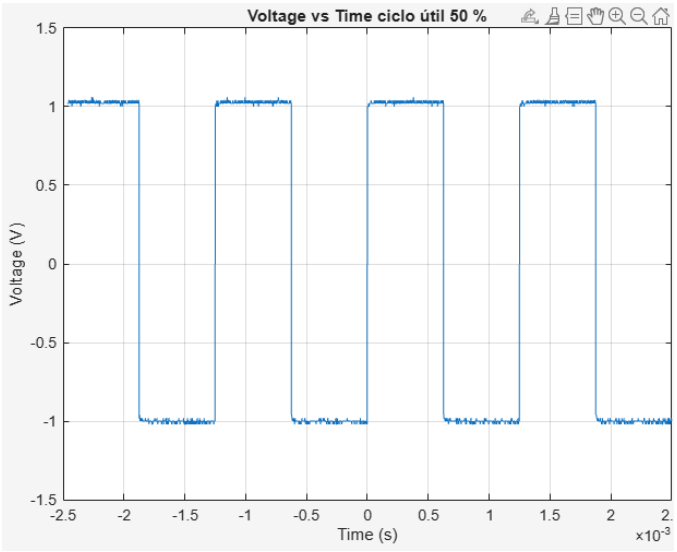


Ilustración 50 grafica generada por Matlab con los datos exportados-tiempo.

Es una señal cuadrada periódica claramente definida, La señal está centrada en cero, lo que indica que no hay presencia de componente DC positiva.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOFRECUENCIAPULSO1.csv');
opts.DataLines = [18 Inf];          % Solo desde la fila 18 en adelante
opts.Delimiter = ',';               % Delimitador por coma
opts.VariableNamesLine = 0;         % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOFRECUENCIAPULSO1.csv', opts);
% Extraer columnas (frecuencia y magnitud en dB)
f = datos{:,4};    % Columna 4 = Frecuencia (Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud (dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
```

```
title('Espectro de Frecuencia - Señal
Triangular (FFT) CICLO UTIL 50%');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); %
Ajuste automático del eje Y
```

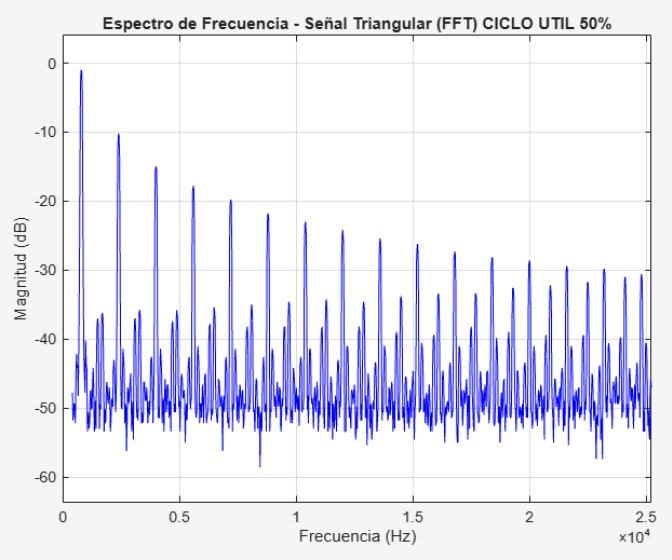


Ilustración 51 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;              % Periodo (s)
f = 1/T;                  % Frecuencia
fundamental (Hz)
Fs = 1e6;                 % Frecuencia de
muestreo (Hz)
t = 0:1/Fs:2*T;           % Tiempo (dos
periodos)
% Coeficientes de armónicos impares
A1=1.273;
A3=0.424;
A5=0.254;
A7=0.181;
A9=0.141;
A11=0.115;
% Construir señal cuadrada por suma de senos
impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT =( v1 + v2 + v3 + v4 + v5 + v6);
% ---------- FFT ----------
N = length(VT);           % Número de
muestras
Y = fft(VT);              % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;         % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  %
Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);     % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);          % Zoom para ver
primeros armónicos
ylim([-80 5]);            % Escala típica en
dB
xticks(0:1000:15000);     % Etiquetas del eje
X cada 1000 Hz
```

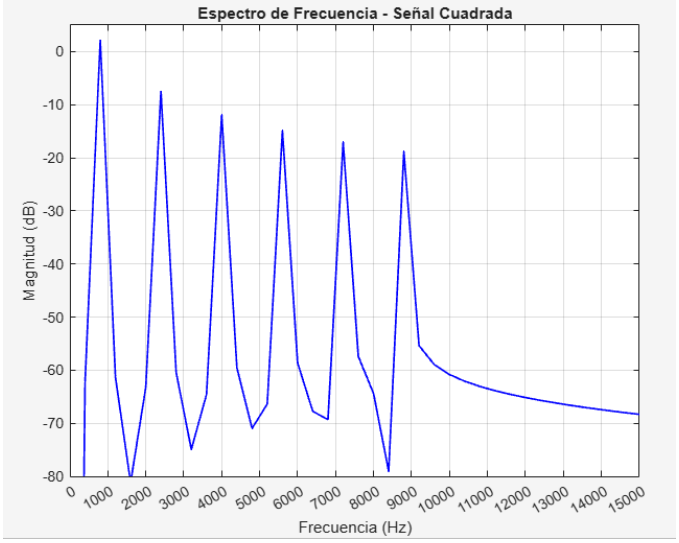


Ilustración 52 grafica generada en Matlab FFT

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que ambas gráficas son coherentes entre sí ya que los gráficos representan una señal cuadrada generada como suma de armónicos impares (frecuencias: 1f, 3f, 5f, 7f, 9f, 11f), las componentes espectrales aparecen a frecuencias específicas (múltiplos impares del fundamental), lo cual es típico de una señal cuadrada. Cuando la señal fue generada con más armónicos, el espectro mostró mayor riqueza, Cuando

provino de datos experimentales, aparecieron armónicos menos exactos (ruido) y el espectro fue mínimamente más disperso.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
% Parámetros de la señal
T = 0.00125;            % Periodo (s)
f = 1/T;                % Frecuencia
fundamental (Hz)
Fs = 1e6;               % Frecuencia de
muestreo (Hz)
t = 0:1/Fs:2*T;         % Tiempo (dos
periodos)
% Coeficientes de armónicos impares
A1=1.273;
A3=0.424;
A5=0.254;
A7=0.181;
A9=0.141;
A11=0.115;
% Construir señal cuadrada por suma de senos
impares
v1 = A1 * sin(2*pi*f*t);
v2 = A3 * sin(2*pi*3*f*t);
v3 = A5 * sin(2*pi*5*f*t);
v4 = A7 * sin(2*pi*7*f*t);
v5 = A9 * sin(2*pi*9*f*t);
v6 = A11 * sin(2*pi*11*f*t);
% Señal cuadrada aproximada
VT = (v1 + v2 + v3 + v4 + v5 + v6);
% ---------- FFT ----------
N = length(VT);             % Número de
muestras
Y = fft(VT);                % Transformada
rápida de Fourier
Y_mag = abs(Y)/N;           % Magnitud
normalizada
Y_mag(2:end-1) = 2*Y_mag(2:end-1);  %
Duplicar magnitudes (excepto DC y Nyquist)
% Convertir a decibelios
Y_dB = 20*log10(Y_mag + eps);     % eps evita
log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -------- Gráfica del espectro --------
% (opcional, se puede comentar si no quieres
mostrarla)
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Cuadrada');
grid on;
xlim([0 15000]);
ylim([-80 5]);
xticks(0:1000:15000);
% -------- IFFT: Reconstruir señal en el
tiempo --------
v_rec = ifft(Y, 'symmetric');  %
Reconstrucción de la señal original
% -------- Gráfica de la señal reconstruida
(IFFT) --------
figure;
plot(t*1000, v_rec, 'r', 'LineWidth', 1.5);
% Tiempo en milisegundos
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

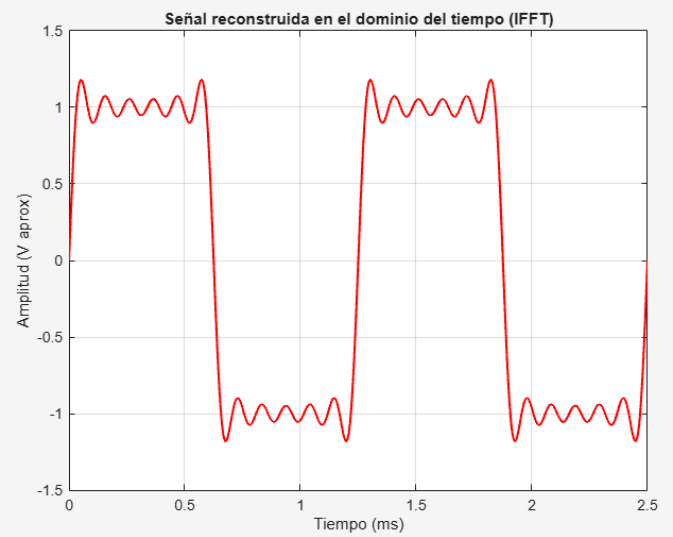


Ilustración 53 Grafica Generada Por Matlab IFFT

La señal no es perfectamente cuadrada, ya que se ha reconstruido utilizando una cantidad limitada de componentes armónicas, lo cual es típico en análisis de Fourier. La señal parece simétrica respecto al eje horizontal, lo cual indica que no contiene componentes de frecuencia par, típico de una señal cuadrada pura con ciclo útil del 50 %, esta gráfica es una representación exitosa de una señal cuadrada reconstruida a partir de sus componentes en frecuencia mediante la IFFT. Aunque no es perfecta debido al número finito de armónicos utilizados, reproduce la forma general y los niveles característicos de una onda cuadrada.

9. señal: PULSO 2 ciclo útil 20%

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.
Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Para aplicar la serie de fourier a un pulso se utiliza la formula:

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

Encontrando los siguientes datos:

f	800 Hz							
Vp	1							
DC	0							
SEÑAL	CICLO UTIL	ARMONICA	f (HZ)	A			TEORICO	%ERROR
				d.B	Vrms	VP(v)	(2*A/n*π)	
PUL2	20%	1	800	-5,39	0,5377	0,7604	0,63662	19,44
		2	1600	-7,39	0,4271	0,6040	0,31831	89,74
		3	2400	-11	0,2818	0,3986	0,21221	87,83
		4	3200	-17,8	0,1288	0,1822	0,15915	14,47
		5	3650	-30,2	0,0309	0,0437	0,12732	65,68
		6	4350	-33	0,0224	0,0317	0,10610	70,16
		7	4800	-21	0,0891	0,1260	0,09095	38,59
		8	5600	-18,2	0,1230	0,1740	0,07958	118,64
		9	6400	-19,4	0,1072	0,1515	0,07074	114,23
		10	7200	-24,6	0,0589	0,0833	0,06366	30,81
		11	8000	-36,2	0,0155	0,0219	0,05787	62,15

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:

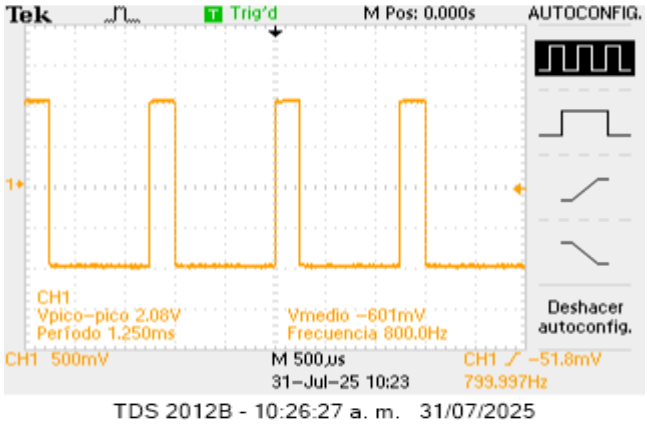


Ilustración 54 señal generada y exportada del osciloscopio - en tiempo

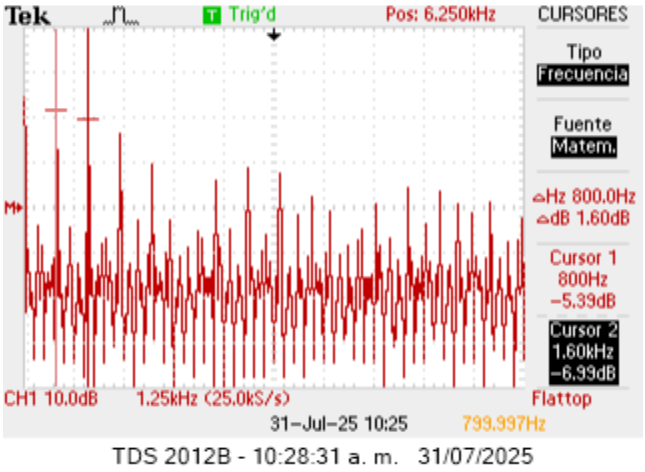


Ilustración 55 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.2;
A1=0.636;
A2=0.318;
A3=0.212;
A4=0.159;
A5=0.127;
A6=0.106;
A7=0.090;
A8=0.079;
A9=0.070;
A10=0.063;
A11=0.057;
v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 20 %');
```

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

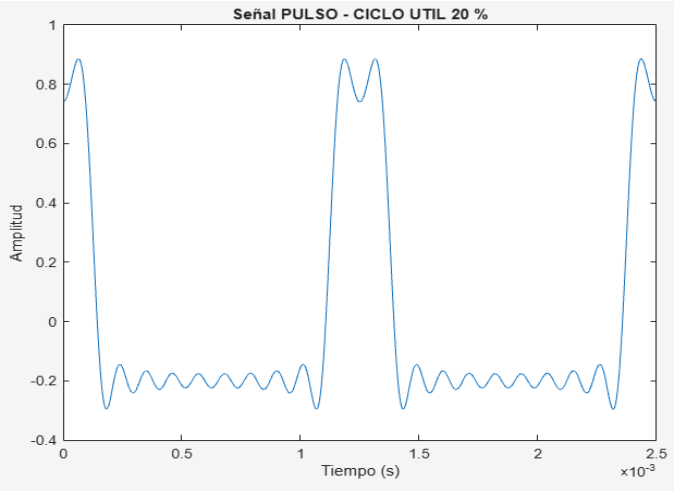


Ilustración 56 grafica generada por Matlab datos teóricos – tiempo.

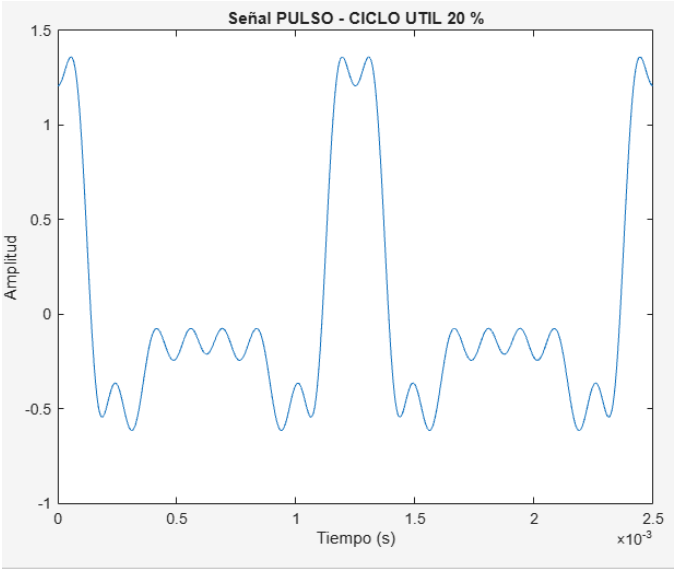


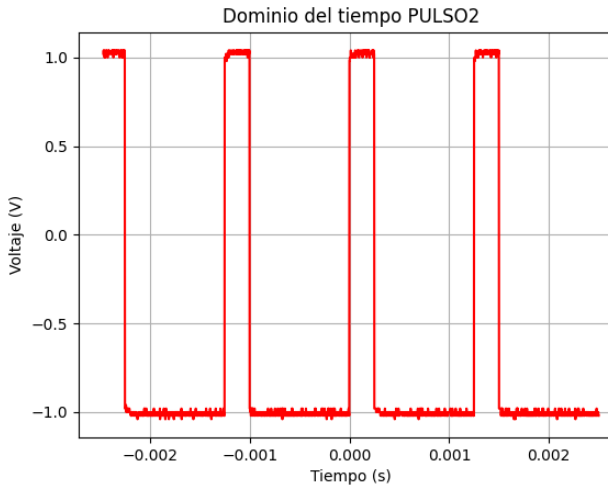
Ilustración 57 grafica generada por Matlab datos experimentales – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.2;
A1=0.760;
A2=0.604;
A3=0.398;
A4=0.182;
A5=0.043;
A6=0.031;
A7=0.126;
A8=0.174;
A9=0.015;
A10=0.08;
A11=0.022;
v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 20 %');
```

Grafica en Python:

```
import pandas as pd
import matplotlib.pyplot as plt
# Ruta al archivo
archivo = 'DOMINIOTIEMPOPULSO2.csv'
datos = pd.read_csv(archivo, skiprows=17, header=None, encoding='latin1')
# Extraer columnas de interés
tiempo = datos.iloc[:, 3]   # Columna 4 en MATLAB (índice 3)
voltaje = datos.iloc[:, 4]  # Columna 5 en MATLAB (índice 4)
# Graficar
plt.plot(tiempo, voltaje, color='r')
plt.title('Dominio del tiempo PULSO2')
plt.xlabel('Tiempo (s)')
plt.ylabel('Voltaje (V)')
plt.grid(True)
plt.show(
```



Al observar las dos gráficas proporcionadas, ambas representan una señal pulso con ciclo útil 20%, en el dominio del tiempo, pero hay algunas diferencias sutiles que pueden indicar variaciones en los parámetros de generación de la señal. Para este caso los datos de amplitud medidos son muy similares a los teóricos por lo que no se ve gran diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran una buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer archivo como tabla, empezando fila 18
opts =
detectImportOptions('DOMINIOTIEMPOPULSO2.csv'
);
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOPULSO2.csv',
opts);
% Convertir las columnas correctas (4 y 5) a
vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time ciclo útil 20 %');
grid on;
```

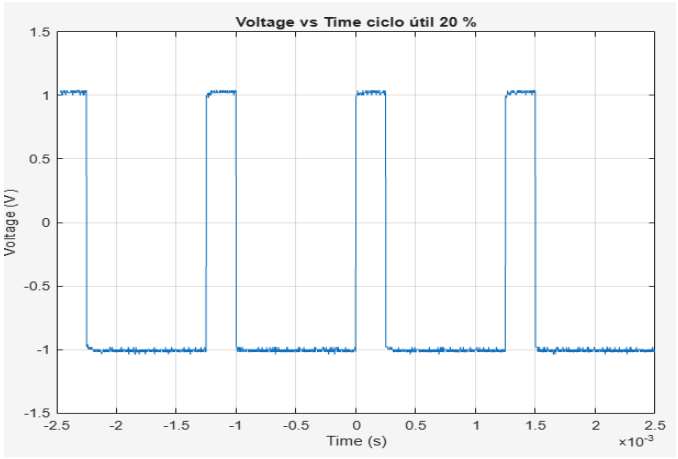


Ilustración 58 grafica generada por Matlab con los datos exportados-tiempo.

El ciclo útil del 20% significa que la señal pasa el 20% de su período en nivel alto y el resto en nivel bajo, la parte en nivel bajo (cerca de -1 V) ocupa el 80% del ciclo, permaneciendo en ese valor por mucho más tiempo que en la parte alta.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOFRECUENCIAPULSO2.
csv');
opts.DataLines = [18 Inf];          % Solo
desde la fila 18 en adelante
opts.Delimiter = ',';               %
Delimitador por coma
opts.VariableNamesLine = 0;         % No hay
nombres de variables en la fila 18
% Leer los datos
datos =
readtable('DOMINIOFRECUENCIAPULSO2.csv',
opts);
% Extraer columnas (frecuenciamagnitud en dB)
f = datos{:,4};   % Columna 4 = Frecuencia
(Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud
(dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Triangular (FFT) CICLO UTIL 50%');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); %
Ajuste automático del eje Y
```

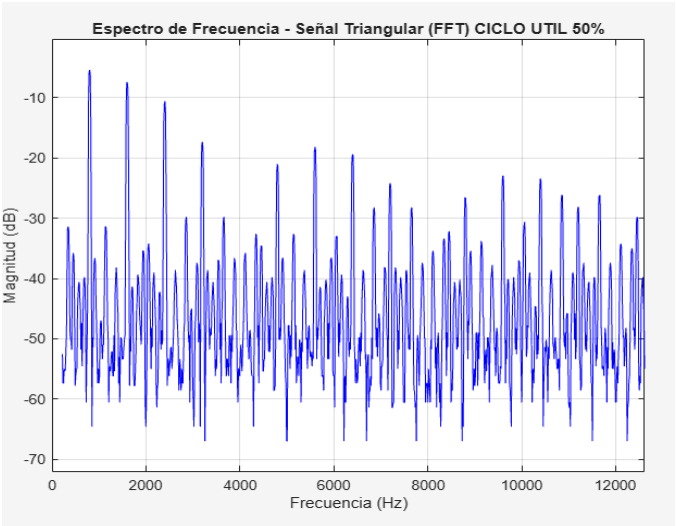


Ilustración 59 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9;      % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
D = 0.2; % Ciclo útil del 20%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
% Construcción de la señal con componentes de
coseno
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);
% Señal final compuesta
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9 + v10 + v11;
% --------------- GRÁFICA EN TIEMPO
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 20 %');
grid on;

%  FFT Y ESPECTRO ---------------------
N = length(VT);  % Número de muestras
Y = fft(VT);        % FFT de la señal
Y_mag = abs(Y) / N;   % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  %
Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps); % Convertir a
dB (evitar log(0) con eps)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);

% -- GRÁFICA EN FRECUENCIA----------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.3);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 20%');
grid on;
xlim([0 20000]); % Zoom para primeros
armónicos
ylim([-80 5]);
xticks(0:1000:20000); % Marcas cada 1 kHz
```

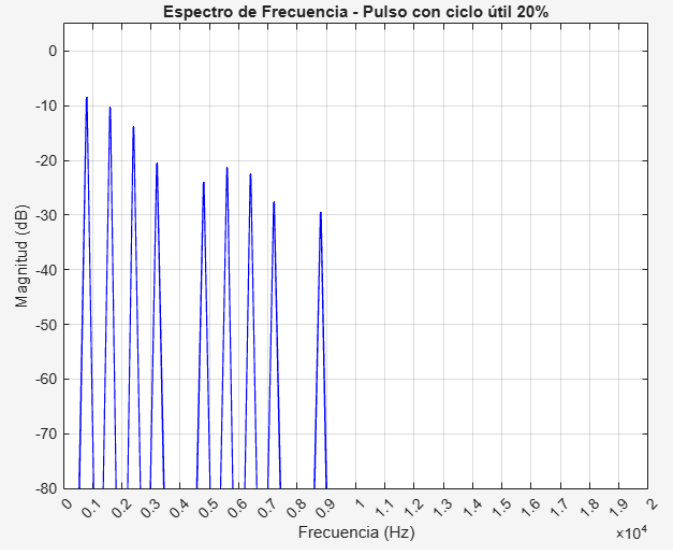


Ilustración 60 grafica generada en Matlab FFT de pulso2

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que debido a que el pulso sólo dura el 20% del período, la magnitud de los armónicos decrece rápidamente con el orden. La forma del espectro muestra picos principales en la frecuencia fundamental y armónicos, a diferencia de una onda cuadrada (que solo tiene armónicos impares en su espectro), en esta señal de pulso rectangular con ciclo útil del 20%, todos los múltiplos (pares e impares) aparecen en el espectro.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9; % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
```

```
D = 0.2;  % Ciclo útil del 20%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
% Construcción de la señal con componentes de
coseno
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);

% Señal compuesta original
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9 + v10 + v11;
% - GRÁFICA EN TIEMPO ORIGINAL -------------
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 20 %');
grid on;

% ------ FFT Y ESPECTRO -------------
N = length(VT);   % Número de muestras
Y = fft(VT);% Transformada rápida de Fourier
Y_mag = abs(Y) / N; % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  %
Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps) % Evita log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);

% -GRÁFICA EN FRECUENCIA -------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.3);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 20%');
grid on;
xlim([0 20000]);
ylim([-80 5]);
xticks(0:1000:20000);

% - IFFT Y SEÑAL RECONSTRUIDA ------------
vt_rec = ifft(Y, 'symmetric');    %
Reconstrucción en tiempo (real)

% --- GRÁFICA SEÑAL RECONSTRUIDA -----------
figure;
plot(t*1000, vt_rec, 'r');   % t en
milisegundos para mejor visualización
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

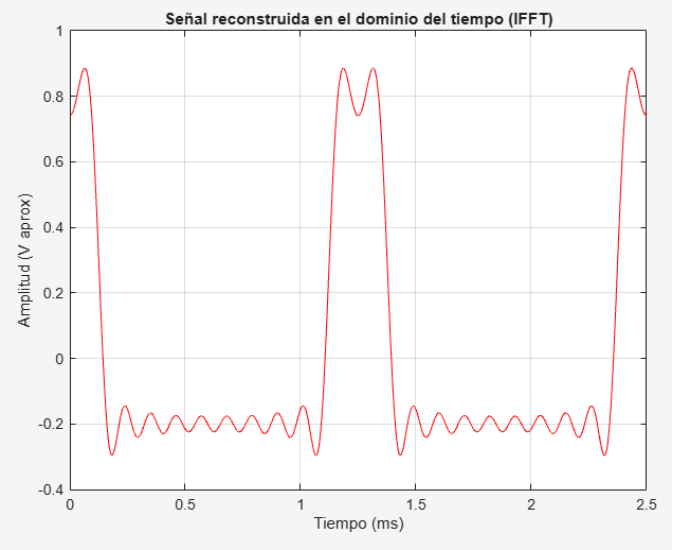


Ilustración 61 Grafica Generada Por Matlab IFFT

La gráfica muestra una onda no sinusoidal, típica de una señal con componentes armónicas múltiples, resultado de la expansión en serie de Fourier para un pulso rectangular con ciclo útil distinto del 50%, el Ciclo útil del 20% significa que el pulse activo está solo en el 20% del ciclo completo, por lo que la forma general es similar a un pulso estrecho

10. señal: PULSO 3 ciclo útil 30%

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Para aplicar la serie de fourier a un pulso se utiliza la formula:

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

Encontrando los siguientes datos:

f	800 Hz							
Vp	1							
DC	0							
SEÑAL	CICLO UTIL	ARMONICA	f (HZ)	A			TEORICO	
				d.B	Vrms	VP(v)	(2*A/n*π)	%ERROR
PUL3	30%	1	800	-4,59	0,5895	0,8337	0,6366	30,96
		2	1600	-7,39	0,4271	0,6040	0,3183	89,74
		3	2400	-20,6	0,0933	0,1320	0,2122	37,80
		4	3200	-17,4	0,1349	0,1908	0,1592	19,87
		5	4000	-15	0,1778	0,2515	0,1273	97,52
		6	4800	-21	0,0891	0,1260	0,1061	18,79
		7	5600	-28,2	0,0389	0,0550	0,0909	39,50
		8	6400	-19,4	0,1072	0,1515	0,0796	90,43
		9	7200	-21,8	0,0813	0,1150	0,0707	62,51
		10	8800	-23,8	0,0646	0,0913	0,0637	43,43
		11	9600	-23	0,0708	0,1001	0,0579	72,99

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:

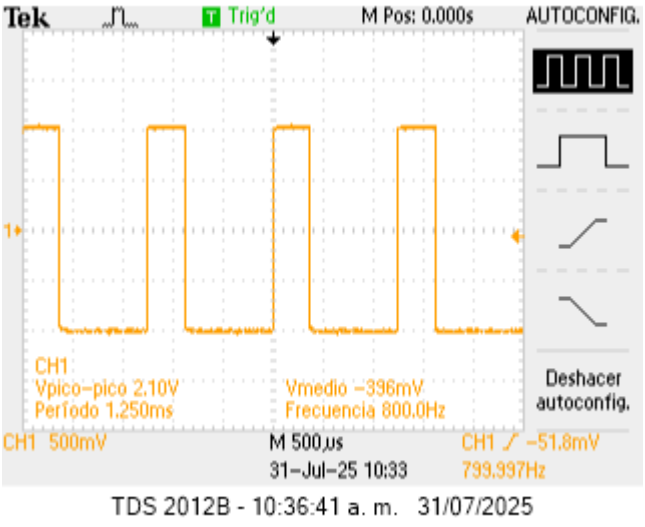


Ilustración 62 señal generada y exportada del osciloscopio - en tiempo

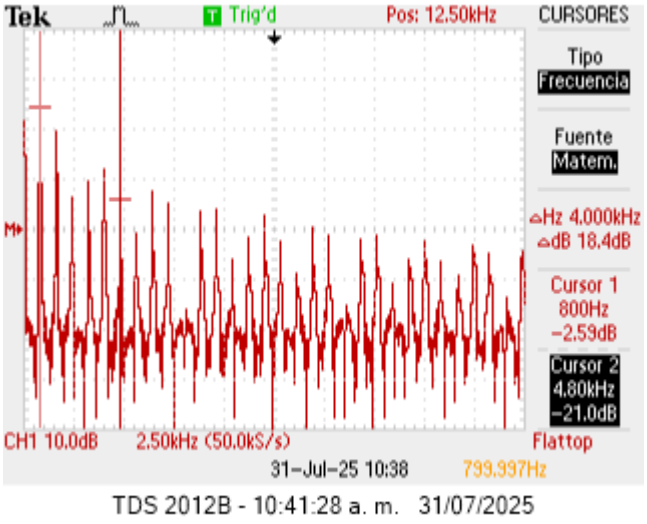


Ilustración 63 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.3;
A1=0.636;
A2=0.318;
A3=0.212;
A4=0.159;
A5=0.127;
A6=0.106;
A7=0.090;
A8=0.079;
A9=0.070;
A10=0.063;
A11=0.057;
v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 30 %');
```

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

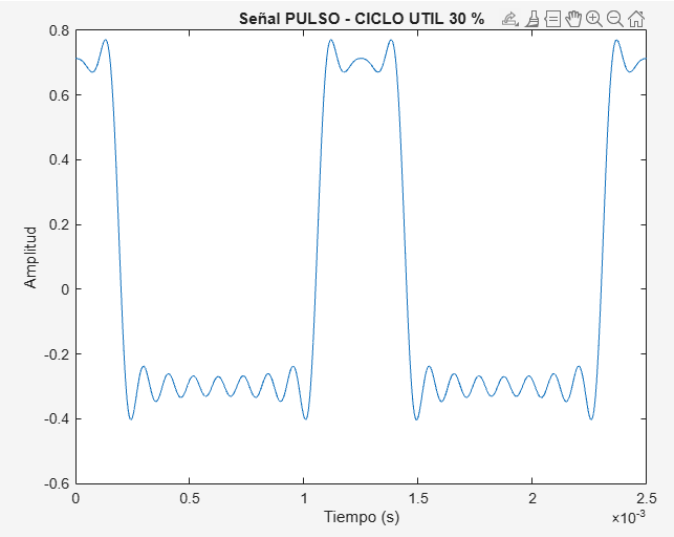


Ilustración 64 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.2;
A1=0.760;
A2=0.604;
A3=0.398;
A4=0.182;
A5=0.043;
A6=0.031;
A7=0.126;
A8=0.174;
A9=0.015;
A10=0.08;
A11=0.022;
v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 20 %');
```

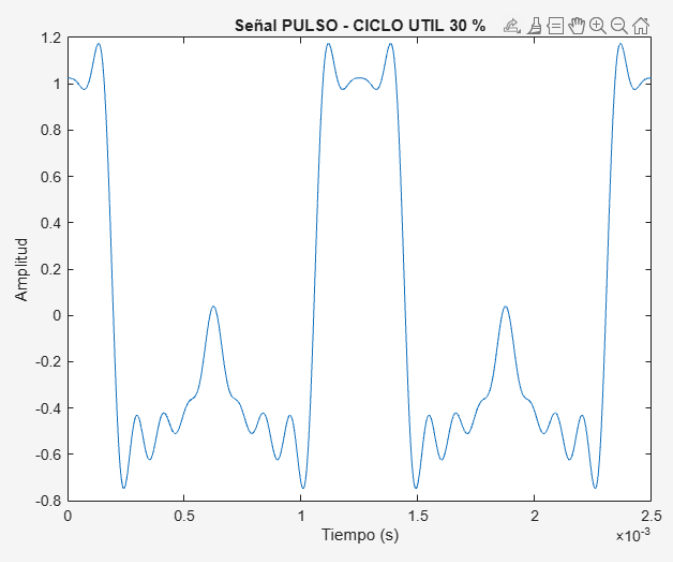


Ilustración 65 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal pulso con ciclo útil 30%, en el dominio del tiempo, pero hay algunas diferencias que pueden indicar variaciones en los parámetros de medición de la señal, podría atribuirse a error humano que se haya tomado la medición experimental de armónicas que no corresponden, los datos de amplitud medidos no son muy similares a los teóricos por lo que se ve alguna diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran sin embargo buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer archivo como tabla, empezando fila 18
opts =
detectImportOptions('DOMINIOTIEMPOPULSO3.csv'
);
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0; % No hay nombres
de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOTIEMPOPULSO3.csv',
opts);
```

```
% Convertir las columnas correctas (4 y 5) a
vectores
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time ciclo útil 30 %');
grid on;
```

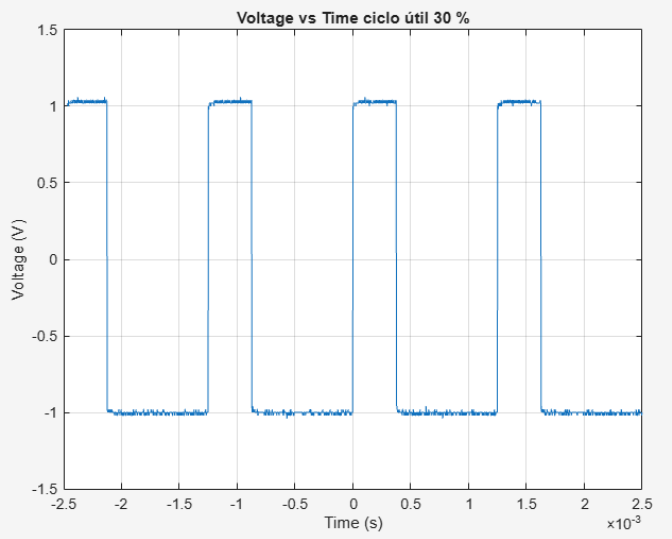


Ilustración 66 grafica generada por Matlab con los datos exportados-tiempo.

El ciclo útil del 30% significa que la señal pasa el 30% de su período en nivel alto y el resto en nivel bajo, la parte en nivel bajo (cerca de -1 V) ocupa el 70% del ciclo, permaneciendo en ese valor por mucho más tiempo que en la parte alta.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde
la fila 18
opts =
detectImportOptions('DOMINIOFRECUENCIAPULSO3.
csv');
opts.DataLines = [18 Inf];          % Solo
desde la fila 18 en adelante
opts.Delimiter = ',';               %
Delimitador por coma
opts.VariableNamesLine = 0;         % No hay
nombres de variables en la fila 18
% Leer los datos
datos =
readtable('DOMINIOFRECUENCIAPULSO3.csv',
opts);
% Extraer columnas (frecuenciamagnitud en dB)
f = datos{:,4};   % Columna 4 = Frecuencia
(Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud
(dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal
Triangular (FFT) CICLO UTIL 30%');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); %
Ajuste automático del eje Y
```

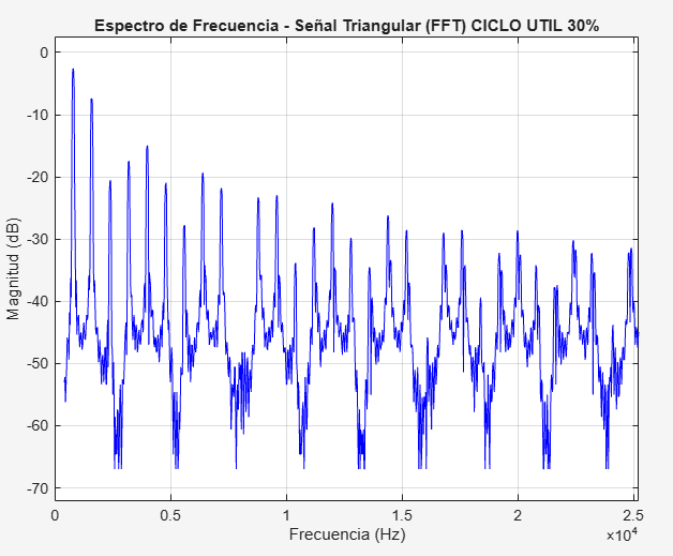


Ilustración 68 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9;      % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
D = 0.3; % Ciclo útil del 30%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
% Construcción de la señal con componentes de
coseno
```

```
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);
% Señal final compuesta
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9 + v10 + v11;
% ------- GRÁFICA EN TIEMPO
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 30 %');
grid on;
%  FFT Y ESPECTRO ------------
N = length(VT);  % Número de muestras
Y = fft(VT);        % FFT de la señal
Y_mag = abs(Y) / N;   % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  %
Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps); % Convertir a
dB (evitar log(0) con eps)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -- GRÁFICA EN FRECUENCIA-------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.3);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 30%');
grid on;
xlim([0 20000]); % Zoom para primeros
armónicos
ylim([-80 5]);
xticks(0:1000:20000); % Marcas cada 1 kHz
```

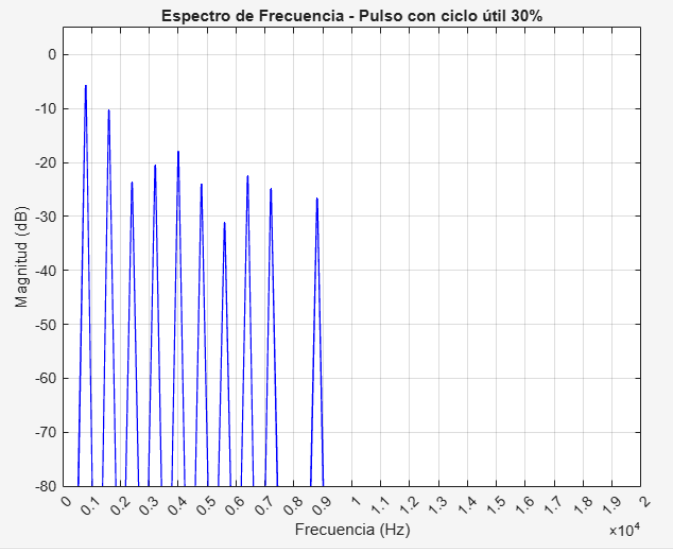


Ilustración 69 grafica generada en Matlab FFT de pulso3

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que debido a que el pulso sólo dura el 30% del período en alto, la magnitud de los armónicos decrece rápidamente con el orden. La forma del espectro muestra picos principales en la frecuencia fundamental y armónicos, a diferencia de una onda cuadrada (que solo tiene armónicos impares en su espectro), en esta señal de pulso rectangular con ciclo útil del 30%, todos los múltiplos (pares e impares) aparecen en el espectro.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9; % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
D = 0.3;  % Ciclo útil del 30%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
```

```
% Construcción de la señal con componentes de
coseno
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);
% Señal compuesta original
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9 + v10 + v11;
% - GRÁFICA EN TIEMPO ORIGINAL -------------
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 30 %');
grid on;
% ------ FFT Y ESPECTRO -------------
N = length(VT);   % Número de muestras
Y = fft(VT);% Transformada rápida de Fourier
Y_mag = abs(Y) / N; % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  %
Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps); % Evita log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -GRÁFICA EN FRECUENCIA -------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.3);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 30%');
grid on;
xlim([0 20000]);
ylim([-80 5]);
xticks(0:1000:20000);
% - IFFT Y SEÑAL RECONSTRUIDA ------------
vt_rec = ifft(Y, 'symmetric');    %
Reconstrucción en tiempo (real)
% --- GRÁFICA SEÑAL RECONSTRUIDA -----------
figure;
plot(t*1000, vt_rec, 'r');   % t en
milisegundos para mejor visualización
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');

title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

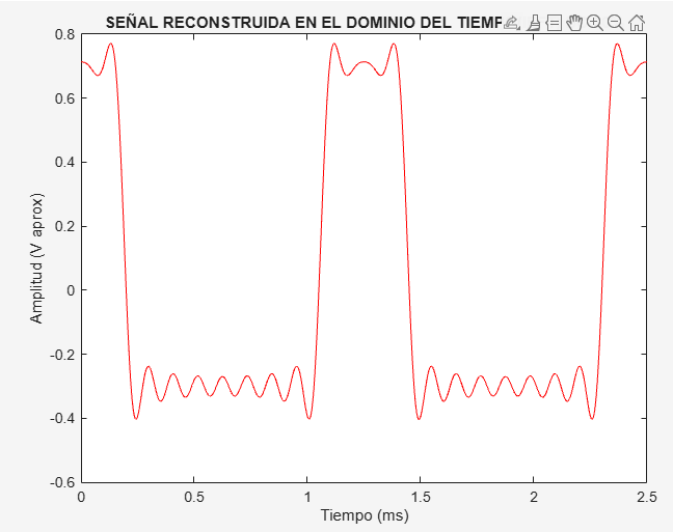


Ilustración 70 Grafica Generada Por Matlab IFFT

La gráfica muestra una típica de una señal con componentes armónicas múltiples, resultado de la expansión en serie de Fourier para un pulso rectangular con ciclo útil distinto del 50%, el Ciclo útil del 30% significa que el pulse activo está solo en el 30% del ciclo completo, por lo que la forma general es similar a un pulso estrecho

11. señal: PULSO 4 ciclo útil 80%

Se procedió a tomar los valores experimentales haciendo las mediciones en el osciloscopio.

Los valores mostrados por el osciloscopio están en decibles por lo que aplicamos la siguiente fórmula para pasar a voltaje Vrms.

$$V_{\text{en volts}} = V_{\text{referencia}} \times 10^{\frac{\text{dB}}{20}}$$

posteriormente multiplicado por raíz de dos nos da el voltaje pico.

$$V_{pico} = V_{RMS} \times \sqrt{2}$$

Para aplicar la serie de fourier a un pulso se utiliza la formula:

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

Encontrando los siguientes datos:

f	800 Hz						
Vp	1						
DC	0						
SEÑAL	CICLO UTIL	ARMONICA	f (HZ)	A			TEORICO
				d.B	Vrms	VP(v)	(2*A/n*π)
P4	80%	1	800	-6,49	0,4737	0,6699	0,6366
		2	1600	-12,8	0,2291	0,3240	0,3183
		3	2400	-15,6	0,1660	0,2347	0,2122
		4	3200	-18,8	0,1148	0,1624	0,1592
		5	3650	-21,2	0,0871	0,1232	0,1273
		6	4350	-23	0,0708	0,1001	0,1061
		7	4800	-24	0,0631	0,0892	0,0909
		8	5600	-25	0,0562	0,0795	0,0796
		9	6400	-26	0,0501	0,0709	0,0707
		10	7200	-28	0,0398	0,0563	0,0637
		11	8000	-33,2	0,0219	0,0309	0,0579

Imágenes de señal generada y exportada del osciloscopio, tanto en tiempo como en frecuencia:

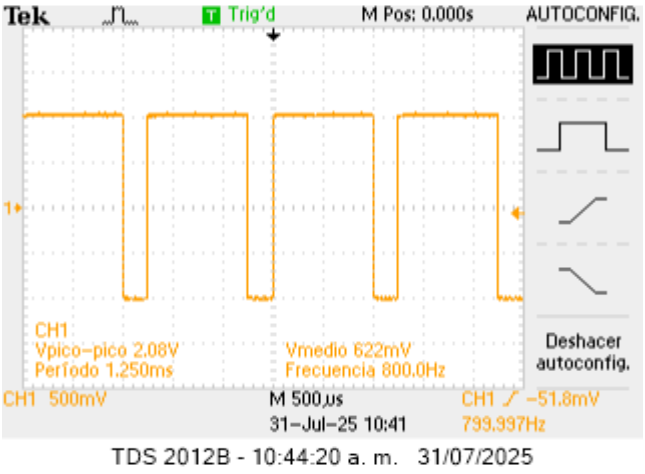


Ilustración 71 señal generada y exportada del osciloscopio - en tiempo

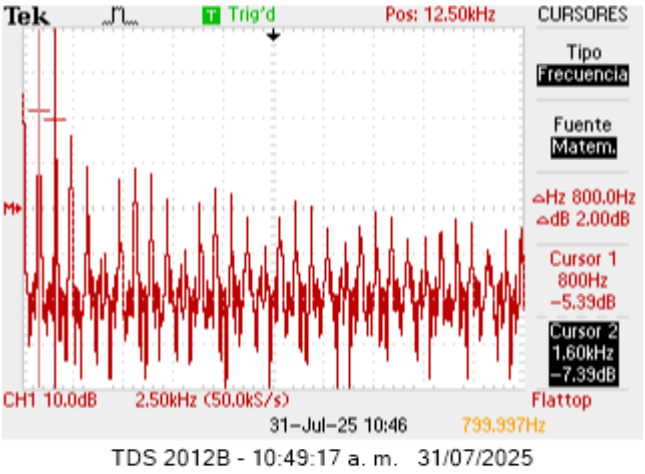


Ilustración 72 señal generada y exportada del osciloscopio - en espectro de frecuencia.

De acuerdo a estos datos teóricos calculados en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.8;
A1=0.636;
A2=0.318;
A3=0.212;
A4=0.159;
A5=0.127;
A6=0.106;
A7=0.090;
A8=0.079;
A9=0.070;
A10=0.063;
A11=0.057;
v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);
VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 80 %');
```

$$v(t) = \sum_{n=1}^{\infty} \frac{2A}{n\pi} \cdot \sin(n\pi D) \cdot \cos(2\pi n f t)$$

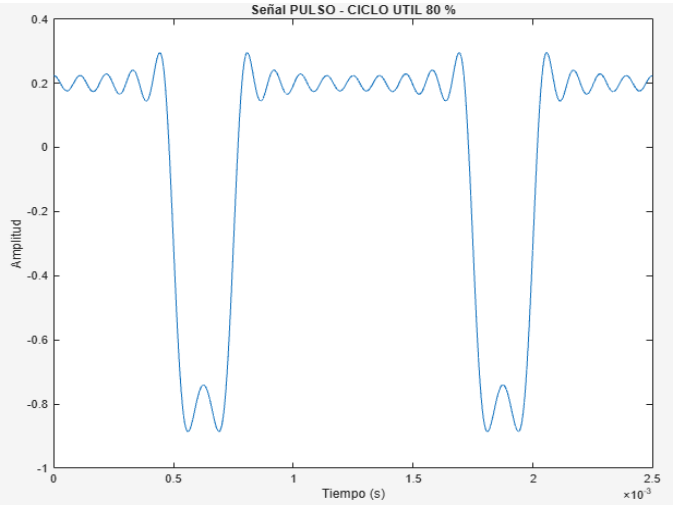


Ilustración 73 grafica generada por Matlab datos teóricos – tiempo.

De acuerdo a los datos experimentales tomados en el laboratorio en el dominio del tiempo graficamos en Matlab con el siguiente código:

```
T=0.00125;
f=1/T;
t=0:1/1000000000:2*T;
D=0.8;
A1=0.669;
A2=0.324;
A3=0.234;
A4=0.162;
A5=0.123;
A6=0.100;
A7=0.089;
A8=0.079;
A9=0.070;
A10=0.056;
A11=0.050;

v1=A1*sin(1*pi*D)*cos(2*pi*1*f*t);
v2=A2*sin(2*pi*D)*cos(2*pi*2*f*t);
v3=A3*sin(3*pi*D)*cos(2*pi*3*f*t);
v4=A4*sin(4*pi*D)*cos(2*pi*4*f*t);
v5=A5*sin(5*pi*D)*cos(2*pi*5*f*t);
v6=A6*sin(6*pi*D)*cos(2*pi*6*f*t);
v7=A7*sin(7*pi*D)*cos(2*pi*7*f*t);
v8=A8*sin(8*pi*D)*cos(2*pi*8*f*t);
v9=A9*sin(9*pi*D)*cos(2*pi*9*f*t);
v10=A10*sin(10*pi*D)*cos(2*pi*10*f*t);
v11=A11*sin(11*pi*D)*cos(2*pi*11*f*t);

VT=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11);
plot(t,VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO UTIL 80 %');
```

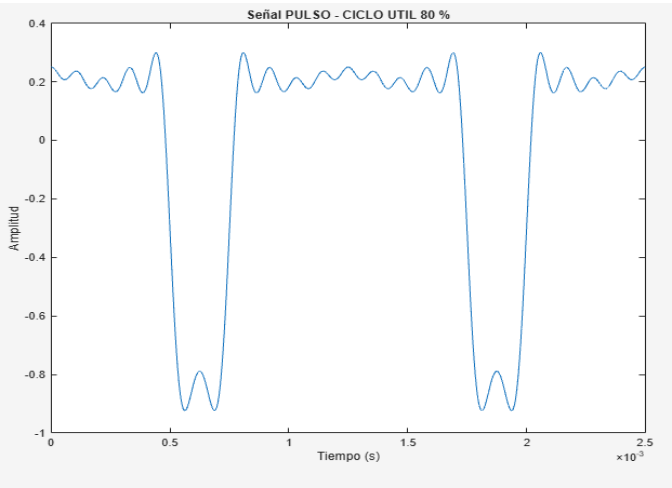


Ilustración 74 grafica generada por Matlab datos experimentales – tiempo.

Al observar las dos gráficas proporcionadas, ambas representan una señal pulso con ciclo útil 80%, en el dominio del tiempo, pero hay algunas diferencias que pueden indicar variaciones en los parámetros de medición de la señal, podría atribuirse a error humano que se haya tomado la medición experimental de armónicas que no corresponden, los datos de amplitud medidos no son muy similares a los teóricos por lo que se ve alguna diferencia (ver tabla de recolección de tatos), Ambas gráficas muestran sin embargo buena coincidencia, lo que valida el comportamiento práctico de la señal cuadrada generada.

Ahora procedemos a graficar en Matlab los datos exportados en formato CSV para el dominio del tiempo, como todos estos archivos contienen en sus primeras 18 líneas mega datos, que no hacen parte de los datos a graficar en cuanto a tiempo voltaje, el código se encarga de excluir estas líneas para poder generar la gráfica:

```
% Leer archivo como tabla, empezando fila 18
opts =
detectImportOptions('DOMINIOTIEMPOPULSO4.csv'
);
opts.DataLines = [18 Inf];  % Solo desde la
fila 18 en adelante
opts.Delimiter = ',';       % Delimitador por
coma
opts.VariableNamesLine = 0; % No hay nombres
datos = readtable('DOMINIOTIEMPOPULSO4.csv',
opts);
t = datos{:,4};   % Columna 4 = Tiempo
v = datos{:,5};   % Columna 5 = Voltaje
% Graficar
plot(t, v);
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Voltage vs Time ciclo útil 80 %');
grid on;
```

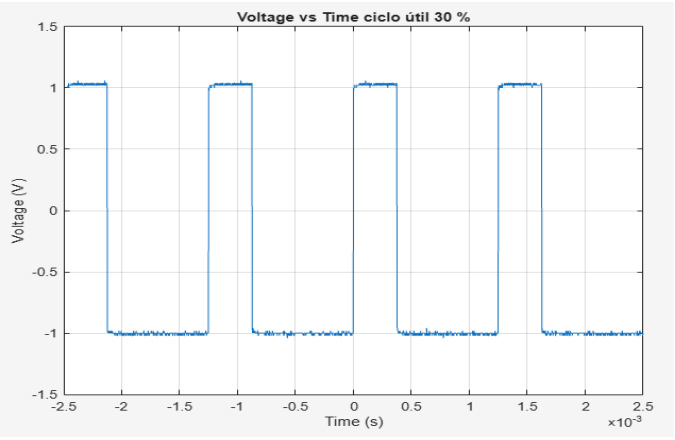


Ilustración 75 grafica generada por Matlab con los datos exportados-tiempo.

El ciclo útil del 30% significa que la señal pasa el 30% de su período en nivel alto y el resto en nivel bajo, la parte en nivel bajo (cerca de -1 V) ocupa el 70% del ciclo, permaneciendo en ese valor por mucho más tiempo que en la parte alta.

De acuerdo a los datos exportados en el dominio de la frecuencia graficamos en Matlab con el siguiente código:

```
% Leer el archivo como tabla, empezando desde la fila 18
opts = detectImportOptions('DOMINIOFRECUENCIAPULSO4.csv');
opts.DataLines = [18 Inf];          % Solo desde la fila 18 en adelante
opts.Delimiter = ',';               % Delimitador por coma
opts.VariableNamesLine = 0;         % No hay nombres de variables en la fila 18
% Leer los datos
datos = readtable('DOMINIOFRECUENCIAPULSO4.csv', opts);
% Extraer columnas (frecuenciamagnitud en dB)
f = datos{:,4};   % Columna 4 = Frecuencia (Hz)
mag_dB = datos{:,5}; % Columna 5 = Magnitud (dB)
% Graficar
plot(f, mag_dB, 'b');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Señal Triangular (FFT) CICLO UTIL 40%');
grid on;
xlim([0 max(f)]);
ylim([min(mag_dB)-5, max(mag_dB)+5]); % Ajuste automático del eje Y
```

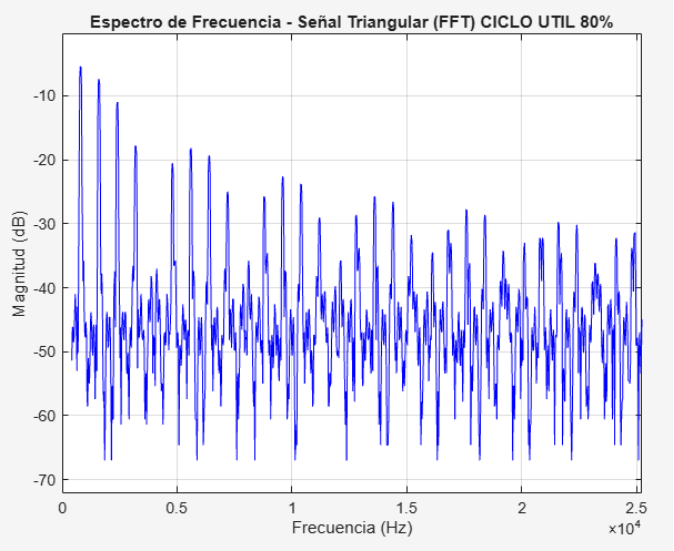


Ilustración 76 grafica generada por Matlab (dB-Hz) con los datos exportados-frecuencia.

A Continuación, se genera un código que permite obtener a partir de los datos tomados en dominio de tiempo el espectro en frecuencia de las señales mediante el cálculo de la FFT.

```
% Parámetros de la señal
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9;      % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
D = 0.8; % Ciclo útil del 30%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
% Construcción de la señal con componentes de coseno
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);
% Señal final compuesta
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9 + v10 + v11;
% ------- GRÁFICA EN TIEMPO
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 80 %');
grid on;
%  FFT Y ESPECTRO ------------
N = length(VT);  % Número de muestras
Y = fft(VT);        % FFT de la señal
Y_mag = abs(Y) / N;   % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  % Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps); % Convertir a dB (evitar log(0) con eps)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -- GRÁFICA EN FRECUENCIA------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b', 'LineWidth', 1.3);
```

```
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 80%');
grid on;
xlim([0 20000]); % Zoom para primeros
armónicos
ylim([-80 5]);
xticks(0:1000:20000); % Marcas cada 1 kHz
```

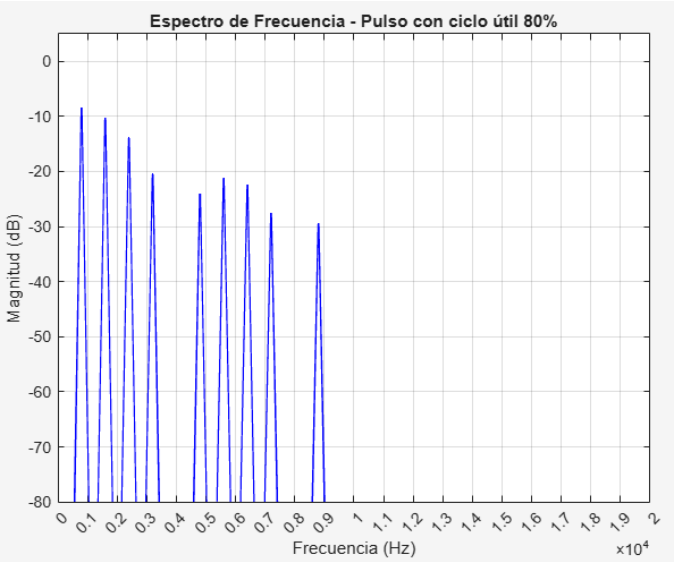


Ilustración 77 grafica generada en Matlab FFT de pulso 4

Haciendo el análisis entre la gráfica generada del espectro en frecuencia de los datos exportados y la gráfica generada aplicando la FFT a los datos tomados en dominio de tiempo, encontramos que el pulso dura el 80% del período en alto, la magnitud de los armónicos decrece con el orden. La forma del espectro muestra picos principales en la frecuencia fundamental y armónicos, a diferencia de una onda cuadrada (que solo tiene armónicos impares en su espectro), en esta señal de pulso rectangular con ciclo útil del 80%, todos los múltiplos (pares e impares) aparecen en el espectro.

Teniendo en cuenta que La FFT es una transformación compleja, necesitamos magnitud y fase para reconstruir completamente la señal, como el .csv solo contiene la magnitud (en dB), no la parte imaginaria ni la fase, Por eso, no es posible graficar $y(t)$ directamente desde esa tabla csv.

Con el siguiente código aplicamos la IFFT para reconstruirla desde los datos de la FFT antes hecha.

```
T = 0.00125;   % Periodo de la señal (s)
f = 1/T;  % Frecuencia fundamental (Hz)
Fs = 1e9; % Frecuencia de muestreo (Hz)
t = 0:1/Fs:2*T; % Tiempo (dos periodos)
D = 0.8;  % Ciclo útil del 80%
% Armónicos calculados para el pulso
A1 = 0.636;
A2 = 0.318;
A3 = 0.212;
A4 = 0.159;
A5 = 0.127;
A6 = 0.106;
A7 = 0.090;
A8 = 0.079;
A9 = 0.070;
A10 = 0.063;
A11 = 0.057;
% Construcción de la señal con componentes de
coseno
v1 = A1 * sin(1*pi*D) * cos(2*pi*1*f*t);
v2 = A2 * sin(2*pi*D) * cos(2*pi*2*f*t);
v3 = A3 * sin(3*pi*D) * cos(2*pi*3*f*t);
v4 = A4 * sin(4*pi*D) * cos(2*pi*4*f*t);
v5 = A5 * sin(5*pi*D) * cos(2*pi*5*f*t);
v6 = A6 * sin(6*pi*D) * cos(2*pi*6*f*t);
v7 = A7 * sin(7*pi*D) * cos(2*pi*7*f*t);
v8 = A8 * sin(8*pi*D) * cos(2*pi*8*f*t);
v9 = A9 * sin(9*pi*D) * cos(2*pi*9*f*t);
v10 = A10 * sin(10*pi*D) * cos(2*pi*10*f*t);
v11 = A11 * sin(11*pi*D) * cos(2*pi*11*f*t);
% Señal compuesta original
VT = v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 +
v9 + v10 + v11;
% - GRÁFICA EN TIEMPO ORIGINAL ------------
figure;
plot(t, VT);
xlabel('Tiempo (s)');
ylabel('Amplitud');
title('Señal PULSO - CICLO ÚTIL 80 %');
grid on;
% ------ FFT Y ESPECTRO ------------
N = length(VT);   % Número de muestras
Y = fft(VT);% Transformada rápida de Fourier
Y_mag = abs(Y) / N; % Magnitud normalizada
Y_mag(2:end-1) = 2 * Y_mag(2:end-1);  %
Duplicar menos DC y Nyquist
Y_dB = 20*log10(Y_mag + eps); % Evita log(0)
% Eje de frecuencias
f_axis = (0:N-1)*(Fs/N);
% -GRÁFICA EN FRECUENCIA ------------
figure;
plot(f_axis(1:N/2), Y_dB(1:N/2), 'b',
'LineWidth', 1.3);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud (dB)');
title('Espectro de Frecuencia - Pulso con
ciclo útil 30%');
grid on;
xlim([0 20000]);
ylim([-80 5]);
xticks(0:1000:20000);
% - IFFT Y SEÑAL RECONSTRUIDA ------------
vt_rec = ifft(Y, 'symmetric');    %
Reconstrucción en tiempo (real)
% --- GRÁFICA SEÑAL RECONSTRUIDA -----------
figure;
```

```
plot(t*1000, vt_rec, 'r');   % t en
milisegundos para mejor visualización
xlabel('Tiempo (ms)');
ylabel('Amplitud (V aprox)');
title('Señal reconstruida en el dominio del
tiempo (IFFT)');
grid on;
```

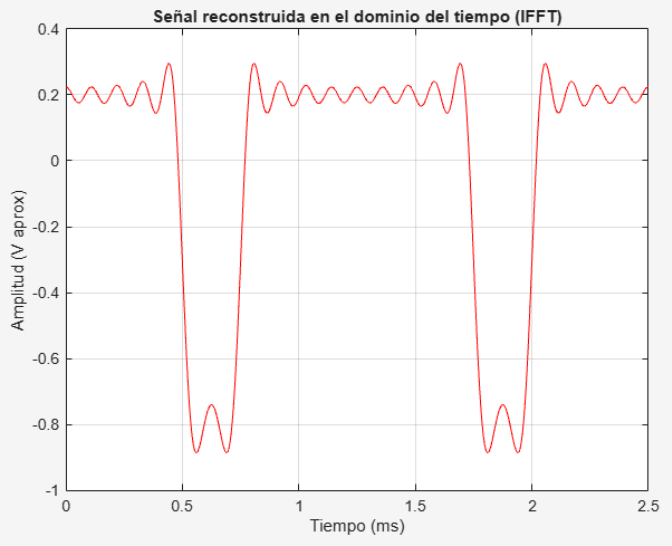


Ilustración 78 Grafica Generada Por Matlab IFFT

La gráfica muestra un pulso típico de una señal con componentes armónicas múltiples, resultado de la expansión en serie de Fourier para un pulso rectangular con ciclo útil distinto del 50%, el Ciclo útil del 80% significa que el pulse activo está solo en el 80% del ciclo completo, por lo que la forma general es similar a un pulso estrecho

Se crea un repositorio en GitHub donde se suben las demás evidencias del desarrollo del laboratorio al cual se podrá acceder en este enlace: https://github.com/fernandoriano/LABORATORIO2-COMUNICACION-DIGITAL.git

II. CONCLUSIÓN ES

Las señales reconstruidas a partir de armónicos teóricos mostraron una gran similitud con las señales obtenidas experimentalmente en el laboratorio, validando así la aplicabilidad de las series de Fourier en el análisis de señales periódicas.

El uso conjunto del osciloscopio digital, MATLAB y Python facilitó una visualización precisa de las señales en los dominios del tiempo y la frecuencia. Esto permitió contrastar las representaciones gráficas experimentales con los modelos teóricos, fortaleciendo el aprendizaje práctico en procesamiento de señales.

Se evidenció que la presencia o ausencia de componente continua (DC) afecta directamente el desplazamiento vertical de la señal en el dominio del tiempo, lo cual se refleja en la aparición de un pico en la frecuencia cero del espectro.

Las señales de tipo pulso con ciclos útiles distintos al 50% mostraron contenido armónico tanto en múltiplos pares como impares de la frecuencia fundamental. A medida que el ciclo útil se aleja del 50%, se amplía la dispersión espectral, lo que influye directamente en la fidelidad de la reconstrucción de la señal.

La aplicación de la Transformada Rápida de Fourier (FFT) permitió obtener de manera eficiente el espectro de frecuencia de cada señal. Posteriormente, el uso de la IFFT, aunque limitado por la falta de fase en los archivos .csv, fue útil para aproximar las formas originales de las señales y verificar la consistencia del análisis espectral.

La tasa de muestreo del osciloscopio (500 kHz) fue adecuada para capturar señales de hasta 800 Hz, permitiendo obtener una representación fiel en ambos dominios. Esto ratifica la importancia de cumplir el teorema de Nyquist para evitar distorsiones.

Este laboratorio permitió no solo afianzar conceptos fundamentales del curso de comunicaciones digitales, sino también desarrollar habilidades en el manejo de instrumentos, análisis de datos, programación, y pensamiento crítico frente a la interpretación de fenómenos reales y su modelado matemático.

Aunque las señales teóricas y experimentales fueron similares, se identificaron ligeras discrepancias atribuibles al ruido, errores de medición, o la naturaleza finita de la serie de Fourier utilizada para la reconstrucción. Estos elementos subrayan la importancia de considerar factores reales en el diseño de sistemas de comunicación.

III. REFERENCIAS

[1] MathWorks, "Transformada Rápida de Fourier - FFT," MATLAB Documentation, 2024. [En línea]. Disponible en: https://www.mathworks.com/help/matlab/ref/fft.html

[2] MathWorks, "Transformada Inversa Rápida de Fourier - IFFT," MATLAB Documentation, 2024. [En línea]. Disponible en: https://www.mathworks.com/help/matlab/ref/ifft.html

[3] M. Oppenheim y A. Willsky, Señales y sistemas, 2ª ed., Prentice Hall, 1997.

[4] A. V. Oppenheim, R. W. Schafer, y J. R. Buck, Discrete-Time Signal Processing, 3ª ed., Prentice Hall, 2009.

[5] Lathi, B. P., Linear Systems and Signals, 2ª ed., Oxford University Press, 2005.

[6] OpenAI. (2025). Respuesta generada por ChatGPT sobre simulación de señales y espectros en MATLAB. Comunicación en línea, 6 de agosto de 2025.

[7] J. G. Proakis y D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 4ª ed., Pearson, 2007.

[8] T. K. Rawat, Signal Processing and Linear Systems, Oxford University Press, 2015.

[9] Universidad Nacional de Colombia, "Análisis espectral de señales periódicas usando series de Fourier y transformadas," Curso de Señales y Sistemas, 2022. [En línea]. Disponible en: https://www.virtual.unal.edu.co/cursos/ingenieria/4025051/pdf

[10] MATLAB Central. (2023). "Uso de FFT para análisis de espectros de señales compuestas", Foro de soporte técnico. [En línea]. Disponible en: https://www.mathworks.com/matlabcentral/answers/