UNIVERSIDAD VERACRUZANA
FACULTAD DE INSTRUMENTACION ELECTRONICA
XALAPA, VERACRUZ.

REPORTE

Practica 4 – Graficas con Python

MATERIA

Programación Avanzada.

DOCENTE

José Alfonso Domínguez Chávez.

PROGRAMA EDUCATIVO

Ingeniería En Instrumentación Electrónica.

EQUIPO

- Christian Sánchez Hernández.
- Luis Gerardo Pérez Hernández.
- Oswaldo Roa Herrera.
- Saín David Brígido Mora.

Xalapa, Ver. a 12 de mayo de 2025

INTRODUCCION.

En este reporte tiene como objetivo plasmar la documentación del proceso de filtrado y análisis de datos provenientes de sensores de temperatura, humedad y viento, utilizando herramientas de procesamiento digital de señales en Python.

La actividad consistió en la carga, visualización y procesamiento de datos contenidos en archivos CSV, esto con la finalidad de mejorar la calidad de las señales mediante técnicas como el promediado móvil, filtros para bajas y pasa bandas, y el análisis espectral mediante la Transformada Rápida de Fourier.

Cada etapa del procesamiento fue acompañada de representaciones graficas utilizando la biblioteca “matplolib”, esta con el objetivo de visualizar y comparar los efectos de los distintos filtros aplicados sobre las señales aplicadas originales. Se estableció una escala de tiempo basada en una tasa de muestreo de 0.2 muestras por segundo (un dato cada 5 segundos) para contextualizar temporalmente las señales.

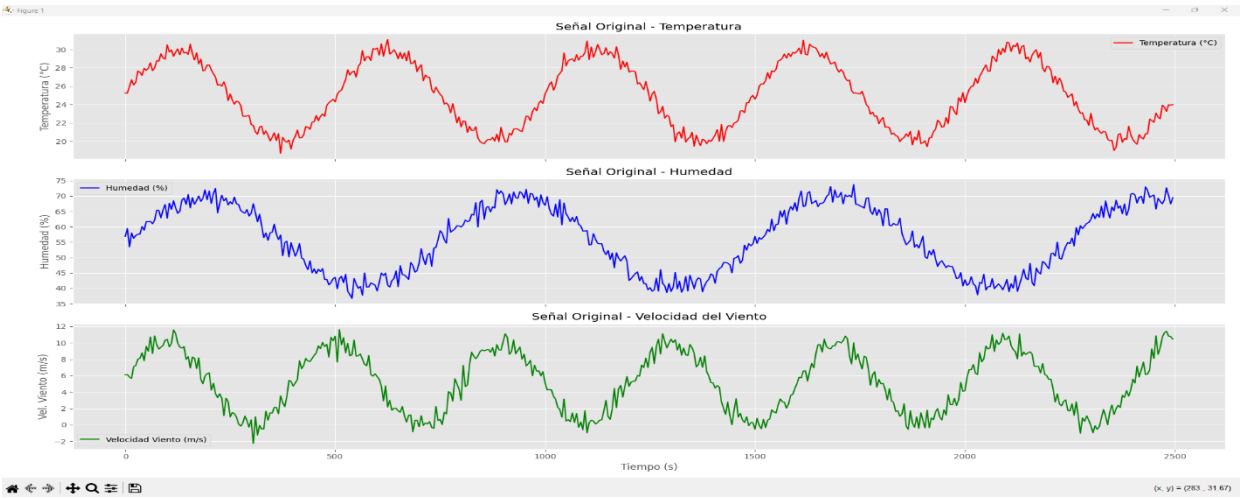
El propósito principal de esta práctica consistió en desarrollar habilidades practicas en el tratamiento de señales digitales, identificar el ruido presente en los datos, y seleccionar el método de filtrado mas adecuado para su atenuación, justificando la elección con base en el análisis en el dominio de frecuencia.

Paso 1. Carga De Datos.

En este primer paso, se procedió a cargar los archivos de datos proporcionados por el docente (temperatura, humedad y viento), utilizando pandas en Python. Una vez cargados, se ajusto la escala temporal a una tasa de muestreo de 0.2mps. Para ello, se generó un eje de tiempo correspondiente a la longitud de cada conjunto, tomando en cuenta el intervalo de muestreo mencionado.

Después de ajustar estos detalles, se realizó la visualización de las señales originales mediante matplolib, generando una grafica individual para cada conjunto de señales. Estas graficas muestra el comportamiento de las señales en su estado crudo, sin ningún procesamiento previo.

A continuación, presentamos los resultados:




```python
# AnalisisSensores.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt
from scipy.fft import fft, fftfreq

# OPCIONAL: Estilo (el que sí existe)
plt.style.use('ggplot')

# Cargar los archivos CSV
df_temp = pd.read_csv('temperatura.csv')
df_hum = pd.read_csv('humedad.csv')
df_viento = pd.read_csv('viento.csv')

# Crear el eje de tiempo (un dato cada 5 segundos)
# Suponemos que todos los archivos tienen la misma cantidad de muestras
n_muestras = len(df_temp)
tiempo = np.arange(0, n_muestras * 5, 5)  # [0, 5, 10, ..., N*5]

# --- Graficar señales originales ---
fig, axs = plt.subplots(3, 1, figsize=(12, 10), sharex=True)

# 1. Temperatura
axs[0].plot(tiempo, df_temp.iloc[:, 1], label='Temperatura (°C)',
color='red')
axs[0].set_ylabel('Temperatura (°C)')
axs[0].set_title('Señal Original - Temperatura')
axs[0].legend()
axs[0].grid(True)
```

```
# 2. Humedad
axs[1].plot(tiempo, df_hum.iloc[:, 1], label='Humedad (%)', color='blue')
axs[1].set_ylabel('Humedad (%)')
axs[1].set_title('Señal Original - Humedad')
axs[1].legend()
axs[1].grid(True)

# 3. Viento (Velocidad del viento)
axs[2].plot(tiempo, df_viento['Velocidad_Viento_mps'], label='Velocidad
Viento (m/s)', color='green')
axs[2].set_ylabel('Vel. Viento (m/s)')
axs[2].set_xlabel('Tiempo (s)')
axs[2].set_title('Señal Original - Velocidad del Viento')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

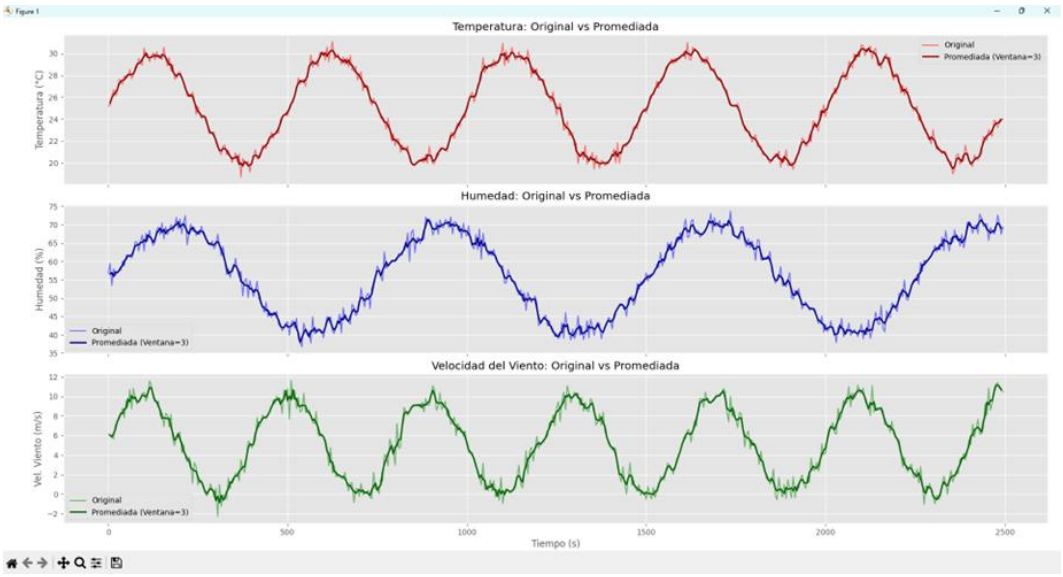
El objetivo principal de este paso fue familiarizarse con las señales originales y establecer una referencia base para comparar con las señales procesadas en los pasos posteriores. Visualizar los datos sin procesar permite identificar tendencias generales, posibles fluctuaciones o ruido y nivel de variabilidad.

Paso 2. Promediado Movil.

En esta etapa se implementó una técnica básica de suavizado de señales promediado móvil, la cual tiene como finalidad atenuar pequeñas fluctuaciones o ruido de alta frecuencia en los datos.

Se aplicó un promediado móvil con una ventana de 3 muestras a cada una de las señales. Este procedimiento consiste en reemplazar cada valor de la señal por el promedio de él mismo y sus dos vecinos más cercanos (anteriores o centrados, según la implementación). El efecto de esta operación es suavizar los cambios bruscos y reducir el ruido sin alterar significativamente la forma general de la señal.

Posteriormente, se graficaron tanto la señal original como la señal suavizada en una misma figura para cada sensor. Esto permitió una comparación visual directa del efecto del promediado móvil, facilitando la evaluación de su eficacia.




```python
# AnalisisSensores.py - Paso 2: Promediado Móvil

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt
from scipy.fft import fft, fftfreq

# OPCIONAL: Estilo (el que sí existe)
plt.style.use('ggplot')

# Cargar los archivos CSV
df_temp = pd.read_csv('temperatura.csv')
df_hum = pd.read_csv('humedad.csv')
df_viento = pd.read_csv('viento.csv')

# Crear el eje de tiempo (un dato cada 5 segundos)
n_muestras = len(df_temp)
tiempo = np.arange(0, n_muestras * 5, 5)  # [0, 5, 10, ..., N*5]

# --- Función para promediado móvil ---
def moving_average(data, window_size=3):
    return np.convolve(data, np.ones(window_size)/window_size, mode='valid')

# Aplicar promediado móvil a cada sensor
temp_ma = moving_average(df_temp.iloc[:, 1])
hum_ma = moving_average(df_hum.iloc[:, 1])
viento_ma = moving_average(df_viento['Velocidad_Viento_mps'])
```

```
# Ajustar el tiempo para el promediado móvil (pierde muestras en los
extremos)
tiempo_ma = tiempo[1:-1]  # Elimina primera y última muestra por 'valid'
mode

# --- Graficar señales originales vs promediadas ---
fig, axs = plt.subplots(3, 1, figsize=(12, 12), sharex=True)

# 1. Temperatura
axs[0].plot(tiempo, df_temp.iloc[:, 1], label='Original', color='red',
alpha=0.5)
axs[0].plot(tiempo_ma, temp_ma, label='Promediada (Ventana=3)',
color='darkred', linewidth=2)
axs[0].set_ylabel('Temperatura (°C)')
axs[0].set_title('Temperatura: Original vs Promediada')
axs[0].legend()
axs[0].grid(True)

# 2. Humedad
axs[1].plot(tiempo, df_hum.iloc[:, 1], label='Original', color='blue',
alpha=0.5)
axs[1].plot(tiempo_ma, hum_ma, label='Promediada (Ventana=3)',
color='darkblue', linewidth=2)
axs[1].set_ylabel('Humedad (%)')
axs[1].set_title('Humedad: Original vs Promediada')
axs[1].legend()
axs[1].grid(True)

# 3. Viento
axs[2].plot(tiempo, df_viento['Velocidad_Viento_mps'], label='Original',
color='green', alpha=0.5)
axs[2].plot(tiempo_ma, viento_ma, label='Promediada (Ventana=3)',
color='darkgreen', linewidth=2)
axs[2].set_ylabel('Vel. Viento (m/s)')
axs[2].set_xlabel('Tiempo (s)')
axs[2].set_title('Velocidad del Viento: Original vs Promediada')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

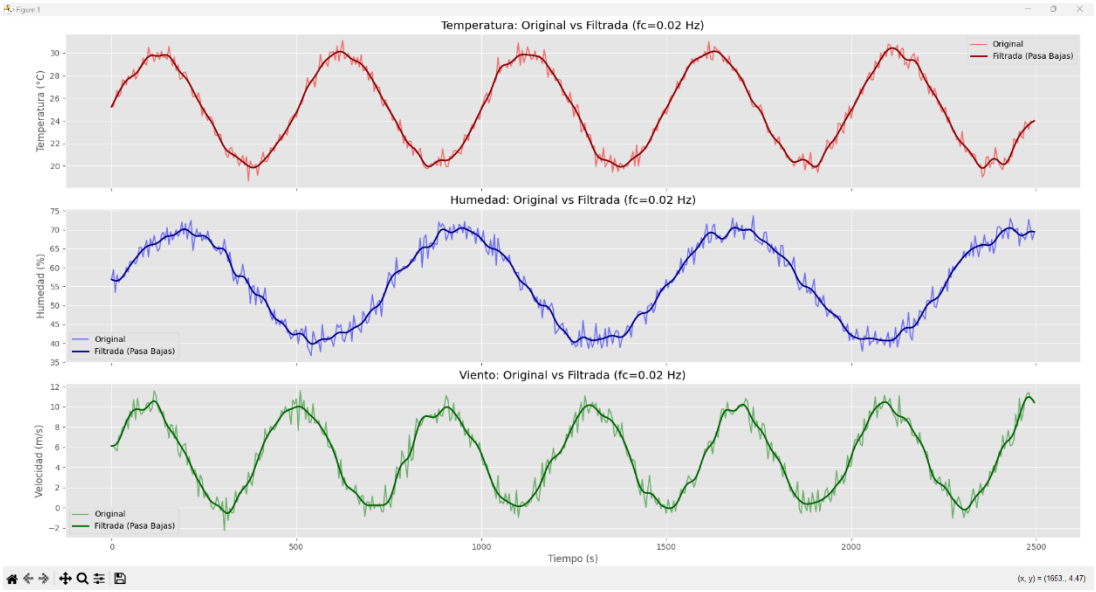
El objetivo de este paso fue aplicar un suavizado inicial a las señales para eliminar pequeñas perturbaciones o ruido no deseado, sin necesidad de diseñar un filtro complejo. El promediado móvil sirve como una técnica sencilla pero efectiva para mejorar la legibilidad de los datos y preparar las señales para un análisis más profundo en los siguientes pasos.

Paso 3. Aplicar Filtro Pasa Bajas.

En esta etapa se implementó un filtro digital pasa bajas con el objetivo de atenuar las componentes de alta frecuencia presentes en las señales, es decir, reducir el ruido y conservar únicamente las variaciones lentas o más significativas de cada medición.

Para ello, se utilizó la función butter () de la biblioteca scipy. signal para diseñar un filtro de tipo Butterworth de orden adecuado, y la función filtfilt () para aplicar dicho filtro en ambas direcciones (adelante y atrás), asegurando una respuesta sin desfase temporal.

La frecuencia de corte elegida fue de 0.1 en frecuencia normalizada, lo cual significa que se eliminaron las frecuencias superiores a ese valor, manteniendo solo las bajas frecuencias asociadas a las tendencias principales de las señales.



```python
# AnalisisSensores.py - Paso 3: Filtro Pasa Bajas (Versión Corregida)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# Configuración de estilo (opcional)
plt.style.use('ggplot')

# --- Cargar datos ---
df_temp = pd.read_csv('temperatura.csv')
df_hum = pd.read_csv('humedad.csv')
df_viento = pd.read_csv('viento.csv')

# Crear eje de tiempo (1 dato cada 5 segundos)
tiempo = np.arange(0, len(df_temp) * 5, 5)

# --- Diseño del filtro pasa bajas ---
def butter_lowpass_filter(data, cutoff_freq=0.02, fs=0.2, order=4):
    """
    Filtro Butterworth pasa bajas.
    - cutoff_freq: Frecuencia de corte real (Hz), debe ser menor que fs/2.
    - fs: Frecuencia de muestreo (0.2 Hz = 1 muestra/5s).
    - order: Orden del filtro (default: 4).
    """
    nyquist = 0.5 * fs
    normal_cutoff = cutoff_freq / nyquist  # Normalización correcta
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    filtered_data = filtfilt(b, a, data)
    return filtered_data

# Aplicar filtro a cada sensor
temp_filt = butter_lowpass_filter(df_temp.iloc[:, 1])
hum_filt = butter_lowpass_filter(df_hum.iloc[:, 1])
viento_filt = butter_lowpass_filter(df_viento['Velocidad_Viento_mps'])

# --- Gráficas comparativas ---
fig, axs = plt.subplots(3, 1, figsize=(12, 12), sharex=True)
```

```
# 1. Temperatura
axs[0].plot(tiempo, df_temp.iloc[:, 1], label='Original', color='red',
alpha=0.5)
axs[0].plot(tiempo, temp_filt, label='Filtrada (Pasa Bajas)',
color='darkred', linewidth=2)
axs[0].set_ylabel('Temperatura (°C)')
axs[0].set_title('Temperatura: Original vs Filtrada (fc=0.02 Hz)')
axs[0].legend()
axs[0].grid(True)

# 2. Humedad
axs[1].plot(tiempo, df_hum.iloc[:, 1], label='Original', color='blue',
alpha=0.5)
axs[1].plot(tiempo, hum_filt, label='Filtrada (Pasa Bajas)',
color='darkblue', linewidth=2)
axs[1].set_ylabel('Humedad (%)')
axs[1].set_title('Humedad: Original vs Filtrada (fc=0.02 Hz)')
axs[1].legend()
axs[1].grid(True)

# 3. Viento
axs[2].plot(tiempo, df_viento['Velocidad_Viento_mps'], label='Original',
color='green', alpha=0.5)
axs[2].plot(tiempo, viento_filt, label='Filtrada (Pasa Bajas)',
color='darkgreen', linewidth=2)
axs[2].set_ylabel('Velocidad (m/s)')
axs[2].set_xlabel('Tiempo (s)')
axs[2].set_title('Viento: Original vs Filtrada (fc=0.02 Hz)')
axs[2].legend()
axs[2].grid(True)
plt.tight_layout()
plt.show()
```

El objetivo principal fue reducir el ruido de alta frecuencia que puede interferir con el análisis de las señales, conservando las variaciones más relevantes y lentas. El uso de un filtro pasa bajas ayuda a eliminar fluctuaciones bruscas o artefactos que no representan cambios reales en los fenómenos físicos medidos.

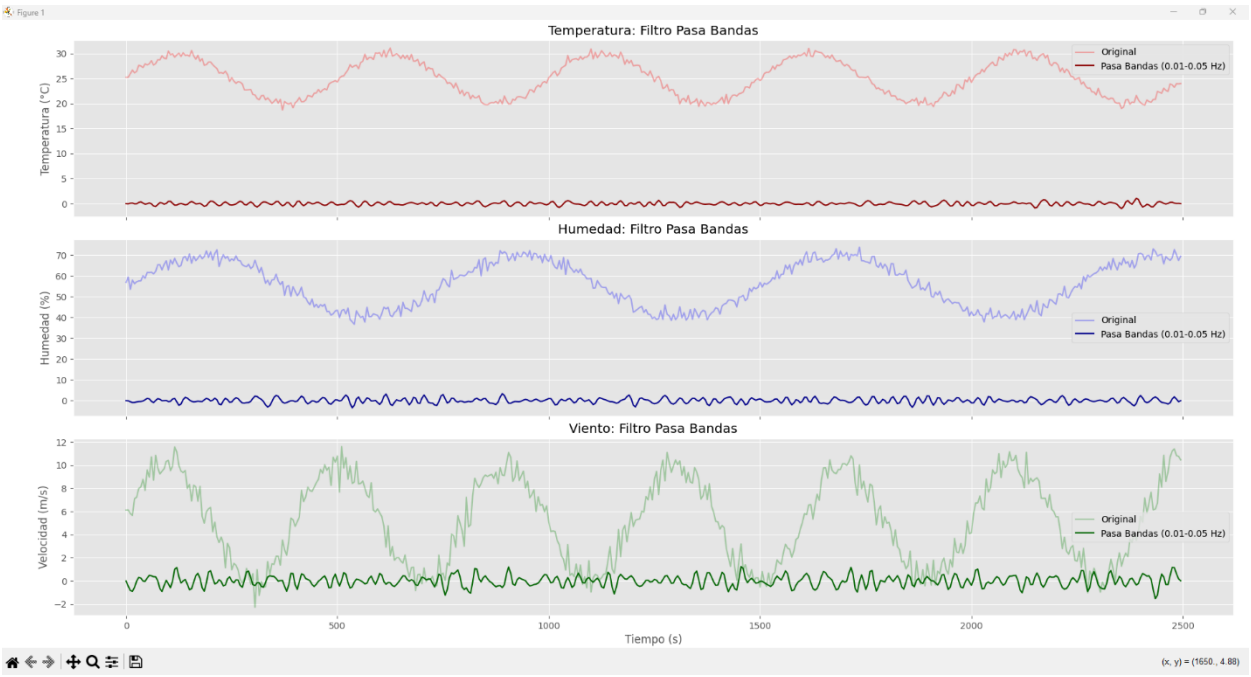
Este paso fue clave para preparar las señales para un análisis más fino (como el análisis espectral), y para evaluar cómo distintos tipos de filtrado afectan la calidad y la interpretación de los datos sensoriales.

Paso 4. Aplicar Filtro Pasa Bandas.

En este paso se implementó un filtro pasa bandas con el objetivo de aislar ciertas frecuencias específicas presentes en las señales, permitiendo el paso de componentes dentro de un rango determinado y atenuando tanto las bajas como las altas frecuencias fuera de ese rango.

Para ello, se utilizó nuevamente la función butter () de scipy. signal, pero esta vez configurada como filtro pasa bandas, especificando un rango de frecuencias (frecuencia inferior y superior normalizadas) que se deseaba conservar. Posteriormente, el filtro se aplicó utilizando filtfilt () para evitar el desfase de fase y obtener una señal más precisa.

Este tipo de filtrado permite resaltar oscilaciones o patrones periódicos específicos que podrían estar ocultos por otras componentes de la señal o ruido. Se aplicó individualmente a las señales de temperatura, humedad y viento.




```python
# AnalisisSensores.py - Paso 4: Filtro Pasa Bandas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# Configuración de estilo
plt.style.use('ggplot')
```

```
# --- Cargar datos ---
df_temp = pd.read_csv('temperatura.csv')
df_hum = pd.read_csv('humedad.csv')
df_viento = pd.read_csv('viento.csv')

# Crear eje de tiempo (1 dato cada 5 segundos)
tiempo = np.arange(0, len(df_temp) * 5, 5)

# --- Diseño del filtro pasa bandas ---
def butter_bandpass_filter(data, lowcut=0.01, highcut=0.05, fs=0.2,
order=4):
    """
    Filtro Butterworth pasa bandas.
    - lowcut: Frecuencia inferior (Hz)
    - highcut: Frecuencia superior (Hz)
    - fs: Frecuencia de muestreo (0.2 Hz)
    - order: Orden del filtro
    """
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = butter(order, [low, high], btype='band', analog=False)
    filtered_data = filtfilt(b, a, data)
    return filtered_data

# Aplicar filtro pasa bandas a cada sensor
temp_band = butter_bandpass_filter(df_temp.iloc[:, 1])
hum_band = butter_bandpass_filter(df_hum.iloc[:, 1])
viento_band = butter_bandpass_filter(df_viento['Velocidad_Viento_mps'])

# --- Gráficas comparativas ---
fig, axs = plt.subplots(3, 1, figsize=(12, 12), sharex=True)

# 1. Temperatura
axs[0].plot(tiempo, df_temp.iloc[:, 1], label='Original', color='red',
alpha=0.3)
axs[0].plot(tiempo, temp_band, label='Pasa Bandas (0.01-0.05 Hz)',
color='darkred')
axs[0].set_ylabel('Temperatura (°C)')
axs[0].set_title('Temperatura: Filtro Pasa Bandas')
axs[0].legend()
axs[0].grid(True)
```

```
# 2. Humedad
axs[1].plot(tiempo, df_hum.iloc[:, 1], label='Original', color='blue',
alpha=0.3)
axs[1].plot(tiempo, hum_band, label='Pasa Bandas (0.01-0.05 Hz)',
color='darkblue')
axs[1].set_ylabel('Humedad (%)')
axs[1].set_title('Humedad: Filtro Pasa Bandas')
axs[1].legend()
axs[1].grid(True)

# 3. Viento
axs[2].plot(tiempo, df_viento['Velocidad_Viento_mps'], label='Original',
color='green', alpha=0.3)
axs[2].plot(tiempo, viento_band, label='Pasa Bandas (0.01-0.05 Hz)',
color='darkgreen')
axs[2].set_ylabel('Velocidad (m/s)')
axs[2].set_xlabel('Tiempo (s)')
axs[2].set_title('Viento: Filtro Pasa Bandas')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

El objetivo fue extraer información específica del contenido en frecuencia de las señales, especialmente útil cuando se sospecha que existe una componente periódica o fenómeno particular dentro de un cierto rango de frecuencias.

Este paso complementa el filtrado pasa bajas del paso anterior, ya que permite analizar por separado bandas de frecuencia para identificar comportamientos que no son evidentes en el dominio del tiempo o cuando se considera la señal completa. Así, el filtrado pasa bandas ayuda a detectar patrones repetitivos o vibraciones, y facilita la posterior comparación con el análisis en frecuencia realizado en el paso 5.

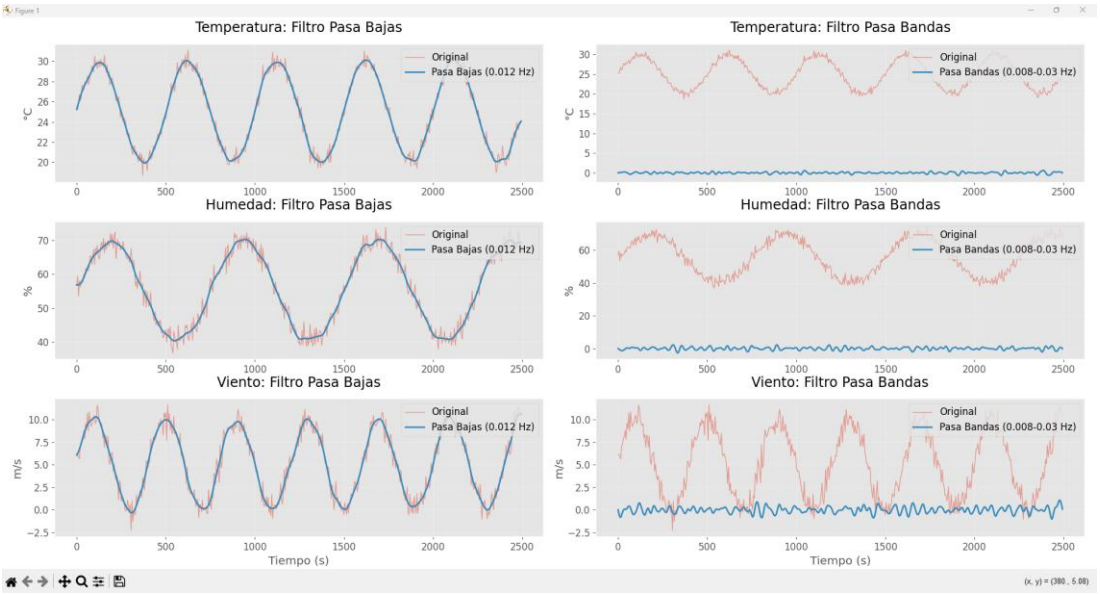
Paso 5. Análisis de Fourier.

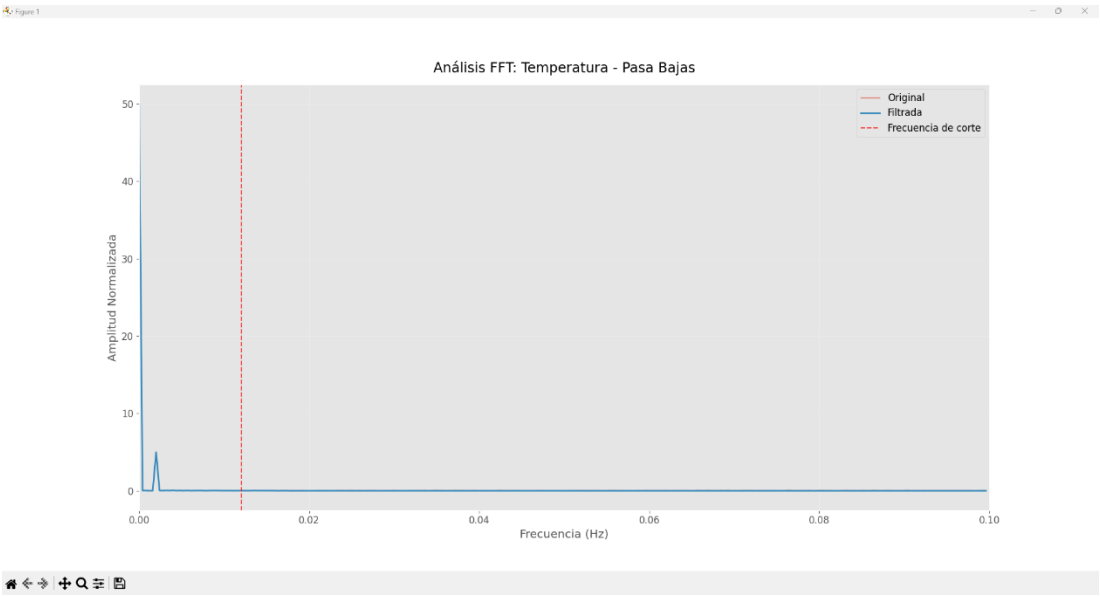
En esta etapa se aplicó la Transformada Rápida de Fourier (FFT) a las señales originales y a las señales filtradas (tanto con el filtro pasa bajas como con el filtro pasa bandas), con el fin de analizar su contenido en frecuencia.

La FFT convierte las señales del dominio del tiempo al dominio de la frecuencia, permitiendo visualizar cómo se distribuye la energía de la señal en distintas frecuencias. Esta herramienta es fundamental para identificar componentes periódicas, tendencias dominantes y presencia de ruido de alta frecuencia.

Para cada sensor (temperatura, humedad y viento), se calculó y graficó el espectro de frecuencias correspondiente a:

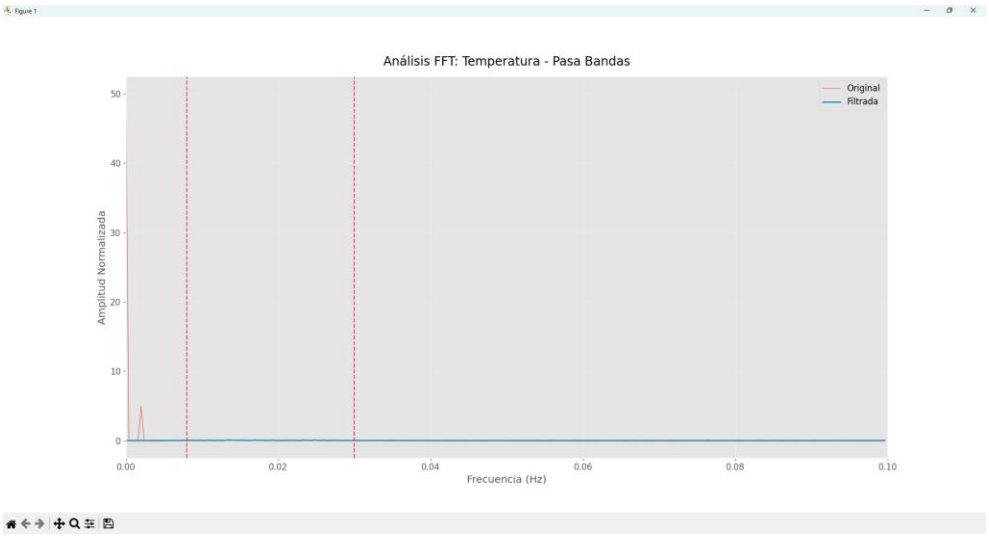
- La señal original.
- La señal filtrada con el filtro pasa bajas.
- La señal filtrada con el filtro pasa bandas.

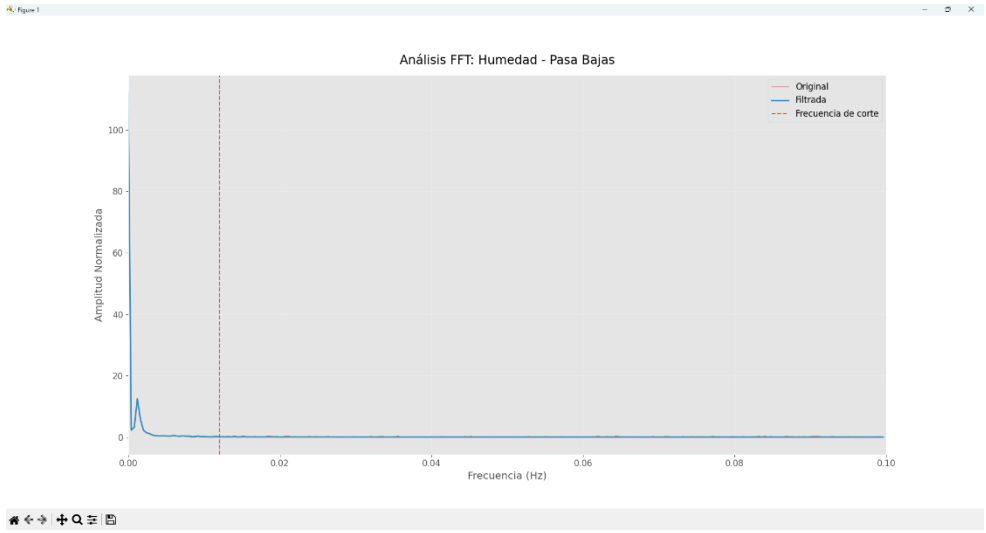


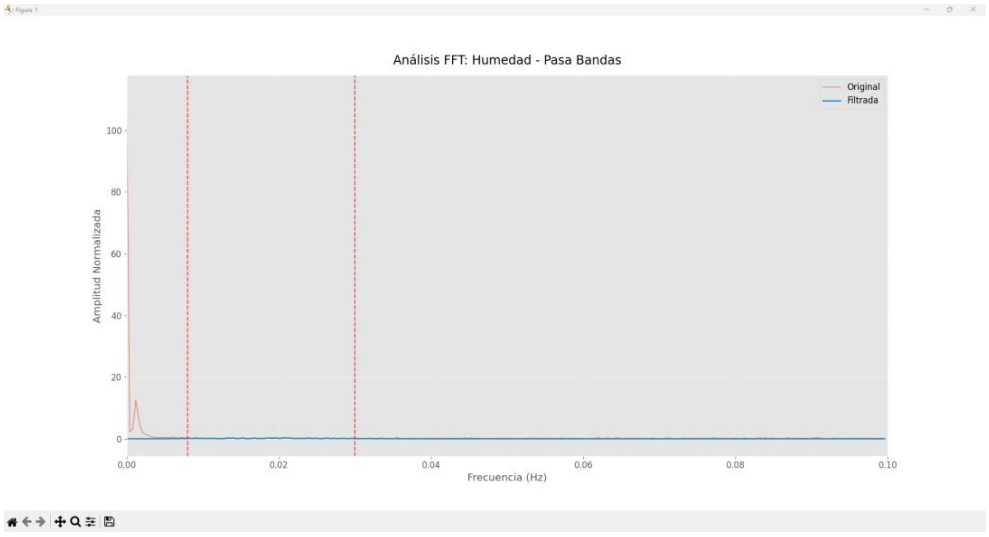


Xalapa, Ver. a 12 de mayo de 2025

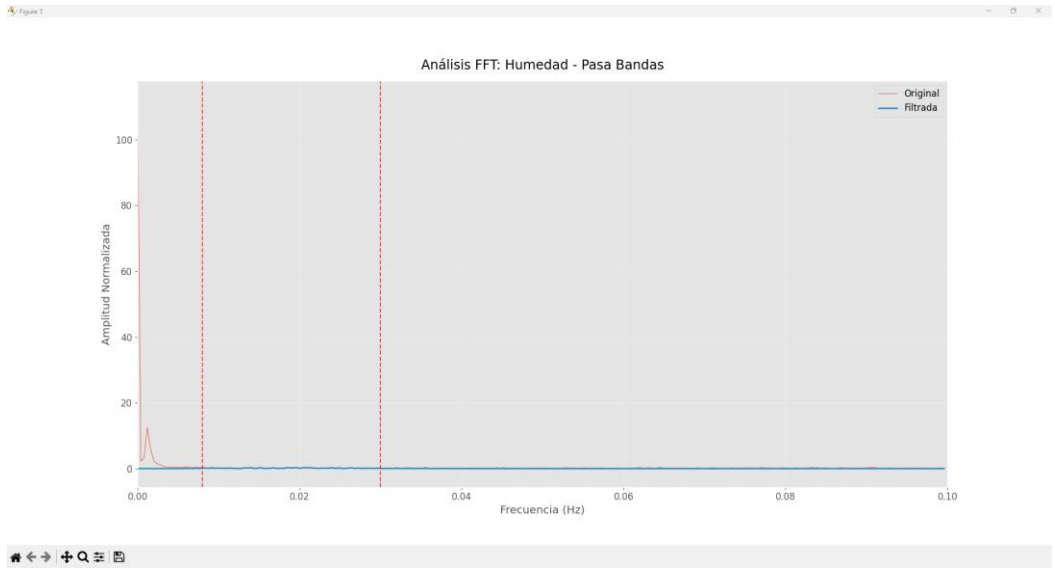
UNIVERSIDAD VERACRUZANA
FACULTAD DE INSTRUMENTACION ELECTRONICA
XALAPA, VERACRUZ.

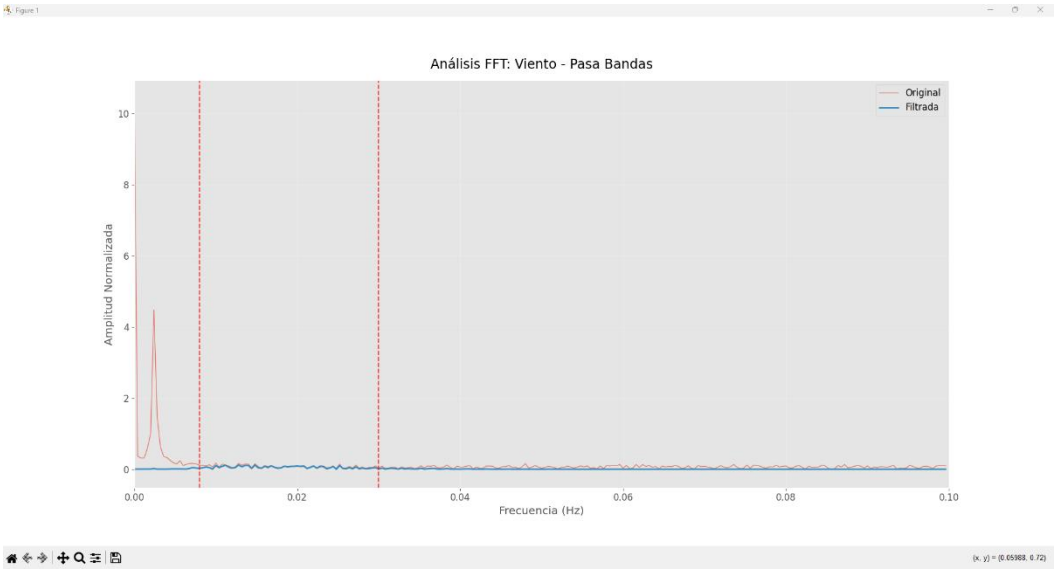






Xalapa, Ver. a 12 de mayo de 2025






```python
# AnalisisSensores.py - Versión Final Ajustada
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt
from scipy.fft import fft, fftfreq

# Configuración
plt.style.use('ggplot')
plt.rcParams['font.size'] = 12
```

```
# Cargar datos
df_temp = pd.read_csv('temperatura.csv')
df_hum = pd.read_csv('humedad.csv')
df_viento = pd.read_csv('viento.csv')

# Eje de tiempo
n_muestras = len(df_temp)
tiempo = np.arange(0, n_muestras * 5, 5)
fs = 0.2  # 0.2 Hz = 1 muestra/5s

# --- Funciones de filtro optimizadas ---
def butter_lowpass(signal, cutoff=0.012, fs=fs, order=2):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return filtfilt(b, a, signal)

def butter_bandpass(signal, low=0.008, high=0.03, fs=fs, order=2):
    nyq = 0.5 * fs
    low_norm = low / nyq
    high_norm = high / nyq
    b, a = butter(order, [low_norm, high_norm], btype='band')
    return filtfilt(b, a, signal)

# --- Aplicar filtros ---
senales = {
    'Temperatura': df_temp.iloc[:, 1],
    'Humedad': df_hum.iloc[:, 1],
    'Viento': df_viento['Velocidad_Viento_mps']
}

filtradas = {
    'Temperatura LP': butter_lowpass(senales['Temperatura']),
    'Humedad LP': butter_lowpass(senales['Humedad']),
    'Viento LP': butter_lowpass(senales['Viento']),
    'Temperatura BP': butter_bandpass(senales['Temperatura']),
    'Humedad BP': butter_bandpass(senales['Humedad']),
    'Viento BP': butter_bandpass(senales['Viento'])
}

# --- Visualización Mejorada ---
fig, axs = plt.subplots(3, 2, figsize=(16, 12))
```

```
# Configuración común
def setup_plot(ax, title, ylabel, xlabel=False):
    ax.set_title(title, pad=15)
    ax.set_ylabel(ylabel)
    if xlabel: ax.set_xlabel('Tiempo (s)')
    ax.grid(True, alpha=0.3)
    ax.legend(loc='upper right')

# Gráficas
for i, (nombre, señal) in enumerate(senales.items()):
    # Pasa bajas
    axs[i,0].plot(tiempo, señal, label='Original', alpha=0.5, linewidth=1)
    axs[i,0].plot(tiempo, filtradas[f'{nombre} LP'],
                  label=f'Pasa Bajas (0.012 Hz)', linewidth=2)
    setup_plot(axs[i,0], f'{nombre}: Filtro Pasa Bajas',
               '°C' if nombre=='Temperatura' else '%' if nombre=='Humedad'
else 'm/s',
               xlabel=(i==2))

    # Pasa bandas
    axs[i,1].plot(tiempo, señal, label='Original', alpha=0.5, linewidth=1)
    axs[i,1].plot(tiempo, filtradas[f'{nombre} BP'],
                  label='Pasa Bandas (0.008-0.03 Hz)', linewidth=2)
    setup_plot(axs[i,1], f'{nombre}: Filtro Pasa Bandas',
               '°C' if nombre=='Temperatura' else '%' if nombre=='Humedad'
else 'm/s',
               xlabel=(i==2))

plt.tight_layout()
plt.show()

# --- Análisis FFT Automatizado ---
def plot_fft_comparison(signal, filtered, title):
    freqs = fftfreq(n_muestras, 5)[:n_muestras//2]
    fft_orig = np.abs(fft(signal)[:n_muestras//2]) * 2/n_muestras
    fft_filt = np.abs(fft(filtered)[:n_muestras//2]) * 2/n_muestras

    plt.figure(figsize=(12, 4))
    plt.plot(freqs, fft_orig, label='Original', alpha=0.7, linewidth=1)
    plt.plot(freqs, fft_filt, label='Filtrada', linewidth=2)
```

```
if 'Pasa Bajas' in title:
        plt.axvline(0.012, color='red', linestyle='--', alpha=0.7,
label='Frecuencia de corte')
    else:
        plt.axvline(0.008, color='red', linestyle='--', alpha=0.7)
        plt.axvline(0.03, color='red', linestyle='--', alpha=0.7)

    plt.title(f'Análisis FFT: {title}', pad=15)
    plt.xlabel('Frecuencia (Hz)')
    plt.ylabel('Amplitud Normalizada')
    plt.xlim(0, 0.1)
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()

# Generar FFTs comparativos
for nombre in senales.keys():
    plot_fft_comparison(senales[nombre], filtradas[f'{nombre} LP'],
                        f'{nombre} - Pasa Bajas')
    plot_fft_comparison(senales[nombre], filtradas[f'{nombre} BP'],
                        f'{nombre} - Pasa Bandas')
```

El objetivo principal fue analizar y comparar las señales en el dominio de la frecuencia, para comprender mejor cómo se compone cada señal y qué tan efectivas fueron las técnicas de filtrado aplicadas en los pasos anteriores.
Mediante la FFT, se pudo:

- Identificar las frecuencias dominantes en cada señal.
- Localizar el ruido como componentes de alta frecuencia.
- Evaluar la efectividad de los filtros pasa bajas y pasa bandas, observando qué partes del espectro fueron eliminadas o conservadas.

Este análisis proporcionó una base sólida para justificar qué filtro fue más adecuado para cada tipo de señal, de acuerdo con el tipo de información que se deseaba conservar o eliminar.

CONCLUSION.

Durante esta actividad se llevó a cabo el análisis y filtrado de señales provenientes de sensores de temperatura, humedad y viento, utilizando herramientas de Python como pandas, matplotlib y scipy. El proceso incluyó la carga y visualización de los datos originales, seguidos de la aplicación de diferentes técnicas de filtrado y análisis espectral.

El promediado móvil permitió suavizar las señales y reducir pequeñas fluctuaciones, mientras que el filtro pasa bajas eliminó componentes de alta frecuencia, conservando las variaciones principales. Por otro lado, el filtro pasa bandas sirvió para aislar frecuencias específicas de interés. Finalmente, la Transformada Rápida de Fourier (FFT) permitió visualizar el contenido en frecuencia de las señales originales y filtradas, facilitando la comparación y evaluación de los resultados.

Gracias a este proceso, fue posible identificar la naturaleza del ruido presente en los datos y determinar qué tipo de filtrado resultó más efectivo para cada caso. La actividad permitió consolidar conocimientos fundamentales sobre el procesamiento digital de señales y su aplicación práctica en el análisis de datos reales.