UNIVERSIDAD NACIONAL DE TUCUMÁN

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

SEÑALES ELÉCTRICAS

TP N°14: Conversor analógico digital, muestreo de señales y análisis de señales en el dominio del tiempo y la frecuencia.

INTEGRANTES:

ROSALES, FACUNDO EZEQUIEL	ING. ELECTRÓNICA
BRITO, GUILLERMO NICOLÁS	ING. ELECTRÓNICA
DE ANGELIS, NICOLAS	ING. ELECTRÓNICA

AÑO 2024

Frecuencia de Muestreo del ADC BluePill

En este proyecto, se ensambló un circuito en una protoboard siguiendo el esquema proporcionado en el enunciado. El divisor de tensión se construyó utilizando dos resistencias de 10kΩ y un capacitor de desacople tipo lenteja 220 nF para filtrar la componente continua. La alimentación de 3.3V se tomó directamente de la propia placa BluePill, asegurando así una fuente de energía estable y adecuada para el circuito.

La frecuencia de muestreo del ADC de la placa BluePill generalmente se especifica en las características técnicas del microcontrolador STM32 que utiliza esta placa. La BluePill suele utilizar microcontroladores de la serie STM32F1, como el STM32F103C8T6, entre otros modelos.

El STM32F103C8T6, por ejemplo, tiene un ADC con una frecuencia de muestreo máxima de 1 Msps (1 Mega-samples por segundo). Esto significa que puede muestrear señales analógicas hasta un máximo de 1 millón de veces por segundo. Sin embargo, la frecuencia de muestreo puede depender de la configuración específica del ADC según sea necesario.

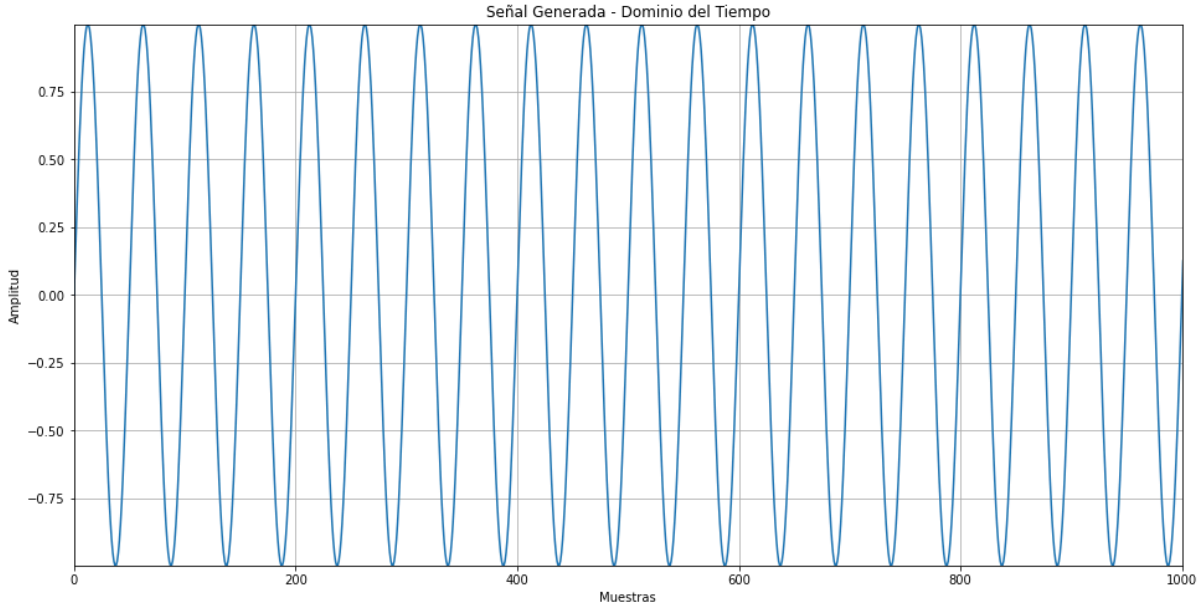
Según el teorema de muestreo de Nyquist, para evitar el aliasing (superposición de frecuencias) al muestrear una señal analógica, la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima de la señal analógica que se quiere digitalizar.

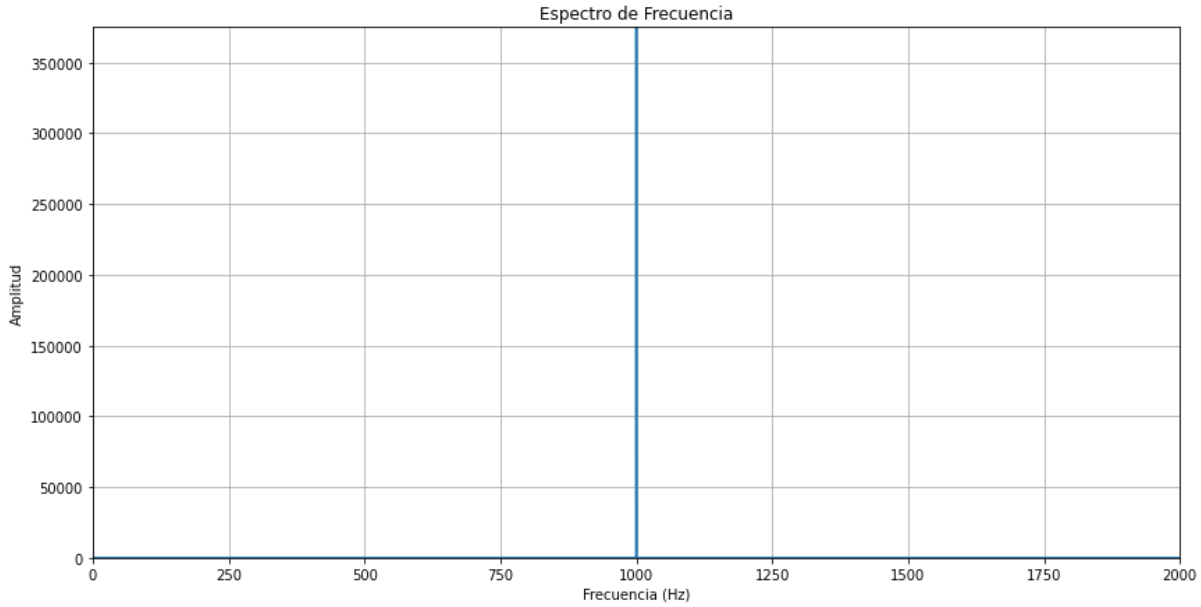
Ejercicio 1:

- Frecuencia de muestreo CAD - 50 kHz
- Frecuencia máxima de la señal a digitalizar fs/2 = 25 kHZ

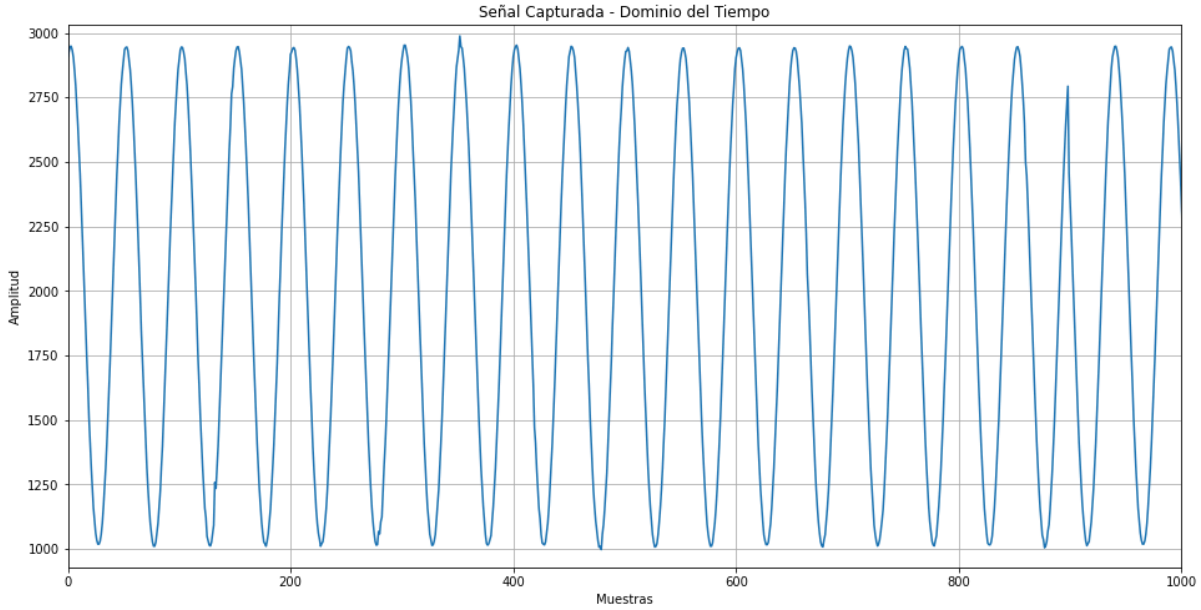
```
uint32_t configAdcContinuo(uint32_t fs);

HardwareTimer tim1(TIM1);

static volatile uint32_t fmuestreo = 50000;
static volatile uint32_t num_bits = 12;
```

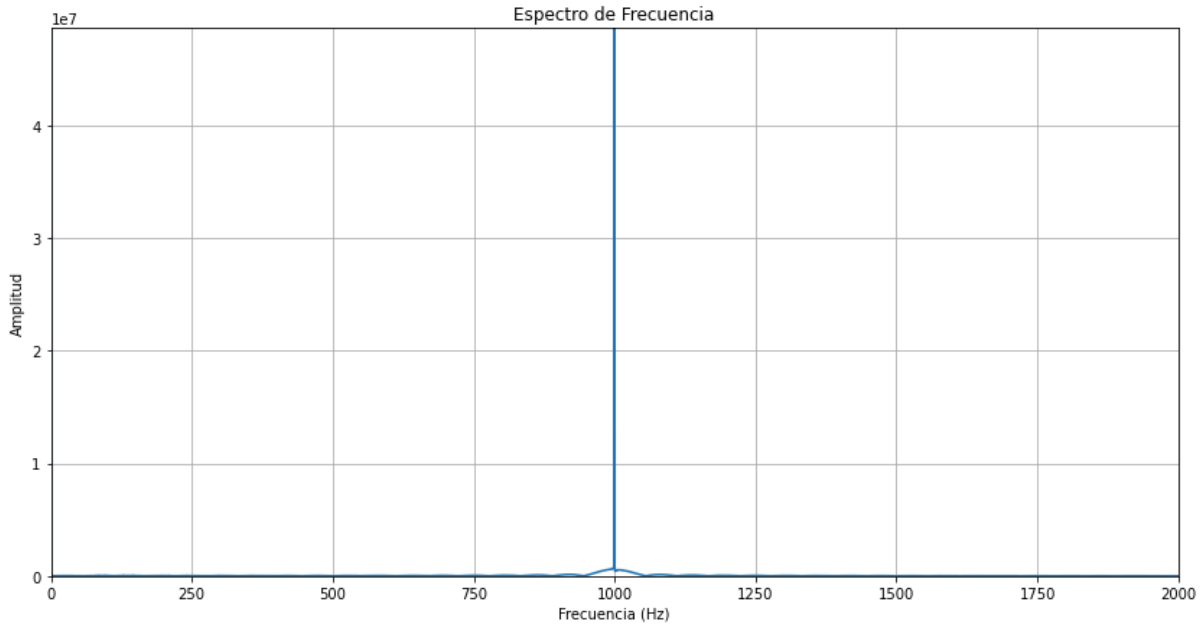
Ejercicio 2: Señal Senoidal de 1 kHz





Ejercicio 3: Señal capturada por la BluePill

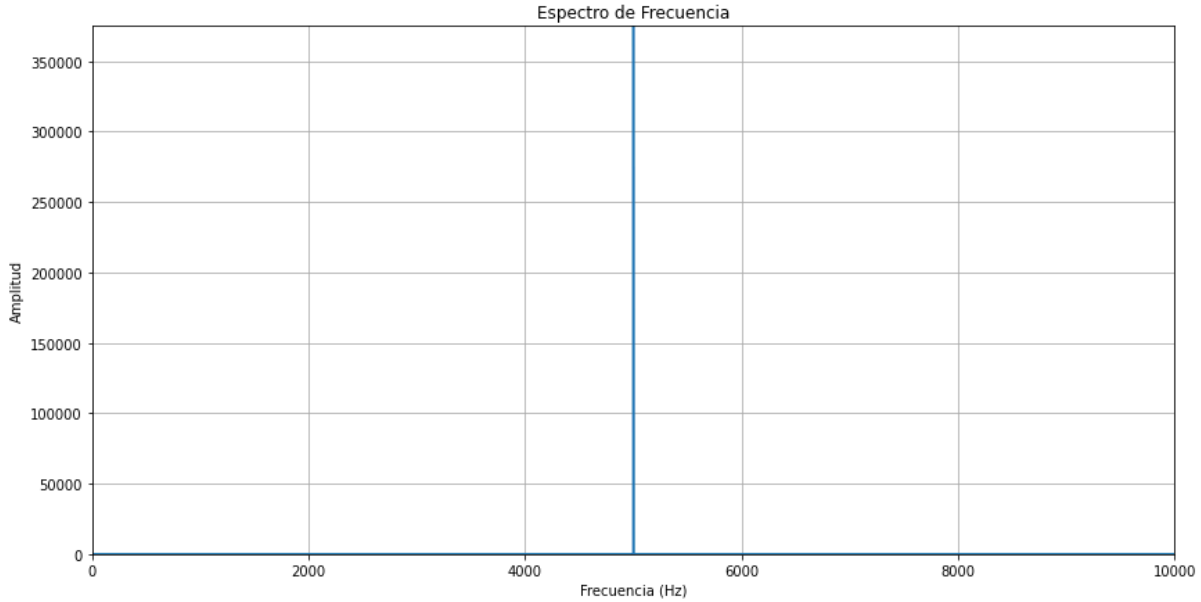


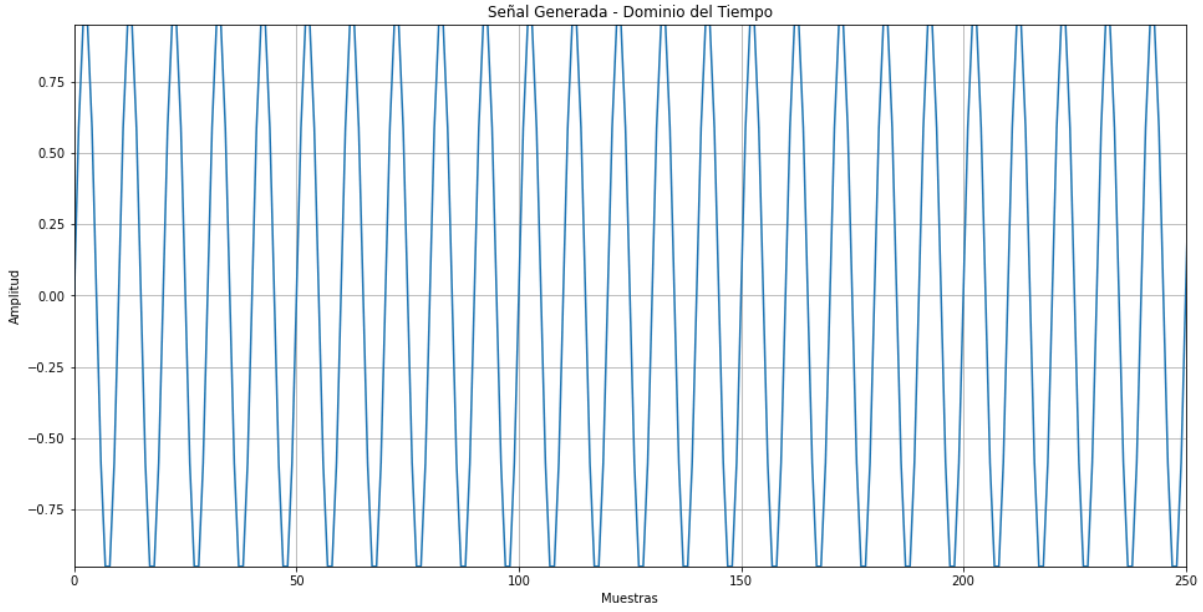


Ejercicio 4

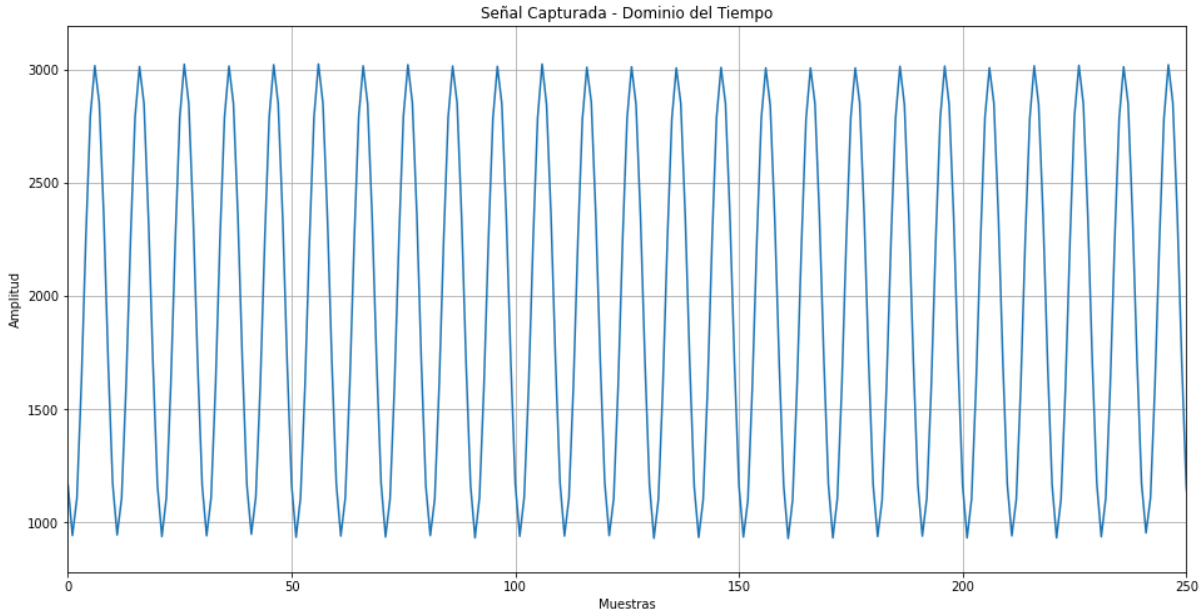
a) $fo = \frac{1}{10} fs$

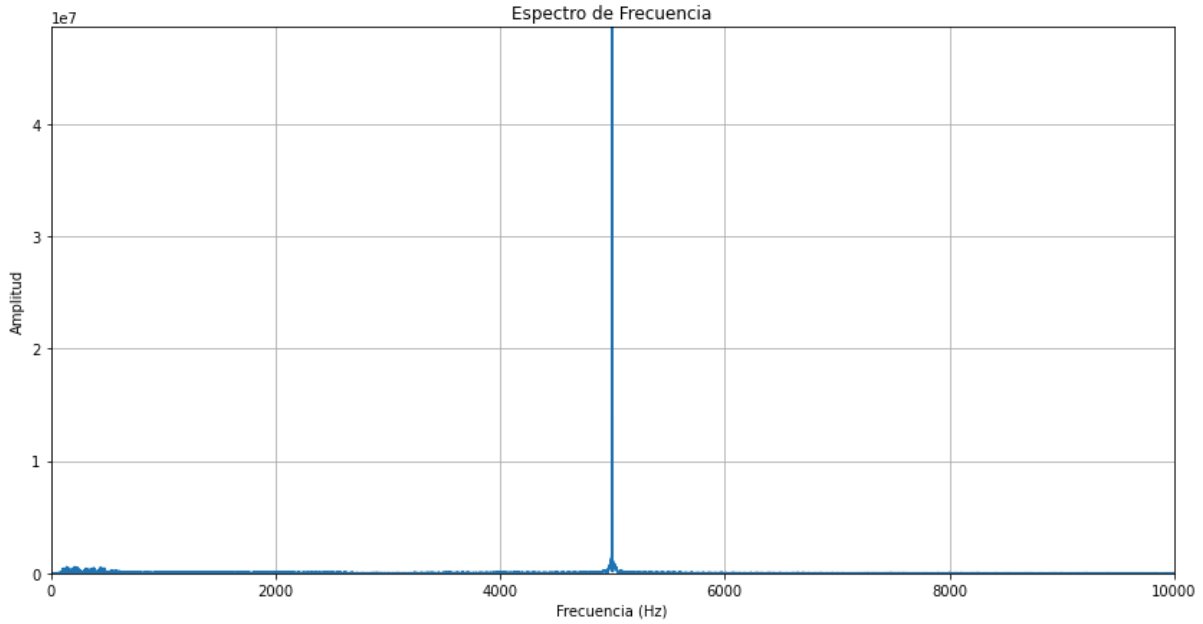
Señal Generada





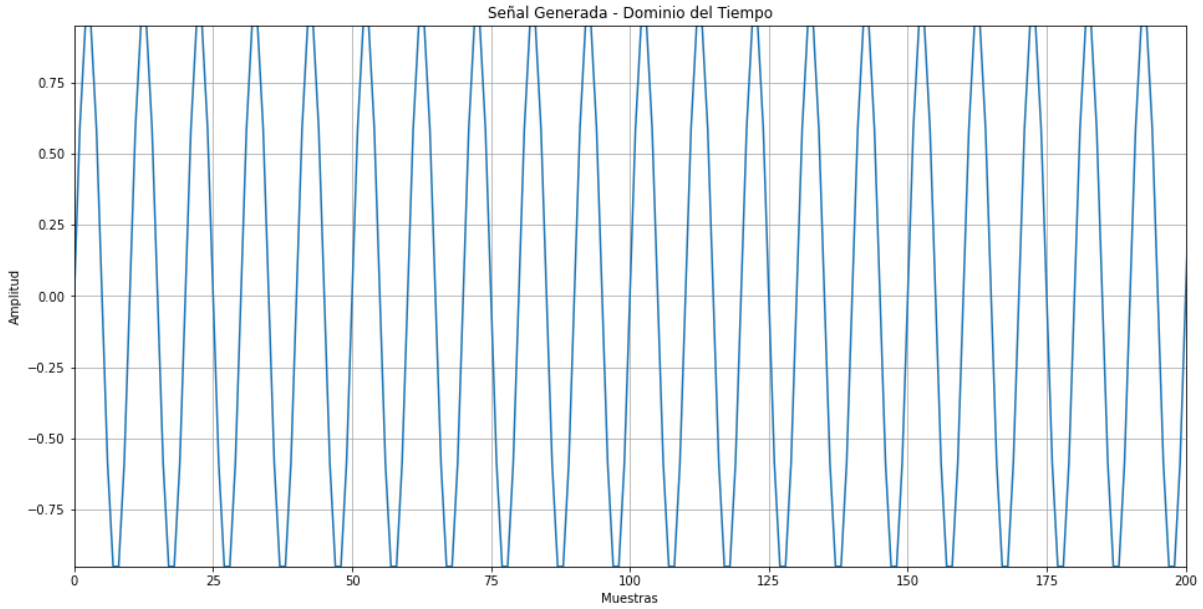
Señal Capturada

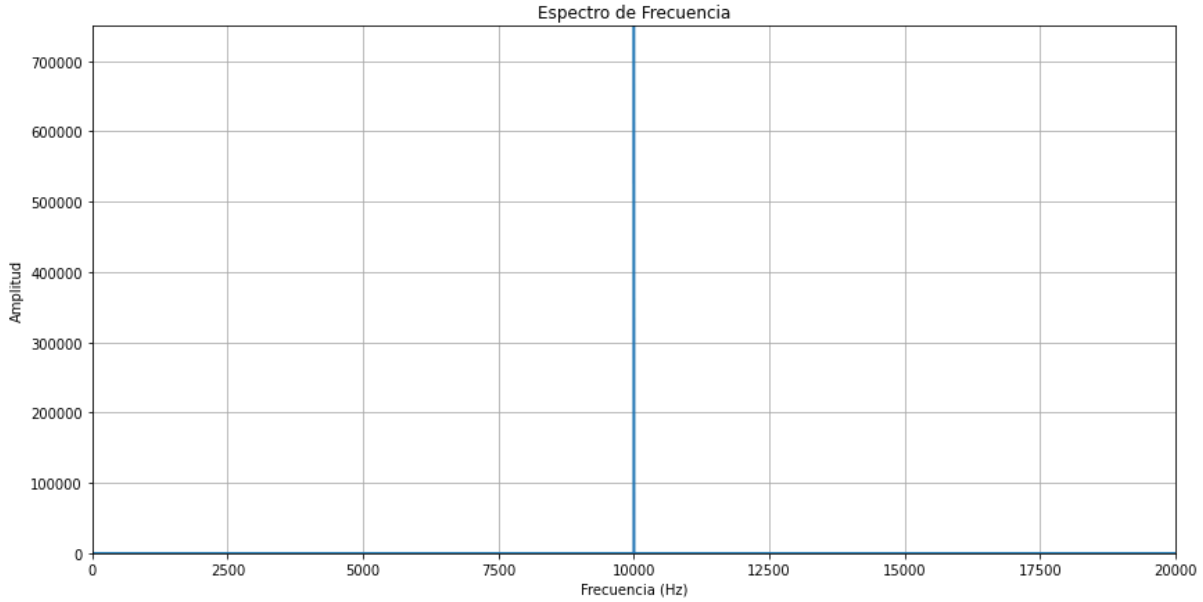




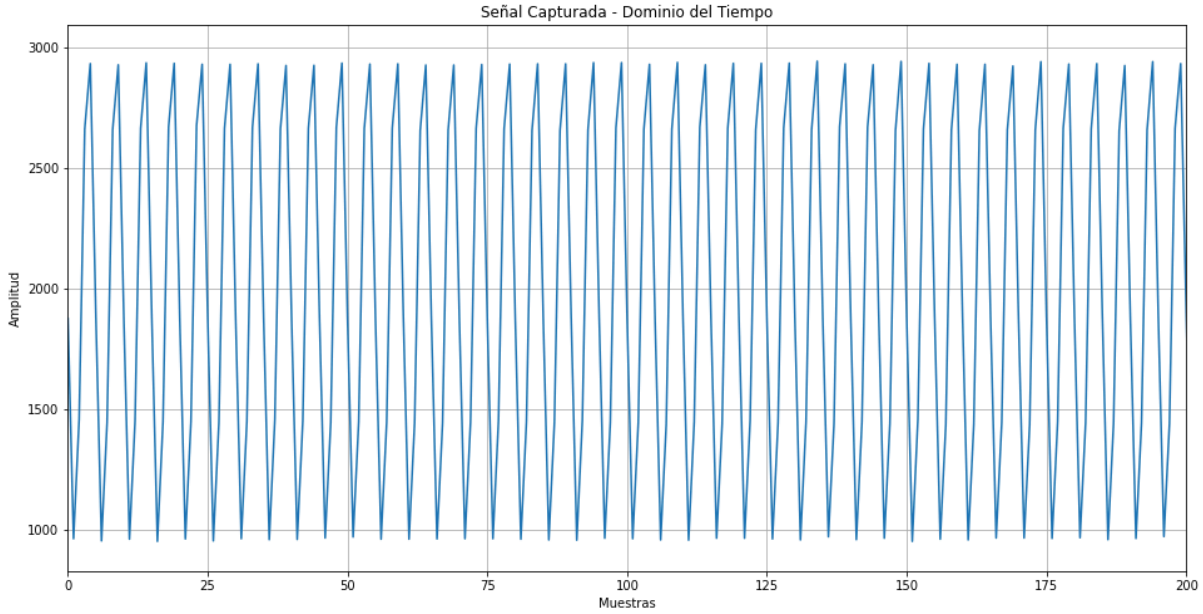
b) $fo = \frac{1}{5} fs$

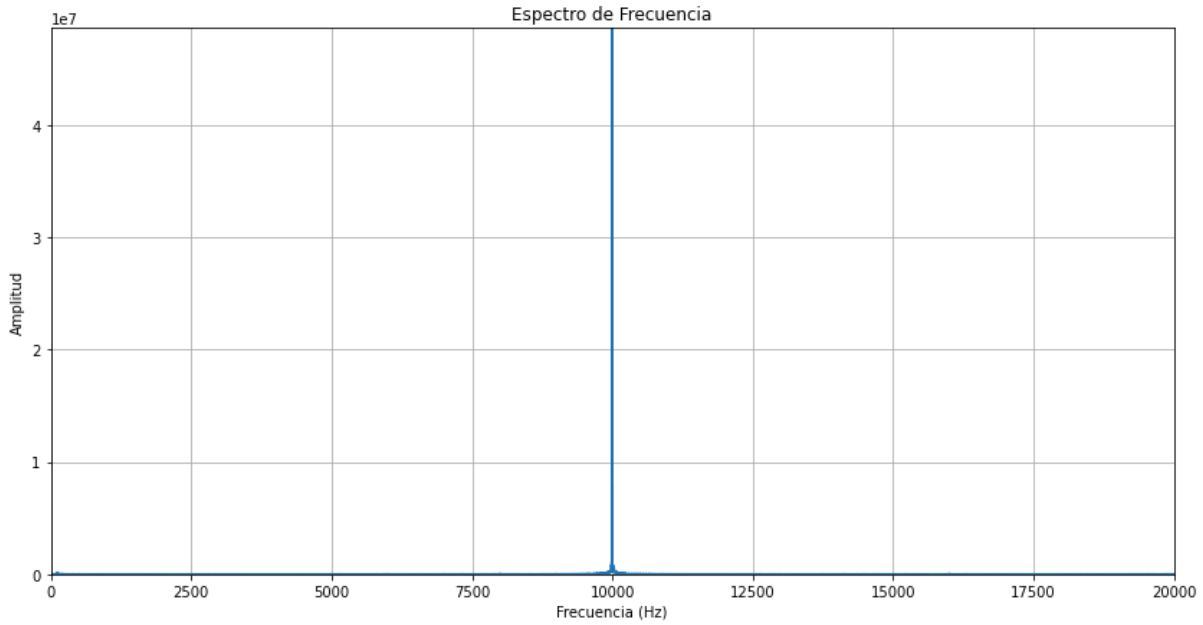
Señal Generada





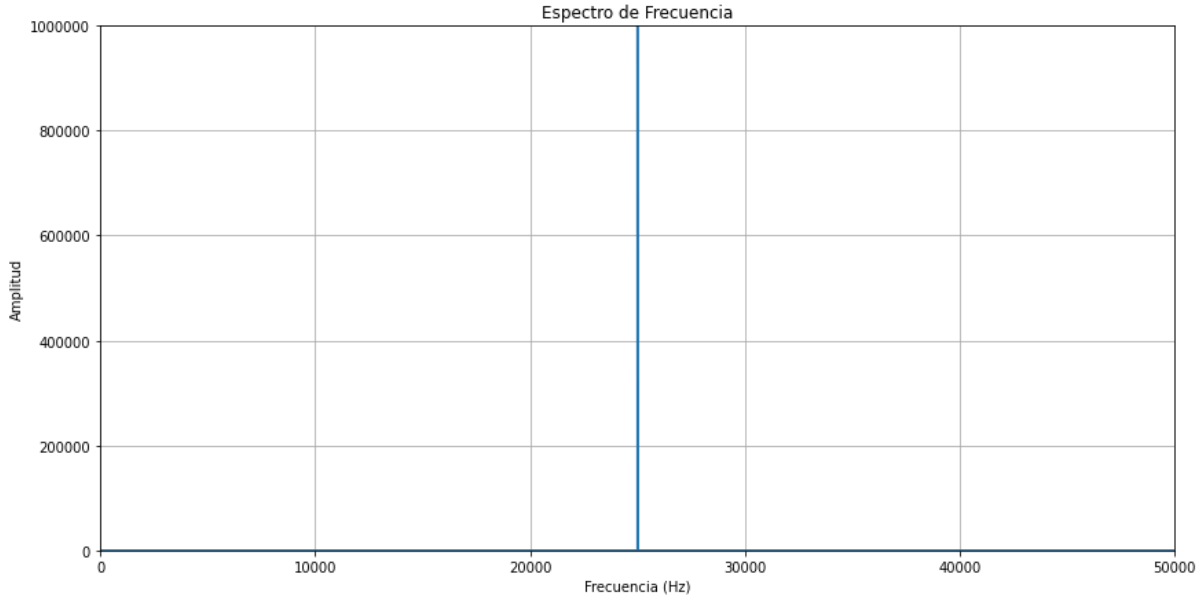
Señal Capturada

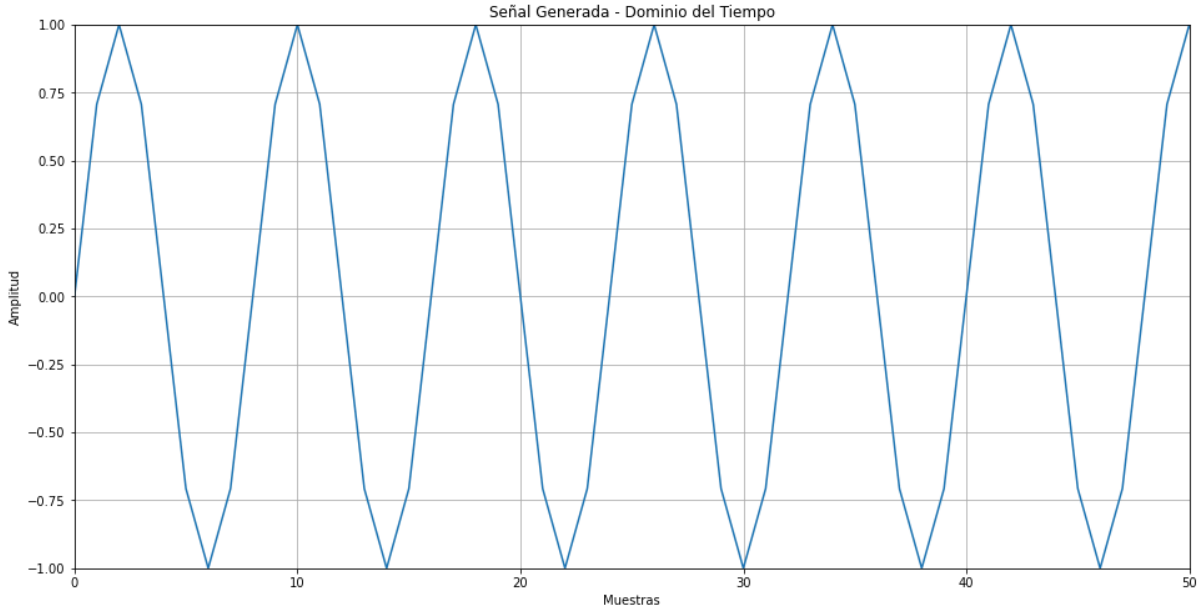




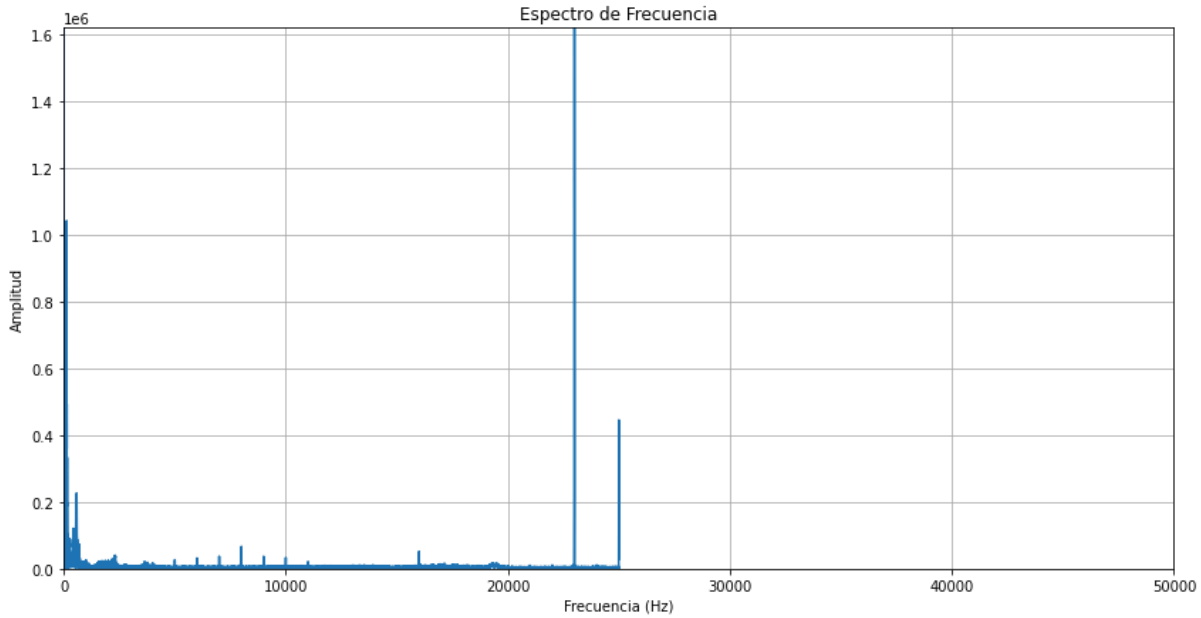
c) $fo = \frac{1}{2} fs$

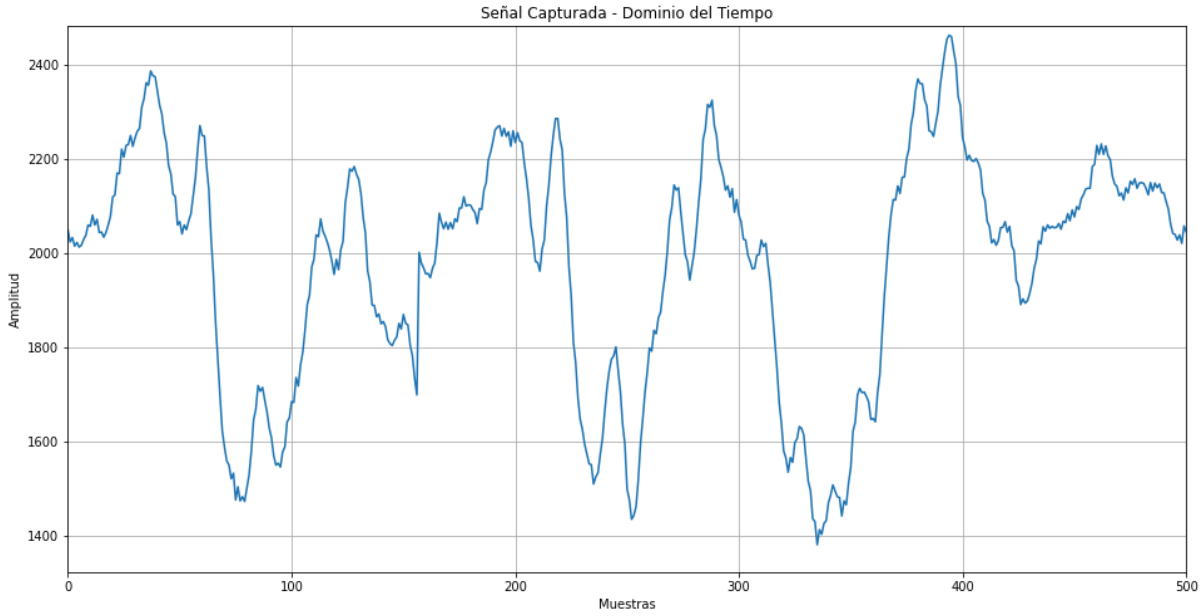
Señal Generada





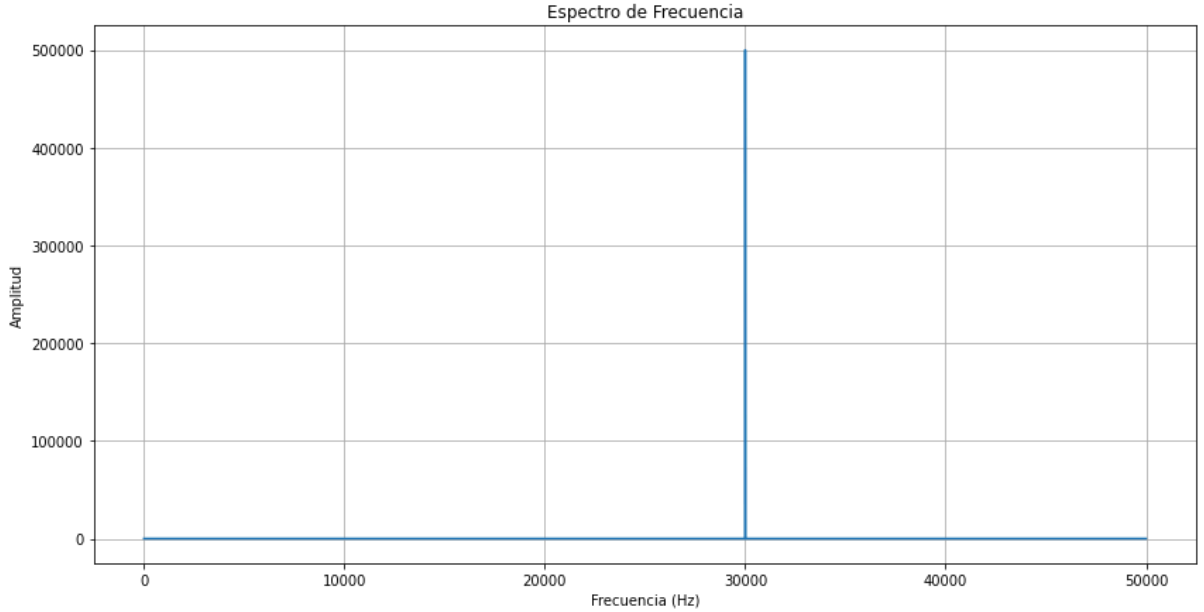
Señal Capturada

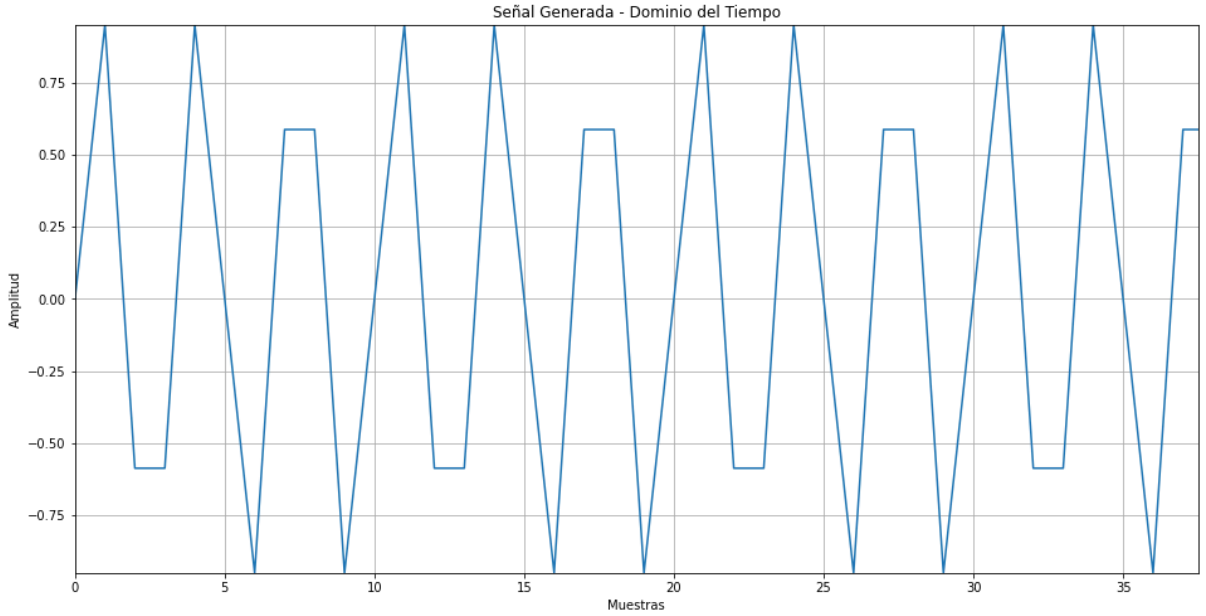




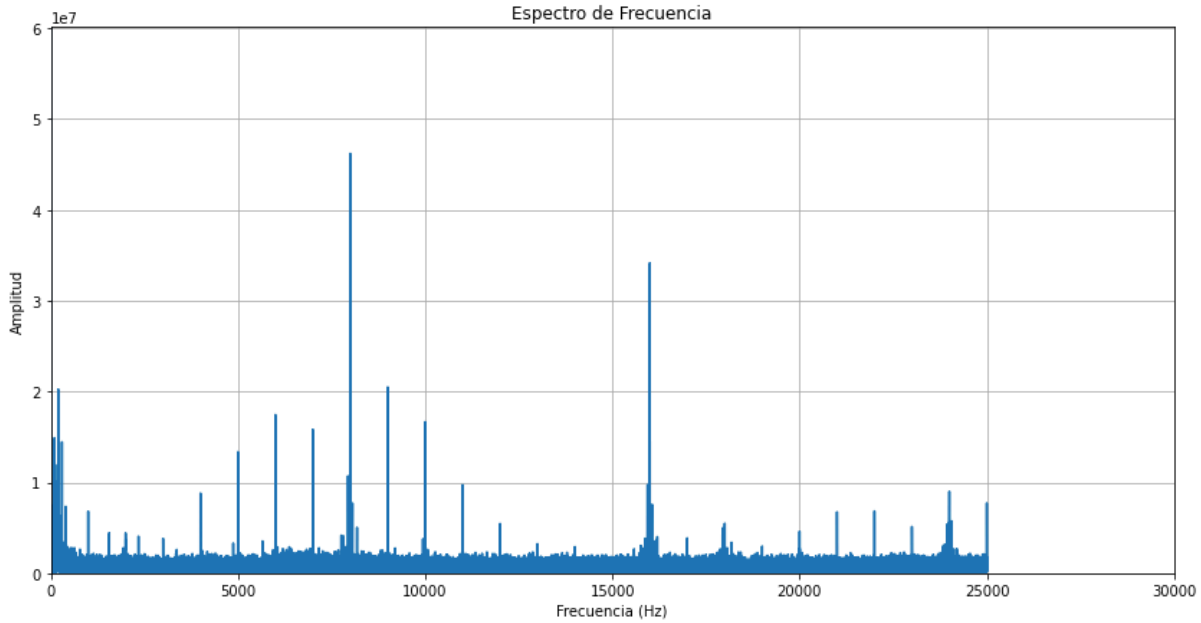
d) $fo = \frac{3}{5} fs$

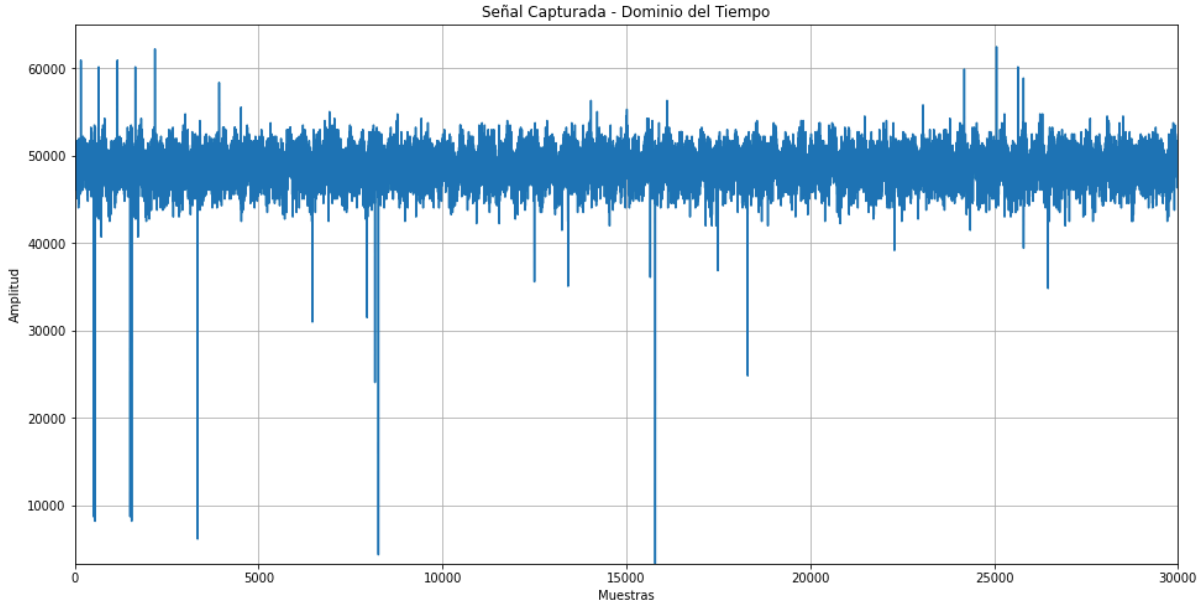
Señal Generada





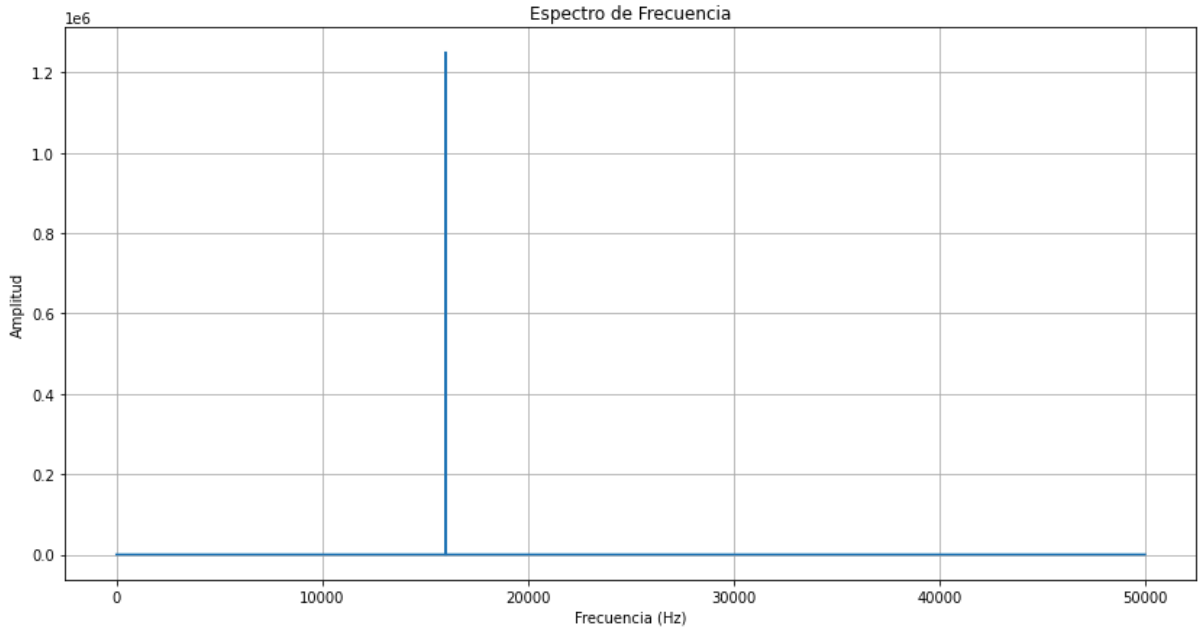
Señal Capturada

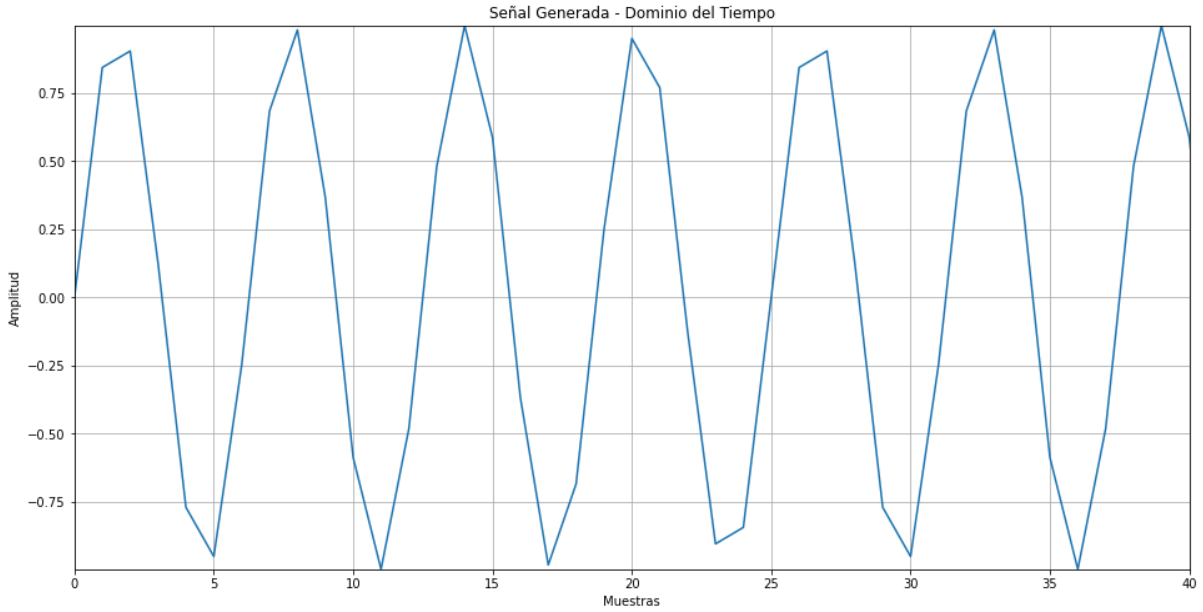




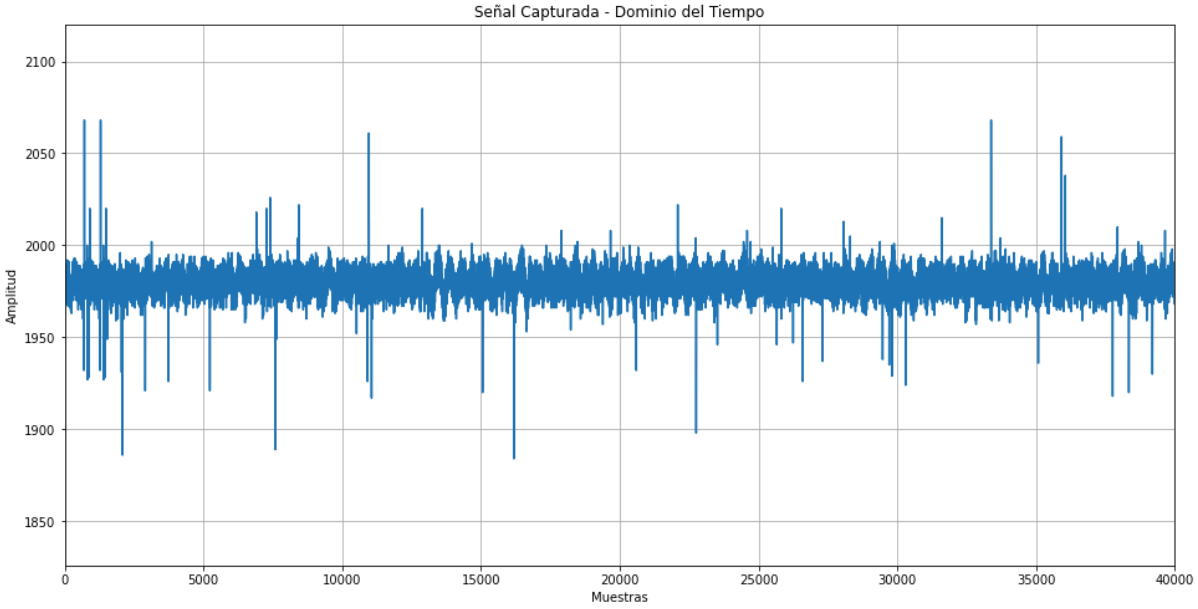
e) $fo = \frac{4}{5} fs$

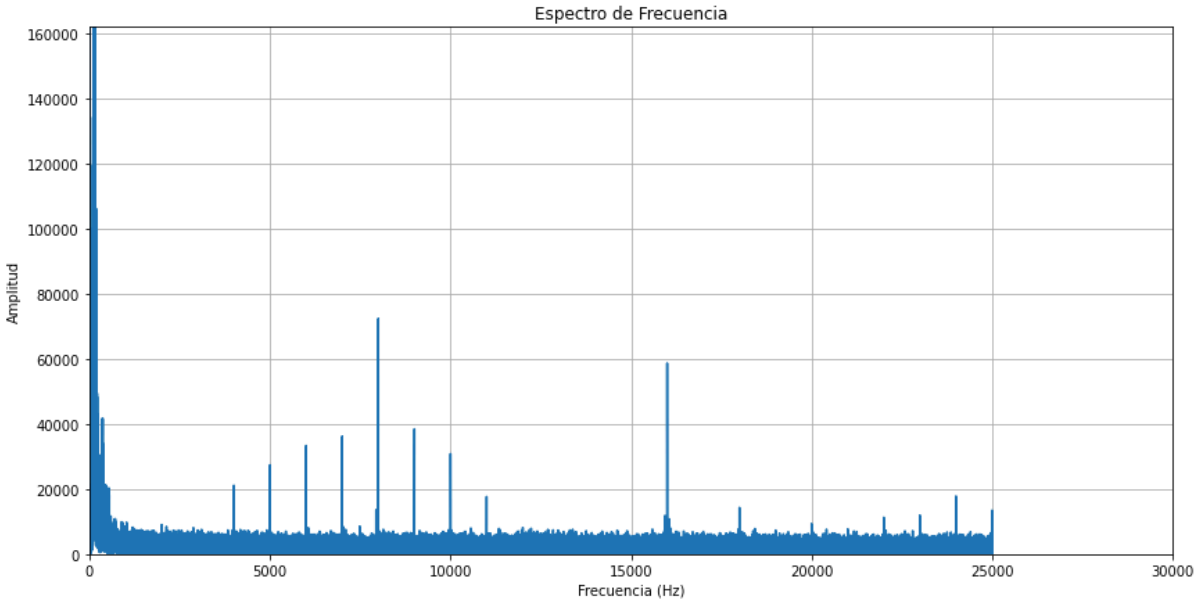
Señal Generada





Señal Capturada





Ejercicio 5: Señal cuadrada de 1 kHz con un Duty Cycle de 0.2

```python
# Parámetros de la señal
fs = 50000  # Frecuencia de muestreo (Hz)
duration = 10.0  # Duración de la señal (segundos)
frequency = 1000  # Frecuencia de la señal (Hz)
duty = 0.2  # Duty cycle
```

✓ 0.0s Python

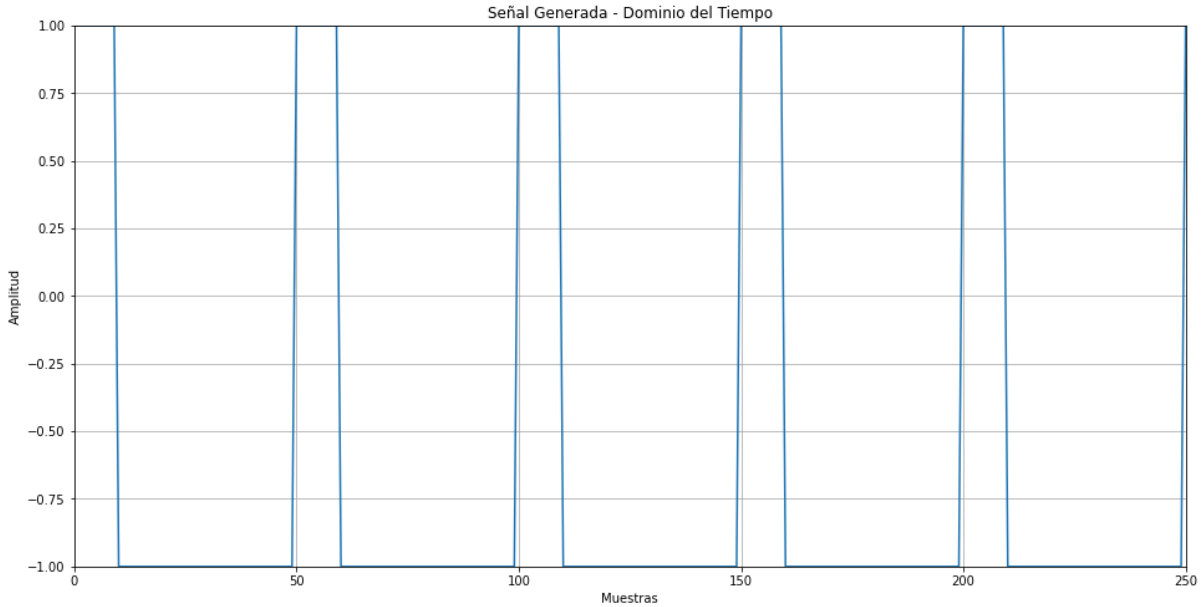
```python
# Generar la señal de onda senoidal
t = np.linspace(0, duration, int(fs * duration), endpoint=False)
signal = 1 * np.sin(2 * np.pi * frequency * t)
signal_2 = 1 * square(2 * np.pi * frequency * t, duty)
```

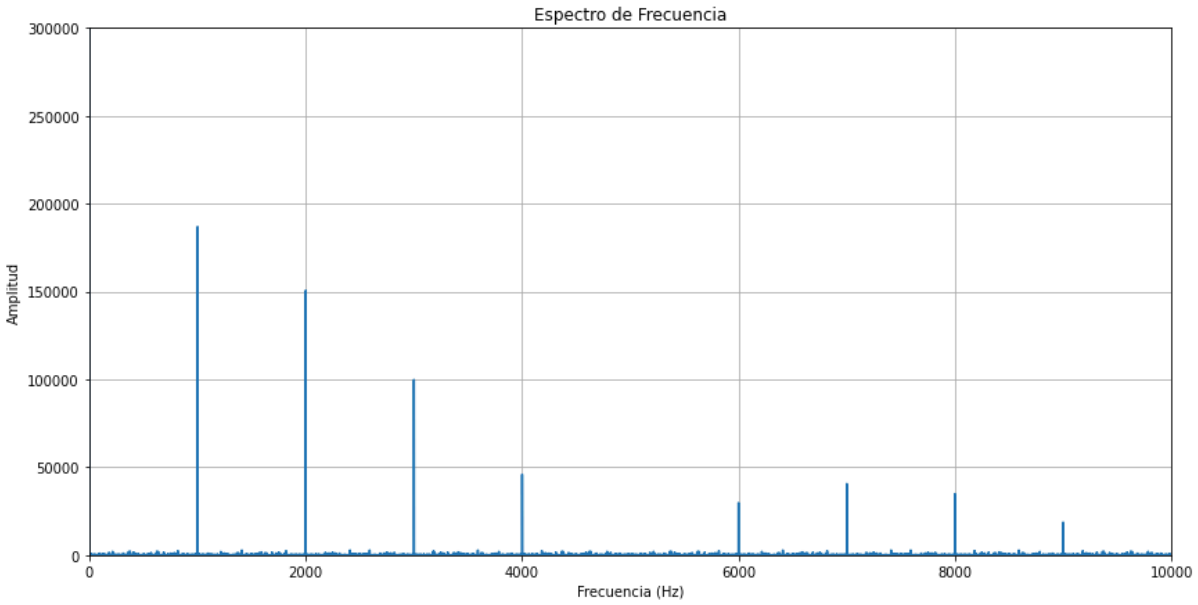
✓ 0.2s Python

```python
# Reproducir la señal
sd.play(signal_2, fs)
print("Reproduciendo señal")
sd.wait()
print("Finalizado")
```

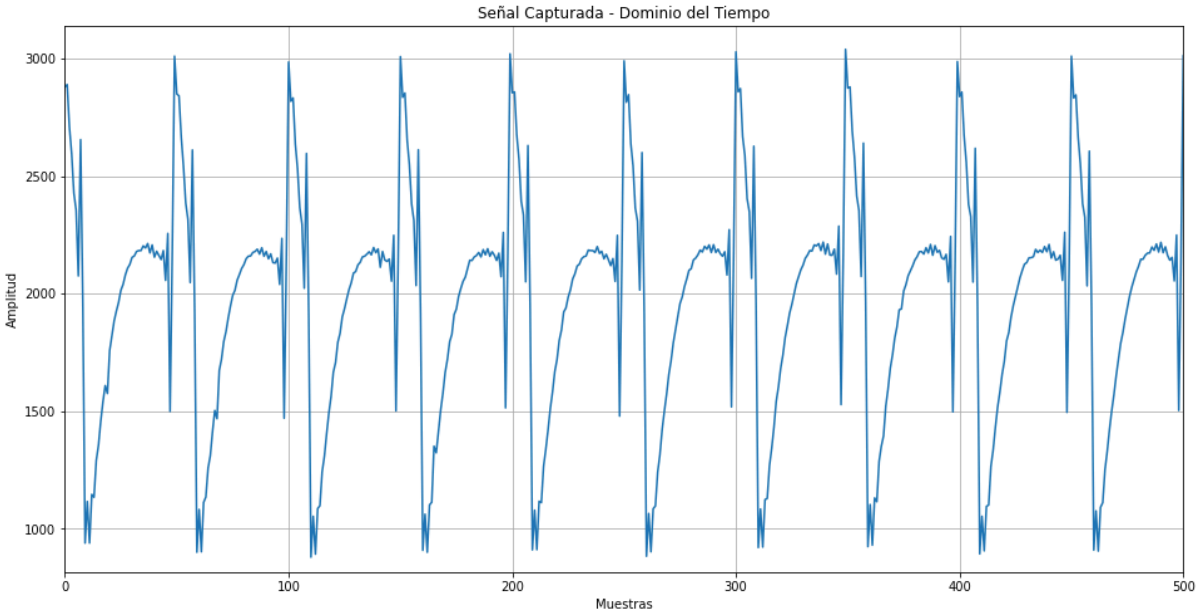
✓ 10.4s Python

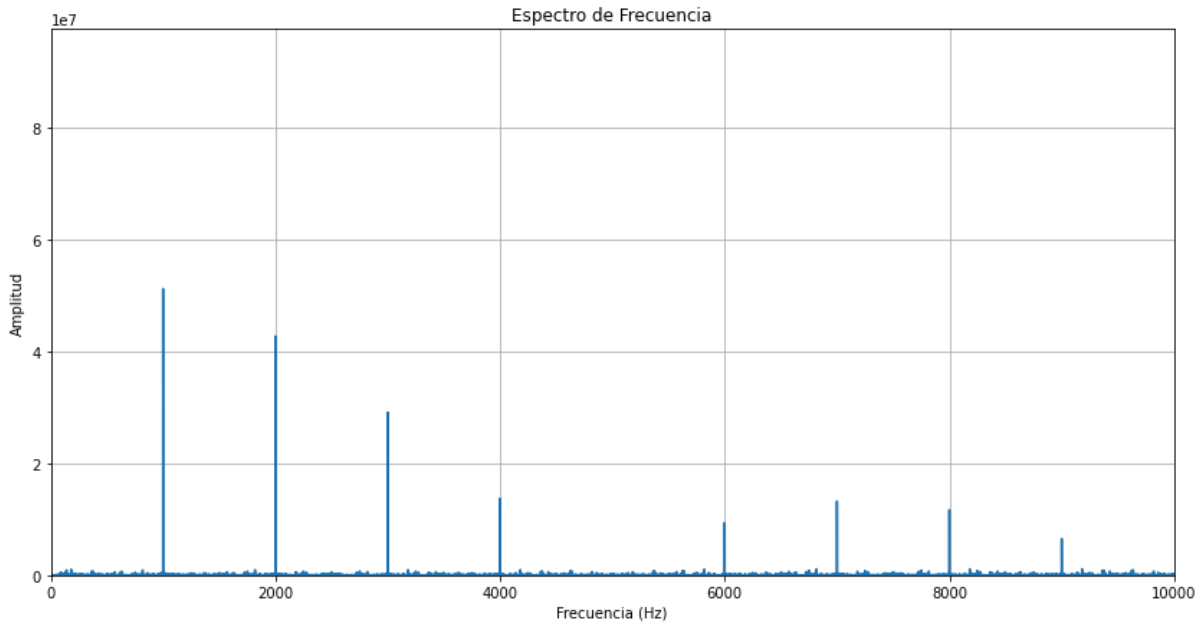
```
Reproduciendo señal
Finalizado
```



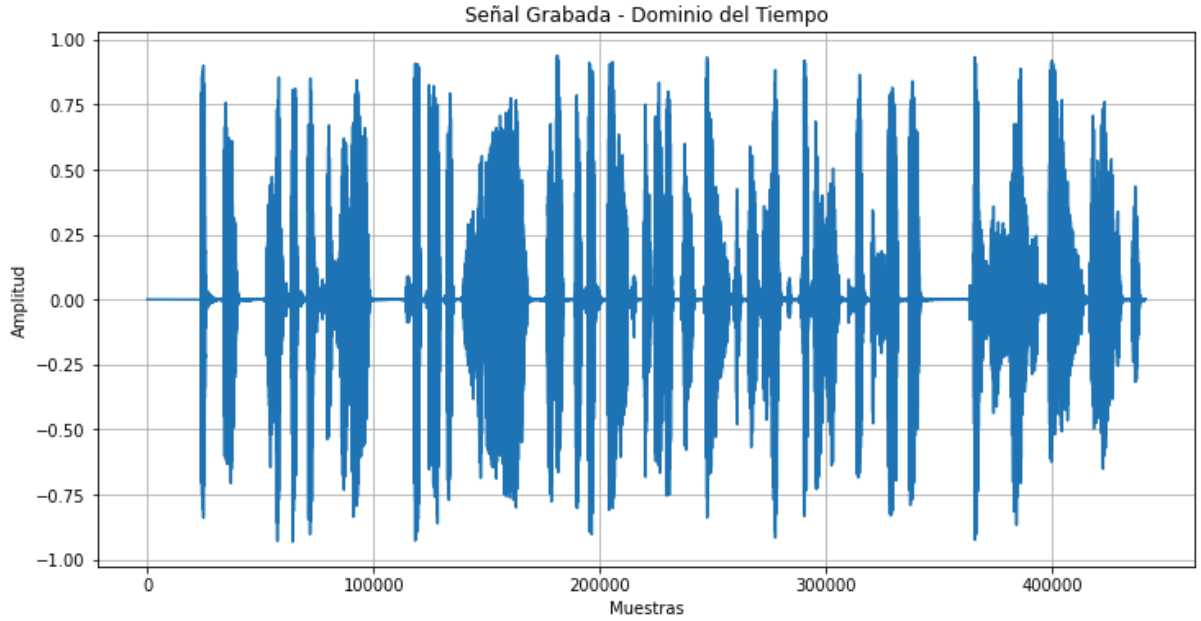


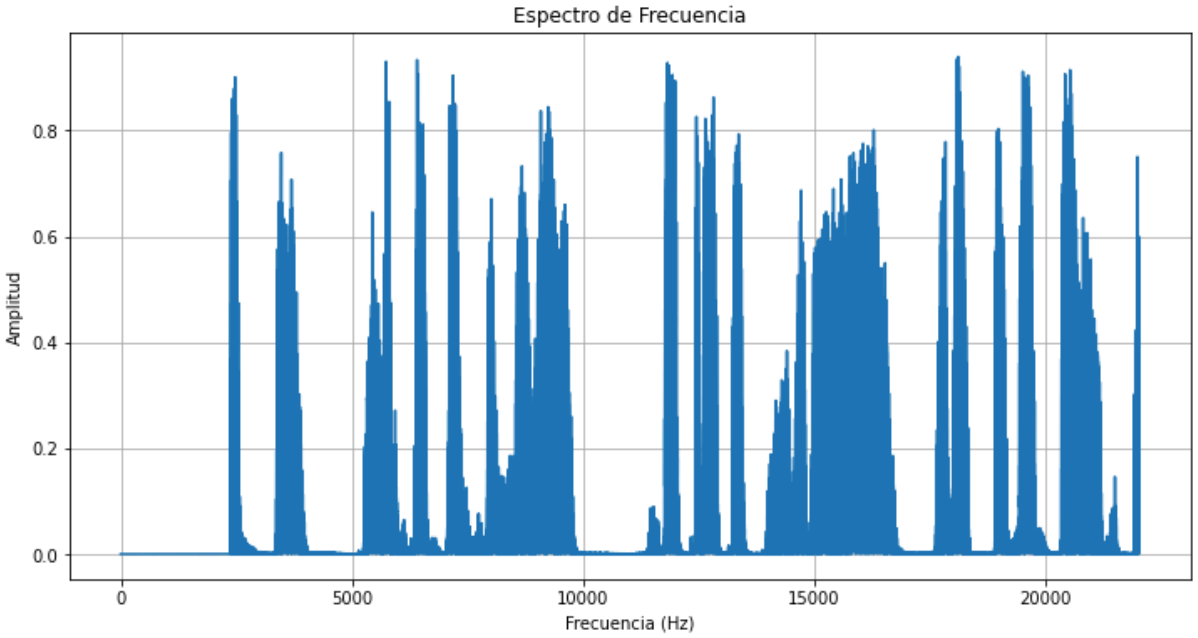
Señal Capturada



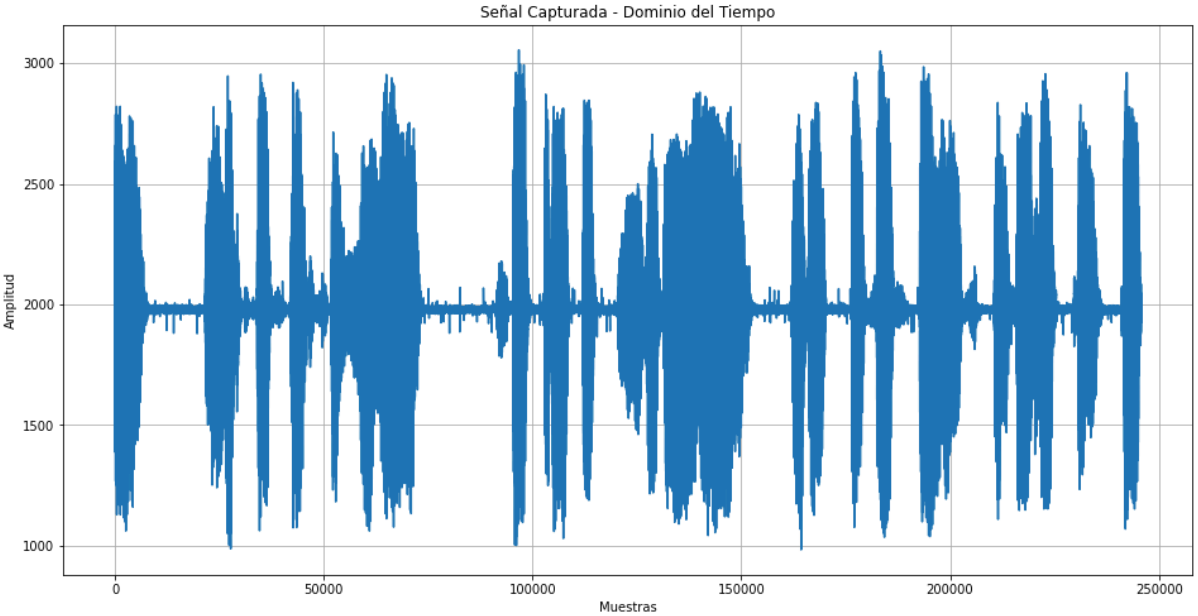


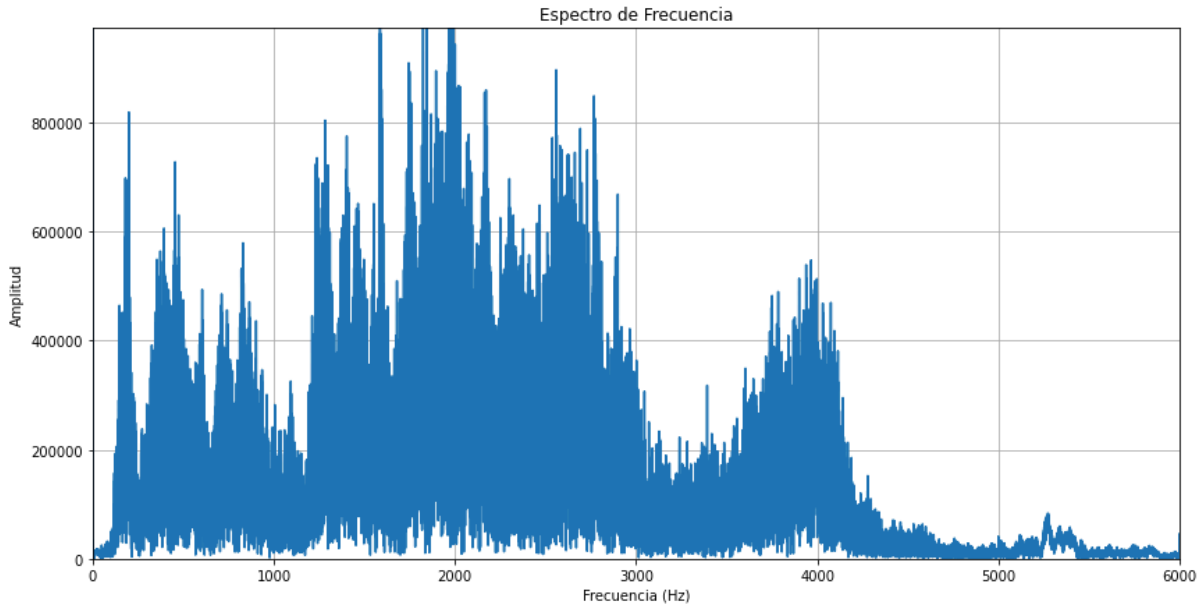
Ejercicio 5: Señal de audio (voz)





Señal Capturada





Conclusiones

En el desarrollo de este proyecto, se logró generar, capturar y analizar señales de audio utilizando una placa BluePill y Python. Sin embargo, se presentaron algunas limitaciones durante la captura de señales.

A medida que se aumentó la frecuencia de las señales generadas, la calidad de las capturas disminuyó significativamente. Esto se debe a las limitaciones en la frecuencia de muestreo del ADC de la BluePill, que no es lo suficientemente alta para captar con precisión señales de alta frecuencia. Como resultado, no se cumplió con el teorema de Nyquist, que establece que la frecuencia de muestreo debe ser al menos el doble de la frecuencia de la señal para evitar el aliasing. Además, los errores de cuantización y la ausencia de filtros adecuados afectaron la fidelidad de las señales capturadas.

Anexo Repositorio

Enlace de Github: **https://github.com/facundorosales96/TP_BLUEPILL_EAL**