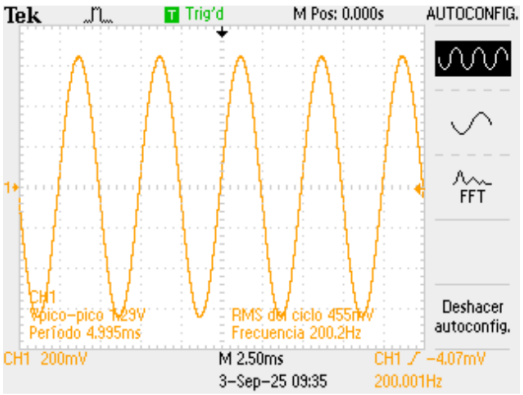
PARTE 1

2.Verificacion de la señal



4. Ejecución del programa

```python
import utime
import array
import math
import cmath

# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 1024
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")

    # Guardar muestras en archivo
    with open("muestras.txt", "w") as f:
        f.write("Tiempo(s)\tVoltaje(V)\n")
        voltajes = [(x / 65535) * 3.3 for x in muestras]
        for t, v in zip(tiempos, voltajes):
            f.write(f"{t:.6f}\t{v:.5f}\n")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.00 Hz
Offset DC removido: 1.595 V
Frecuencia dominante: 199.55 Hz
Amplitud señal: 0.155 V
Piso de ruido: 0.00135 V (-67.76 dB FS)
SNR: 41.22 dB, ENOB: 6.55 bits
```

64

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 64
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.20 Hz
Offset DC removido: 1.592 V
Frecuencia dominante: 206.96 Hz
Amplitud señal: 0.031 V
Piso de ruido: 0.00351 V (-59.46 dB FS)
SNR: 18.99 dB, ENOB: 2.86 bits
```

128

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 128
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.23 Hz
Offset DC removido: 1.592 V
Frecuencia dominante: 206.96 Hz
Amplitud señal: 0.098 V
Piso de ruido: 0.00729 V (-53.11 dB FS)
SNR: 22.58 dB, ENOB: 3.46 bits
```

FRECUENCIAS

100HZ

900HZ

1800HZ

64 100HZ

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 64
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.26 Hz
Offset DC removido: 1.595 V
Frecuencia dominante: 88.70 Hz
Amplitud señal: 0.027 V
Piso de ruido: 0.00370 V (-59.01 dB FS)
SNR: 17.39 dB, ENOB: 2.60 bits
```

128

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 128
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.04 Hz
Offset DC removido: 1.596 V
Frecuencia dominante: 103.47 Hz
Amplitud señal: 0.108 V
Piso de ruido: 0.00577 V (-55.15 dB FS)
SNR: 25.43 dB, ENOB: 3.93 bits
>>>
```

256

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 256
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_0
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1891.77 Hz
Offset DC removido: 1.590 V
Frecuencia dominante: 103.46 Hz
Amplitud señal: 0.266 V
Piso de ruido: 0.00872 V (-51.56 dB FS)
SNR: 29.69 dB, ENOB: 4.64 bits
>>>
```

512

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 512
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_0
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real:
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.10 Hz
Offset DC removido: 1.592 V
Frecuencia dominante: 99.78 Hz
Amplitud señal: 0.317 V
Piso de ruido: 0.00176 V (-65.48 dB FS)
SNR: 45.13 dB, ENOB: 7.20 bits

>>>
```

1024

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 1024
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.43 Hz
Offset DC removido: 1.593 V
Frecuencia dominante: 99.80 Hz
Amplitud señal: 0.159 V
Piso de ruido: 0.00119 V (-68.89 dB FS)
SNR: 42.54 dB, ENOB: 6.77 bits

>>>
```

2048

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 2048
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1891.66 Hz
Offset DC removido: 1.592 V
Frecuencia dominante: 99.76 Hz
Amplitud señal: 0.079 V
Piso de ruido: 0.00066 V (-73.98 dB FS)
SNR: 41.59 dB, ENOB: 6.62 bits

>>>
```

FRECUENCIA 900 HZ

64

```
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 64
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_0
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real:
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.13 Hz
Offset DC removido: 1.593 V
Frecuencia dominante: 916.50 Hz
Amplitud señal: 0.031 V
Piso de ruido: 0.00283 V (-61.33 dB FS)
SNR: 20.79 dB, ENOB: 3.16 bits

>>>
```

128

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 128
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real:
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.36 Hz
Offset DC removido: 1.593 V
Frecuencia dominante: 901.83 Hz
Amplitud señal: 0.122 V
Piso de ruido: 0.00543 V (-55.67 dB FS)
SNR: 27.05 dB, ENOB: 4.20 bits
```

256

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 256
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.12 Hz
Offset DC removido: 1.594 V
Frecuencia dominante: 901.71 Hz
Amplitud señal: 0.321 V
Piso de ruido: 0.00822 V (-52.07 dB FS)
SNR: 31.83 dB, ENOB: 4.99 bits
>>>
```

512

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 512
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz"
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.32 Hz
Offset DC removido: 1.593 V
Frecuencia dominante: 901.81 Hz
Amplitud señal: 0.299 V
Piso de ruido: 0.00427 V (-57.77 dB FS)
SNR: 36.91 dB, ENOB: 5.84 bits
>>>
```

1024

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 1024
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.03 Hz
Offset DC removido: 1.594 V
Frecuencia dominante: 901.67 Hz
Amplitud señal: 0.152 V
Piso de ruido: 0.00238 V (-62.85 dB FS)
SNR: 36.13 dB, ENOB: 5.71 bits
>>>
```

2048

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 2048
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1891.86 Hz
Offset DC removido: 1.594 V
Frecuencia dominante: 901.59 Hz
Amplitud señal: 0.076 V
Piso de ruido: 0.00126 V (-68.39 dB FS)
SNR: 35.64 dB, ENOB: 5.63 bits
```

FRECUENCIA DE 1800HZ

64

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 64
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
    fs_real = N / elapsed_time
    print(f"Frecuencia deseada: {f_muestreo} Hz, frecuencia real: {fs_real:.2f} Hz")
```

Consola

```
MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1892.21 Hz
Offset DC removido: 1.604 V
Frecuencia dominante: 88.70 Hz
Amplitud señal: 0.032 V
Piso de ruido: 0.00354 V (-59.39 dB FS)
SNR: 19.17 dB, ENOB: 2.89 bits
>>>
```

128

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 128
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)
```

Consola

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Iniciando adquisición y análisis...
Frecuencia deseada: 2000 Hz, frecuencia real: 1891.86 Hz
Offset DC removido: 1.600 V
Frecuencia dominante: 88.68 Hz
Amplitud señal: 0.116 V
Piso de ruido: 0.00562 V (-55.38 dB FS)
SNR: 26.28 dB, ENOB: 4.07 bits
>>>
```

256

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 256
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
```

Consola

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
 Iniciando adquisición y análisis...
 Frecuencia deseada: 2000 Hz, frecuencia real: 1892.40 Hz
 Offset DC removido: 1.586 V
 Frecuencia dominante: 88.71 Hz
 Amplitud señal: 0.283 V
 Piso de ruido: 0.00874 V (-51.54 dB FS)
 SNR: 30.19 dB, ENOB: 4.72 bits

>>>
```

512

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 512
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
```

Consola

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
 Iniciando adquisición y análisis...
 Frecuencia deseada: 2000 Hz, frecuencia real: 1891.97 Hz
 Offset DC removido: 1.597 V
 Frecuencia dominante: 88.69 Hz
 Amplitud señal: 0.268 V
 Piso de ruido: 0.00673 V (-53.81 dB FS)
 SNR: 32.01 dB, ENOB: 5.02 bits
```

1024

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 1024
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
```

Consola

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
 Iniciando adquisición y análisis...
 Frecuencia deseada: 2000 Hz, frecuencia real: 1892.07 Hz
 Offset DC removido: 1.587 V
 Frecuencia dominante: 88.69 Hz
 Amplitud señal: 0.136 V
 Piso de ruido: 0.00351 V (-59.47 dB FS)
 SNR: 31.76 dB, ENOB: 4.98 bits
```

2048

```python
# Configuración inicial
adc = ADC(Pin(26))
N = 512
N_FFT = 2048
f_muestreo = 2000  # Frecuencia objetivo de muestreo (Hz)
dt_us = int(1_000_000 / f_muestreo)
data = array.array('H', [0] * N)

# Adquisición de datos con tiempos uniformes
def acquire_data():
    tiempos = []
    muestras = []
    start = utime.ticks_us()

    for i in range(N):
        t_actual = utime.ticks_diff(utime.ticks_us(), start) / 1_000_000
        tiempos.append(t_actual)
        muestras.append(adc.read_u16())
        utime.sleep_us(dt_us)

    elapsed_time = tiempos[-1]
```

Consola

```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
  Iniciando adquisición y análisis...
  Frecuencia deseada: 2000 Hz, frecuencia real: 1892.04 Hz
  Offset DC removido: 1.581 V
  Frecuencia dominante: 88.69 Hz
  Amplitud señal: 0.065 V
  Piso de ruido: 0.00179 V (-65.33 dB FS)
  SNR: 31.22 dB, ENOB: 4.89 bits
```