PRACTICA 1-DSP

JOSE ANTONIO GARCIA GRACIA, GABRIEL DE JESUS GARCÍA TINOCO, RODOLFO REYES GUTIERRÉZ.

Universidad de la Salle bajío

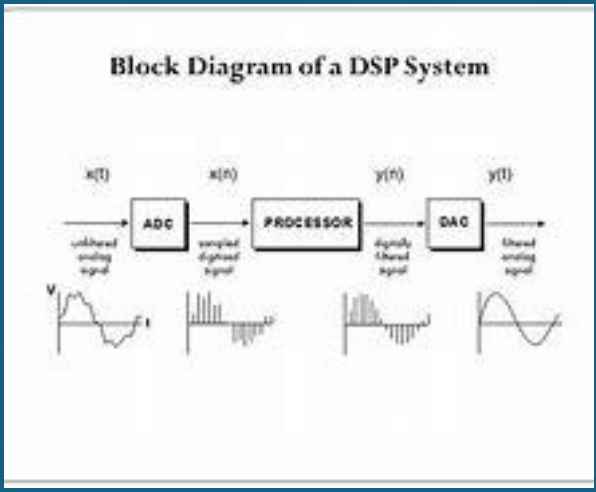
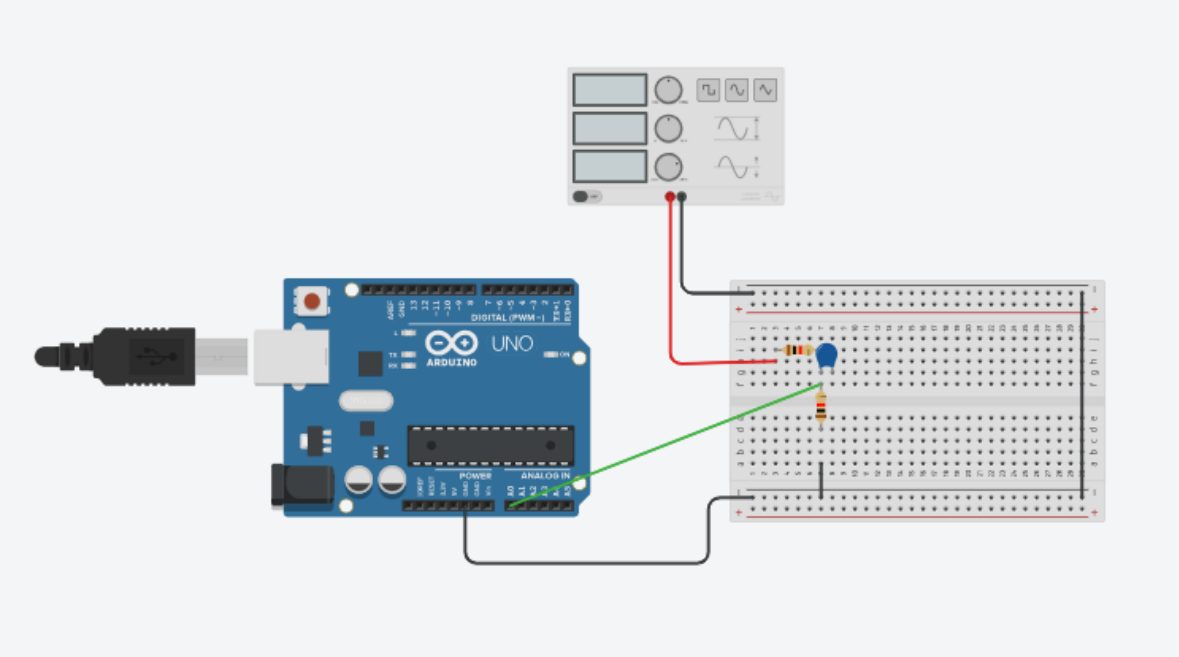
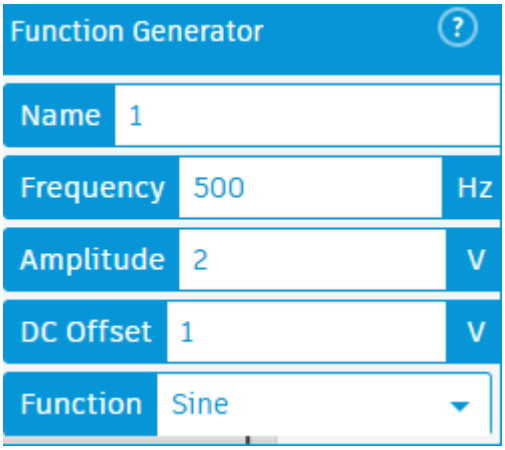


Diagrama esquemático



(Imagen 1.1)



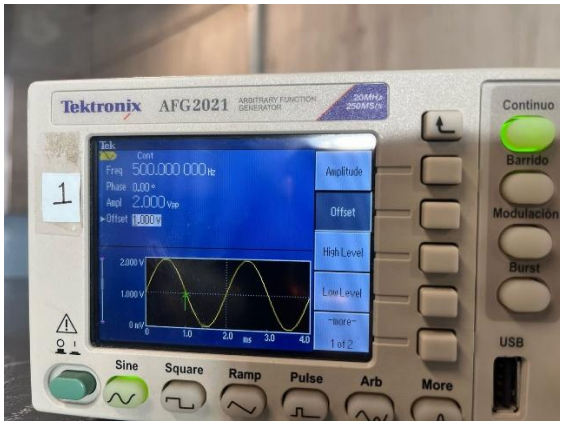
(Imagen 1.2)

En la imagen 1.1 se ve la conexión general del circuito, consta de tres partes:

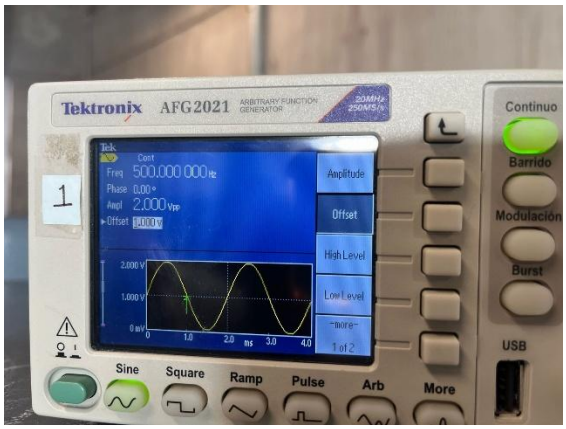
- Divisor de voltaje entre las dos resistencias.
- Capacitor de carga y descarga para la señal (mini filtro).
- Generador de funciones y conexiones.

Desarrollo de la práctica:

Se configuro el generador de funciones para que entregara una onda senoidal de 500 Hz, 2 Vpp y un offset de 1 para que la onda se haga positiva. La configuración y la onda senoidal en el osciloscopio se ve de la siguiente manera:



(Imagen 1.3)



(Imagen 1.4)

Después de esto se le conecta el Arduino a la computadora con su respectivo A0 a mitad del divisor de voltaje y el capacitor (Imagen 1.1) y se le sube este código para poder trabajar:

```
const int analogPin = A0;     // Pin analogico de entrada
const unsigned long samplingPeriod = 200; // Microsegundos entre muestras

void setup() {
  Serial.begin(115200);  // Iniciar comunicación serial

  analogReference(DEFAULT);

  delay(1000);   //Estabilización
}

void loop() {
  // Leer valor analógico
  static unsigned long lastSample = 0;

  unsigned long currentTime = micros();


  if (currentTime - lastSample >= SAMPLE_PERIOD) {
      // Promedio de múltiples lecturas
      int sensorValue = 0;
      for (int i = 0; i < 4; i++) {
           sensorValue += analogRead(analogPin);
           delayMicroseconds(10);
      }
      sensorValue = sensorValue / 4;
      Serial.println(sensorValue);
      lastSample = currentTime;
    }
}
```

(Imagen 1.5)

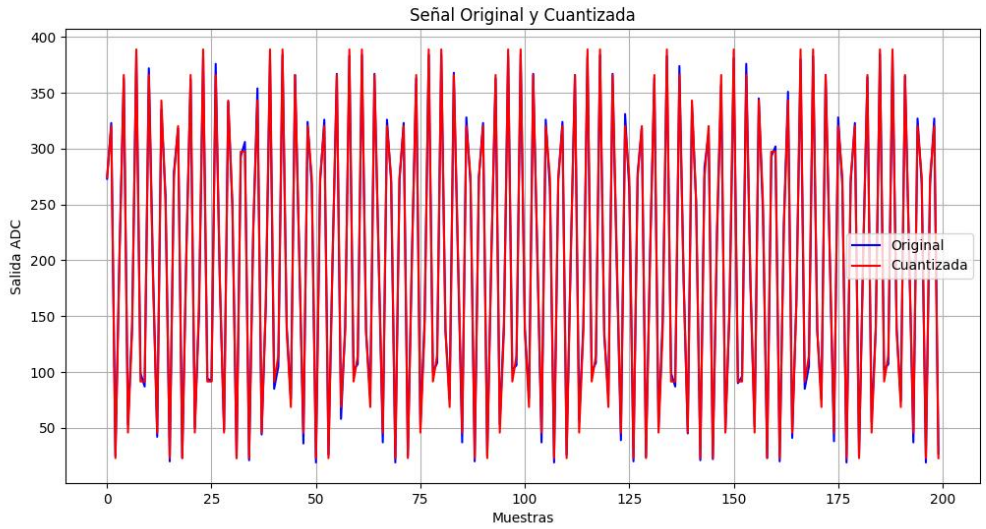
Al estar generando las muestras lo que sigue es poder visualizarlas con Python a lo cual nosotros implementamos este código:

```
import serial
import numpy as np
import matplotlib.pyplot as plt
from time import sleep


def adquirir_datos(puerto='COM3', muestras=1000):


# Configurar puerto serial
   ser = serial.Serial(puerto, 9600)
   sleep(2)  # Esperar inicialización
# Leer datos
   datos = []
   for i in range(muestras):
      if ser.in_waiting:
         linea = ser.readline()
         valor = int(linea.decode().strip())
         datos.append(valor)
   ser.close()
   return np.array(datos)



def cuantizar(datos, bits):
   niveles = 2**bits
   max_val = np.max(datos)
   min_val = np.min(datos)
   paso = (max_val - min_val) / niveles
   return np.round(datos/paso)*paso
```

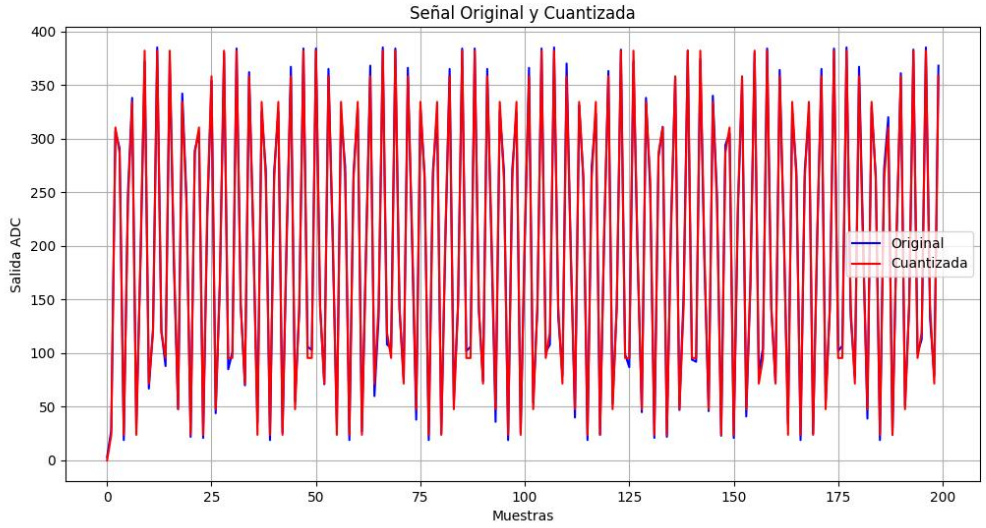
```
# Parámetros de adquisición
PUERTO = 'COM3'  # Cambiar según su sistema
MUESTRAS = 1000
BITS_CUANTIZACION = 3

# Adquirir datos
datos = adquirir_datos(PUERTO, MUESTRAS)
# Cuantizar datos
datos_cuantizados = cuantizar(datos, BITS_CUANTIZACION)
# Graficar
plt.figure(figsize=(12, 6))
plt.plot(datos, 'b-', label='Original')
plt.plot(datos_cuantizados, 'r-', label='Cuantizada')
plt.grid(True)
plt.legend()
plt.show()
```

Finalmente teníamos que modificar los parámetros que se nos solicitaron, los cuales fueron el SAMPLE_PERIOD en Arduino y modificar de 2 bits (4 niveles), 3 bits (8 niveles) y4 bits (16 niveles).

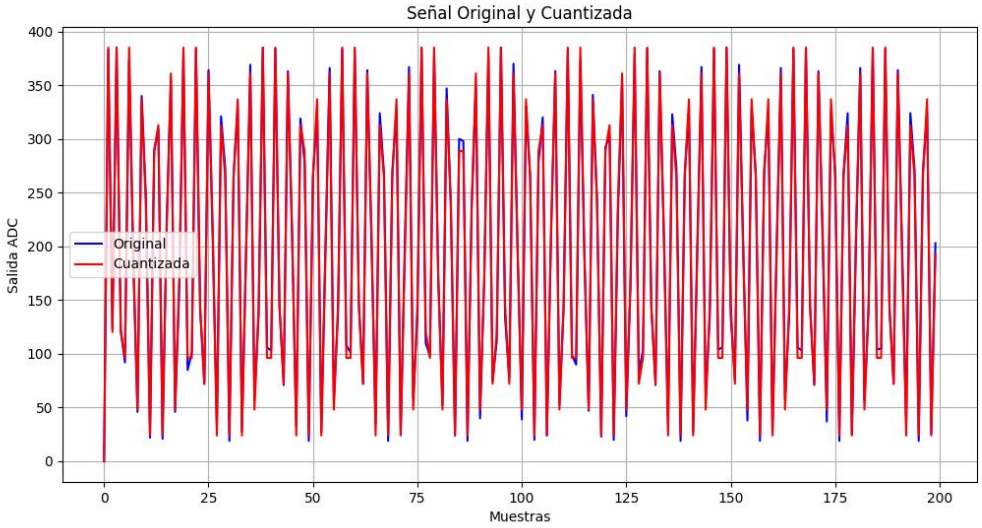
Las muestras de estas modificaciones quedan de la siguiente manera:



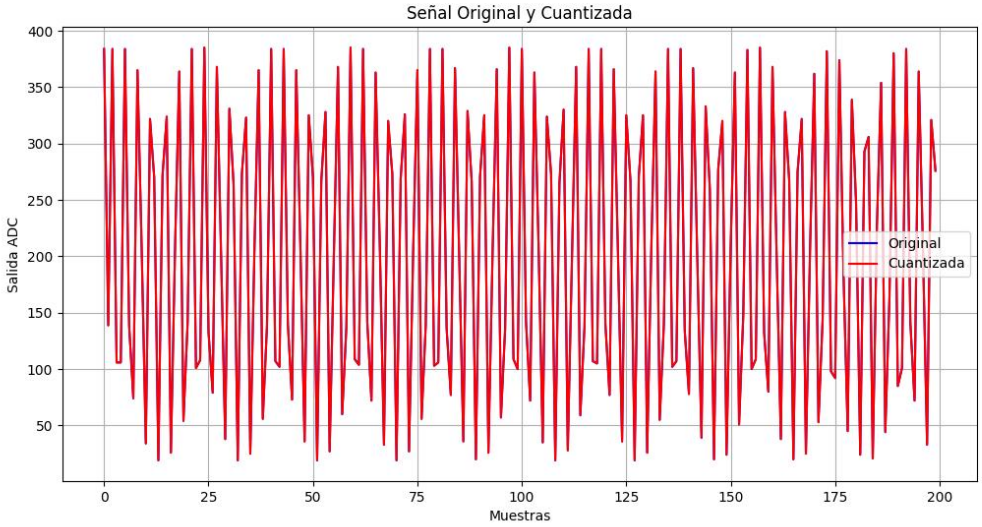
(Imagen 1.6) 2 bits (4 niveles) 100 microsegundos



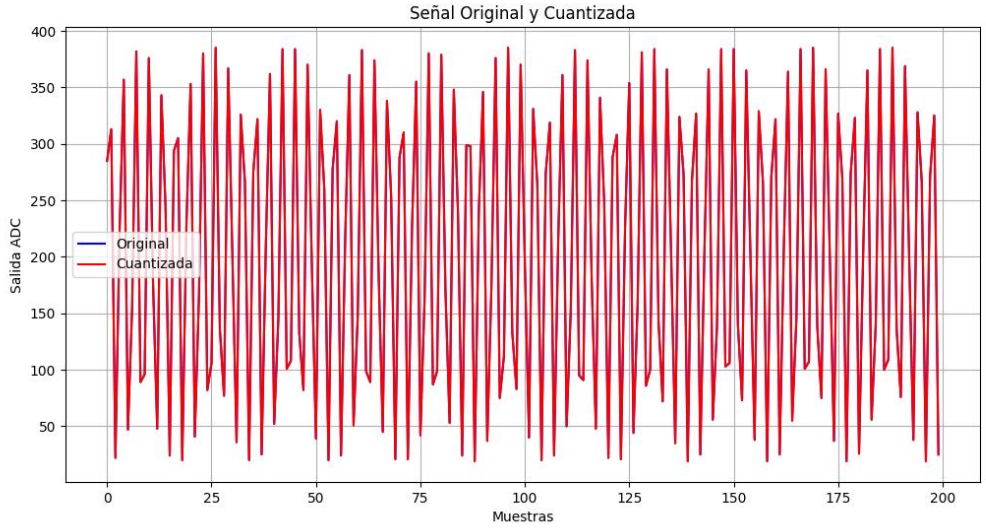
(Imagen 1.7) 2 bits (4 niveles) 500 microsegundos



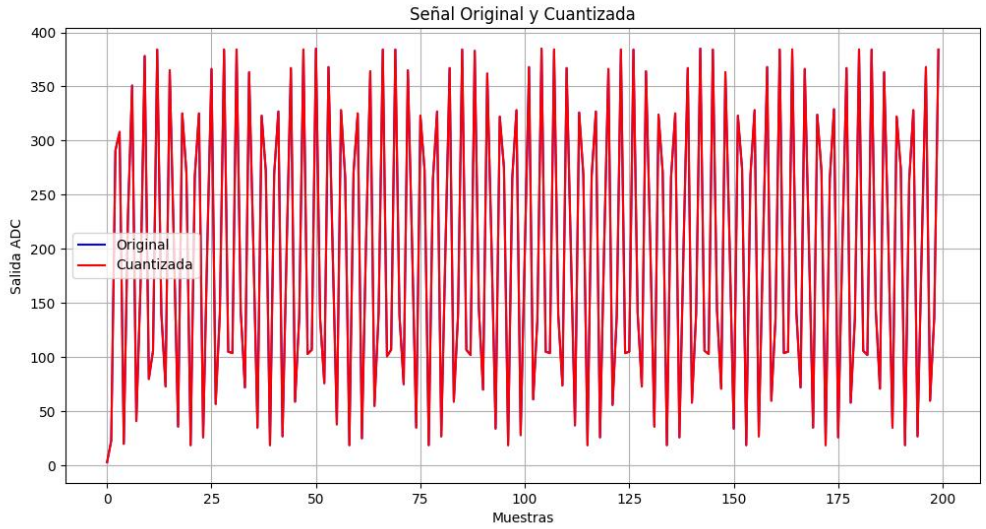
(Imagen 1.8) 2 bits (4 niveles) 1000 microsegundos.



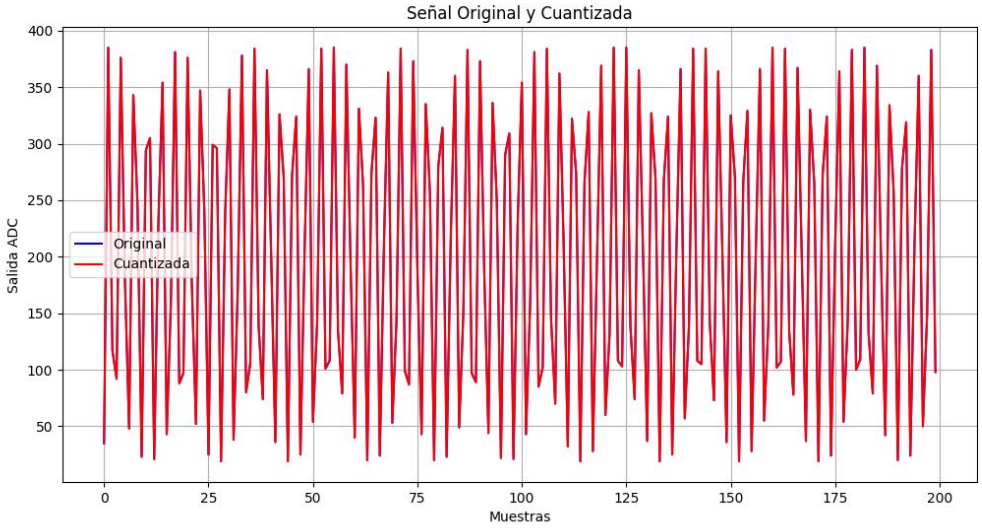
(Imagen 1.9) 3 bits (8 niveles) 100 microsegundos



(Imagen 2.0) 3 bits (8 niveles) 500 microsegundos.



(Imagen 2.1) 3 bits (8 niveles) 1000 microsegundos



(Imagen 2.2) 4 bits (16 niveles) 100 microsegundos.

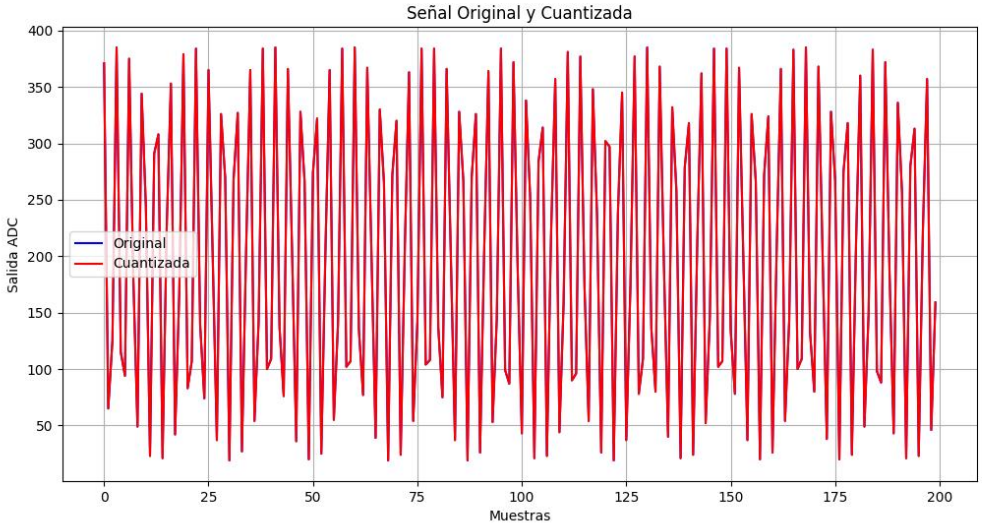
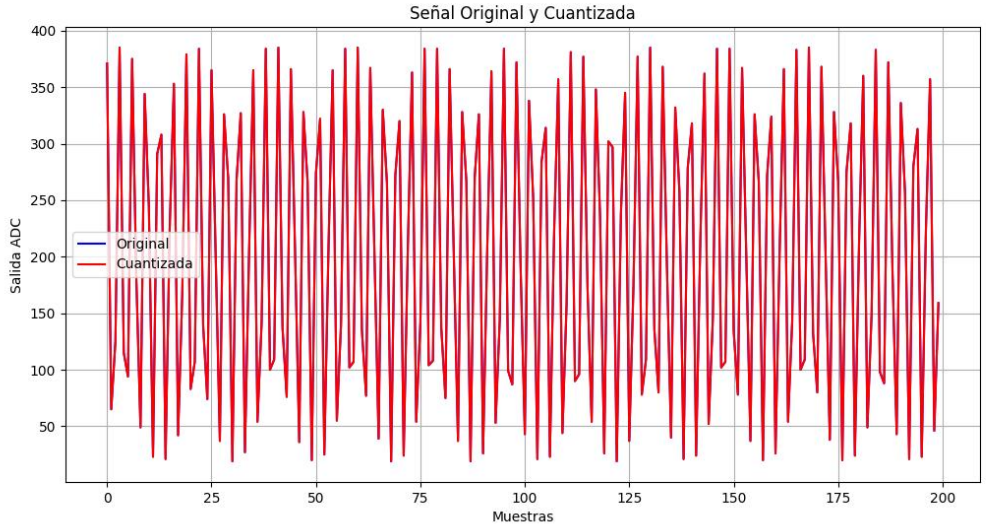


Imagen (2.3) 4 bits (16 niveles) 500 microsegundos.



(Imagen 2.4) 4 bits (16 niveles) 1000 microsegundos.

¿Qué sucede con la señal cuando el periodo de muestreo es muy grande?

- Ocurre un fenómeno llamado aliasing esto ocurre cuando la frecuencia de muestreo es menor al doble de la frecuencia más alta de la señal original.

¿Cómo afecta el número de bits a la calidad de la señal?

- A la resolución y precisión de la señal analógica cuando la pasamos a digital.

¿Por qué es importante el divisor de voltaje en el circuito?

- Proteger al microcontrolador
- Garantizar que no pase los 5 voltios que el Arduino tolera en sus entradas analógicas.
- Estabilidad de la señal.

¿Qué representa cada punto en la gráfica de Python?

- Muestras tomadas en el tiempo.
- Valor del ADC (Y)
- La señal roja es el proceso de cuantización, donde los valores continuos de la señal original se ajustan a niveles discretos, simulando cómo un ADC interpreta la señal.