```python
import wave
import struct

class ProcesadorWav:
    def __init__(self, archivo_entrada, archivo_salida, freq_original, freq_objetivo):
        self.archivo_entrada = archivo_entrada
        self.archivo_salida = archivo_salida
        self.freq_original = freq_original
        self.freq_objetivo = freq_objetivo
        self.frames = None
        self.parametros = None
        self.muestras_derechas = []

    def leer_archivo(self):
        with wave.open(self.archivo_entrada, "rb") as wav:
            self.parametros = wav.getparams()
            self.frames = wav.readframes(self.parametros.nframes)
        print(f"[INFO] Parámetros del WAV: {self.parametros}")

    def extraer_canal_derecho(self):
        n_canales = self.parametros.nchannels
        ancho_muestra = self.parametros.sampwidth
        n_frames = self.parametros.nframes

        for i in range(n_frames):
            offset = i * n_canales * ancho_muestra
            bytes_derecha = self.frames[offset + ancho_muestra : offset + 2 * ancho_muestra]
            muestra = struct.unpack('<h', bytes_derecha)[0]
            self.muestras_derechas.append(muestra)

    def invertir_muestras(self):
        self.muestras_derechas = list(reversed(self.muestras_derechas))

    def reducir_muestreo(self):
        factor = self.freq_original // self.freq_objetivo
        self.muestras_derechas = self.muestras_derechas[::factor]

    def guardar_archivo(self):
        with wave.open(self.archivo_salida, "wb") as wav:
            wav.setnchannels(1)  # Mono
            wav.setsampwidth(2)  # 16 bits
            wav.setframerate(self.freq_objetivo)

            frames_codificados = b''.join(struct.pack('<h', m) for m in self.muestras_derechas)
            wav.writeframes(frames_codificados)
        print(f"[INFO] Archivo procesado guardado como: {self.archivo_salida}")
```