
Laboratorio Práctico: Sistema de Modulación por Pulsos Codificados (PCM) con Arduino

Objetivos

1. Comprender las tres etapas fundamentales del PCM: Muestreo, Cuantificación y Codificación.
2. Verificar experimentalmente el Teorema de Muestreo de Nyquist-Shannon.
3. Implementar un sistema de transmisión digital y reconstrucción de la señal analógica usando Arduino.

Materiales

- 1 Placa Arduino Uno (o compatible).
- 1 Generador de funciones (o reproductor de audio con cable Jack 3.5mm) para generar la señal analógica dinámica. *(Nota: Si no se dispone de generador, se puede usar un potenciómetro para la versión básica).*
- 1 Protoboard y cables de conexión.
- Componentes para filtro paso bajo: 1 Resistencia de 10kΩ y 1 Capacitor de 0.1μF (104).
- Software Arduino IDE y software de análisis (Excel, MATLAB o Python).
- Osciloscopio (opcional, pero altamente recomendado para ver la señal reconstruida).

Descripción General

En este laboratorio se diseñará un sistema PCM básico. El Arduino actuará como el codificador: tomará muestras de una señal analógica, las cuantificará usando su Conversor Analógico-Digital (ADC) de 10 bits y las codificará. Posteriormente, se transmitirá esta información y se utilizará otro pin del Arduino (o un segundo Arduino) para demodular y reconstruir la señal analógica original mediante PWM y un filtro analógico.

Instrucciones

Paso 1: Acondicionamiento de la Señal de Entrada

1. Si usas un generador de funciones/audio, conecta la señal al pin `A0`.
Precaución: Asegúrate de que la señal no supere los 5V y esté centrada en un offset de 2.5V (usando un divisor de voltaje si es necesario) para no dañar el Arduino.
2. Si usas un potenciómetro (versión básica), conecta los extremos a 5V y GND, y la terminal central al pin `A0`.

Paso 2: Muestreo y Cuantificación (El ADC)

El Arduino Uno tiene un ADC de 10 bits, lo que significa que cuantifica la señal en $L = 2^{10} = 1024$ niveles.

Carga el siguiente código base. Notarás que usamos `micros()` para obtener marcas de tiempo precisas, fundamentales para calcular la frecuencia de muestreo real.

[Ver archivo "MuestreoCuantizacion.cpp"](#)

Paso 3: Análisis de Datos y Teorema de Nyquist

1. Abre el Monitor Serial. Copia los datos y guárdalos en un archivo `.csv` para abrirlo en Excel.
2. Cálculos a realizar en la hoja de cálculo:
 - Frecuencia de muestreo (f_s): Calcula la diferencia de tiempo entre muestras consecutivas. $f_s = 1 / T_s$.
 - Tasa de señalización o Bit Rate (R_b): Como cada muestra son 10 bits, $R_b = f_s \times 10$ bits/segundo.
 - Frecuencia máxima de la señal (f_{max}): Si usaste un generador de funciones, anota la frecuencia de la onda senoidal.
3. Pregunta de análisis: ¿Se cumple el Teorema de Nyquist ($f_s \geq 2 \times f_{max}$)? ¿Qué ocurre si aumentamos el `delayMicroseconds` a 5000? (Aliasing).

Paso 4: Transmisión y Demodulación (Reconstrucción de la Señal)

Para demostrar el ciclo completo del PCM, vamos a reconstruir la señal. Usaremos la salida PWM del Arduino como un "DAC" rudimentario, seguida de un filtro paso bajo (RC) para suavizar los pulsos y recuperar la forma de onda analógica.

Circuito de Reconstrucción:

1. Conecta el pin digital 9 del Arduino a una Resistencia de 10kΩ.
2. Conecta el otro extremo de la resistencia a un Capacitor de 0.1μF a GND.
3. La unión entre la Resistencia y el Capacitor es tu Salida Analógica Reconstruida (puedes medirla con un osciloscopio o un multímetro).

Código de Demodulación (Reemplaza el `loop` anterior o usa un segundo Arduino):

[Ver archivo "Demodulacion.cpp"](#)

Extensiones del Laboratorio (Nivel Avanzado)

- Filtro Digital (DSP): Antes de enviar el dato por serial, implementar un filtro de media móvil o un filtro IIR/FIR en el Arduino para reducir el ruido de cuantización.
 - Multiplexación (TDM): Si se tienen dos señales analógicas (ej. dos potenciómetros o dos canales de audio), codificarlas y transmitir las por el mismo canal serial alternando las muestras (Time Division Multiplexing).
 - Codificación de Línea: En lugar de enviar los datos por UART (Serial), usar un pin digital para enviar los 10 bits en serie (usando `shiftOut`) simulando un protocolo SPI o una codificación NRZ (Non-Return-to-Zero).
-

Conceptos Claves a Evaluar

- Muestreo: Proceso de tomar valores de la señal continua en instantes discretos de tiempo.
 - Cuantificación: Proceso de mapear un rango amplio de valores a un rango finito (el ADC de 10 bits del Arduino).
 - Codificación: Asignación de un código binario a cada nivel cuantificado.
 - Teorema de Nyquist: Para reconstruir una señal sin pérdida, la frecuencia de muestreo (f_s) debe ser al menos el doble de la frecuencia máxima contenida en la señal (f_{max}).
 - Ruido de Cuantificación: Error inherente al proceso de redondeo al digitalizar. En el Arduino de 10 bits, el paso de cuantización es $5V / 1024 \approx 4.88 \text{ mV}$.
-

Preguntas de Análisis

1. ¿Cuál es la resolución en voltios de tu sistema PCM?
2. ¿Qué sucede con la señal reconstruida si la frecuencia de muestreo es menor que el doble de la frecuencia de la señal original?
3. ¿Por qué el PWM de 8 bits introduce ruido adicional en la reconstrucción?
4. ¿Qué ventajas y desventajas tiene usar un filtro RC frente a un filtro digital para la reconstrucción?
5. Calcula el Bit Rate de tu sistema si la frecuencia de muestreo es de 2000 Hz.

Configuración Recomendada del Generador de Funciones

Parámetro	Valor Recomendado	Razón
Forma de onda	Senoidal (Sine)	Es la más fácil de analizar en frecuencia y observar distorsión.
Frecuencia (f)	50 Hz a 100 Hz	Con la frecuencia de muestreo del código (~1.5 kHz), esto te da entre 15 y 30 muestras por ciclo, cumpliendo holgadamente el Teorema de Nyquist y permitiendo una reconstrucción visible.
Amplitud (Vpp)	2 Vpp a 3 Vpp (Voltios pico a pico)	Deja un margen de seguridad para que la señal no toque los límites de 0V o 5V, evitando recorte (clipping).
Offset DC (Desplazamiento)	2.5 V (¡CRUCIAL!)	El Arduino NO tolera voltajes negativos. El offset centra la onda senoidal en 2.5V, haciendo que oscile de forma segura (ej. entre 1.0V y 4.0V si usas 3 Vpp).

Advertencia de Seguridad (¡Muy Importante!)

El pin A0 del Arduino Uno tiene diodos de protección internos que se activan si el voltaje es menor a 0 V o mayor a 5V.

- Si conectas una señal senoidal estándar de "1 Vpp" sin Offset, el generador entregará una señal que va de +0.5V a -0.5V. Ese -0.5V puede dañar el pin a largo plazo o generar lecturas erróneas (clipping en 0V).
- Siempre verifica con un osciloscopio o multímetro que la señal del generador esté completamente contenida entre 0V y 5V antes de conectarla al Arduino.

Para inyectar una señal senoidal desde un generador de funciones al Arduino en este laboratorio de PCM, debes tener mucho cuidado con los límites de hardware del microcontrolador.

Aquí tienes las ****recomendaciones exactas**** para que el experimento sea seguro, claro y didáctico:

Dinámica Didáctica Sugerida para el Laboratorio

Puedes dividir la práctica en dos fases para que los estudiantes vean la teoría en acción:

Fase 1: Muestreo Correcto (Cumple Nyquist)

- ****Configuración****: 100 Hz, 2 Vpp, Offset 2.5V.
- ****Frecuencia de muestreo del código****: `delayMicroseconds(500)` (~1.5 kHz real).
- ****Resultado esperado****: Al graficar en Excel o ver en el osciloscopio, la señal reconstruida se ve como una senoidal suave. Se cumple $f_s > 2 f_{max}$ ($1500 > 200$).

Fase 2: Demostración del Aliasing (Violación de Nyquist)

- ****Configuración****: Sube la frecuencia del generador a ****800 Hz o 1000 Hz**** (manteniendo 2 Vpp y Offset 2.5V).
- ****Frecuencia de muestreo del código****: `delayMicroseconds(500)` (~1.5 kHz).
- ****Resultado esperado****: Ahora $f_s < 2 f_{max}$ ($1500 < 2000$). Los estudiantes verán en la reconstrucción (o en la gráfica de Excel) una onda de ****frecuencia fantasma**** mucho más baja y distorsionada. Esto demuestra visualmente el ***aliasing*** y por qué el filtro anti-aliasing y el teorema de Nyquist son obligatorios en sistemas PCM reales.