Nombre:

Roxana Maribel Ramos Guzmán

Matrícula:

2015-2811

Carrera:

Mecatronica

Asignatura

Microcontroladores

Docente

Carlos Antonio Pichardo Viuque

Fecha de entrega:

22/08/2025

Investigación sobre teorema de muestreo

El teorema del muestreo, también conocido como teorema de Nyquist-Shannon, establece que una señal continua puede ser reconstruida sin pérdida de información a partir de muestras tomadas a una frecuencia (f_s) que sea al menos el doble de la frecuencia más alta (f_{max}) presente en la señal original ($f_s > 2 * f_{max}$). Este principio es fundamental para la conversión de señales analógicas a digitales y previene el efecto de aliasing, que ocurre cuando una frecuencia de muestreo demasiado baja provoca que las frecuencias altas se confundan con frecuencias bajas.

Investigación sobre el teorema de Nyquist

El Teorema de Nyquist, o Teorema de Muestreo de Nyquist-Shannon, establece que para reconstruir perfectamente una señal analógica a partir de sus muestras digitales, la frecuencia de muestreo (cuántas veces al segundo se toma una muestra) debe ser al menos el doble de la frecuencia más alta contenida en la señal original. Si esta condición no se cumple, ocurre un fenómeno llamado aliasing, donde las frecuencias más altas se "disfrazan" de frecuencias más bajas, distorsionando la señal reconstruida.

Investigación Transformada de Fourier

La Transformada de Fourier es una operación matemática que descompone una señal (en el dominio del tiempo o el espacio) en sus componentes de frecuencia, permitiendo su análisis en el dominio de la frecuencia. Este proceso revela patrones y características de la señal que son difíciles de identificar en su representación original. Su aplicación es fundamental en diversas áreas como el procesamiento de señales, la física, la ingeniería y la ciencia de datos, donde se utilizan algoritmos como la Transformada Rápida de Fourier (FFT) para un cálculo eficiente.

Ejemplo de uso del ADC con el ESP32/ STM32

Para usar el ADC en ESP32 o STM32, primero configuras el hardware (relojes, pines) y luego lees los valores analógicos en un ciclo, típicamente con un potenciómetro conectado, que puede medirse en rangos de 0 a 4095 en el ESP32 o configurarse con diferentes resoluciones y modos en el STM32.

Para ESP32 (usando el IDE de Arduino)

1. Conexión:

Conecta un potenciómetro al pin GPIO 32 (o cualquier otro pin analógico) y sus otros dos pines a los 3.3V y tierra del módulo ESP32.

2. Código:

Configuración (setup): No se requiere una configuración explícita en el IDE de Arduino, ya que los pines analógicos están listos para su uso.

Lectura (loop):

Lee el valor analógico con analogRead(32) para obtener un valor digital entre 0 y 4095, donde 0 es 0V y 4095 es 3.3V.

Imprime este valor en el monitor serie para ver los cambios al girar el potenciómetro

// Código de ejemplo para ESP32 usando el IDE de Arduino void setup() {
Serial.begin(115200); // Inicializa el monitor serie}

void loop() {

```
  // Lee el valor del potenciómetro conectado al GPIO 32

  int valorADC = analogRead(32);

  // Imprime el valor leído en el monitor serie

  Serial.print("Valor ADC: ");

  Serial.println(valorADC);
```

Para STM32 (con HAL y Code Composer Studio o Keil)

1. **1. Configuración del Hardware:**
   - Habilita el reloj para el periférico ADC usando el registro RCC.
   - Configura la fuente de reloj del ADC, por ejemplo, usando la fuente principal del sistema.
   - Configura el pin GPIO que se utilizará como entrada analógica.
2. **2. Configuración del ADC:**
   - Establece la resolución del ADC (por ejemplo, 12 bits para 4096 niveles).
   - Configura el modo de operación, como el modo de sondeo (polling).
3. **3. Inicio de la Conversión y Lectura:**
   - Inicia la conversión en el canal analógico deseado.
   - Espera a que termine la conversión.
   - Lee el valor digital convertido del registro del ADC.

```
  delay(100); // Pequeña pausa

}
```

// Ejemplo simplificado usando la biblioteca HAL de STM32 (Arduino-like para STM32)

// Esto asume que el ADC y el pin ya han sido configurados en el setup.

```
void loop() {
  // Inicia la conversión en el canal 0 (por ejemplo)
  HAL_ADC_Start(&hadc1);

  // Espera a que la conversión termine (en modo de sondeo)
  HAL_ADC_PollForConversion(&hadc1, 100);

  // Lee el valor convertido
  uint32_t valorADC = HAL_ADC_GetValue(&hadc1);

  // Puedes usar este valor para controlar algo, como un LED
  // ...
}
```