**Diseño de un Sistema de Adquisición de Señales Analógicas con Arduino**

Juan David Naranjo Jara y José Alejandro Pabón Leal  
  
  
  
Informa2  
  
  
  
12/09/24

**Introducción**

En este proyecto, se ha desarrollado una solución que permite medir y analizar las características de una onda electromagnética, incluyendo su amplitud, frecuencia y forma de onda, utilizando un Arduino Uno. Las señales que se analizarán pueden tomar diversas formas, como senoidal, triangular o cuadrada, y la correcta medición de estas propiedades es esencial en aplicaciones que requieren sincronización precisa de sistemas.

El principal objetivo es diseñar un algoritmo que permita adquirir datos de una señal y determinar sus características clave de manera eficiente y precisa, utilizando los recursos limitados del Arduino Uno, en particular su memoria dinámica.

**Análisis del Problema**

Las señales analógicas representan variaciones continuas de magnitudes físicas en el tiempo. En este proyecto, se procesarán señales generadas artificialmente que se leerán a través de una entrada analógica del Arduino Uno. El enfoque se centra en medir:

Forma de la señal: Identificada a partir de los datos adquiridos con el tiempo.

Precisión en las lecturas: Utilizando el ADC de 10 bits del Arduino, que ofrece un rango de 0 a 1023. Las señales rápidas pueden perder precisión por una baja tasa de muestreo.

Amplitud y Frecuencia: Las lecturas continuas detectarán los valores máximos y mínimos de la señal para calcular la amplitud y medir el tiempo entre picos para determinar la frecuencia.

Uso eficiente de la memoria: La memoria disponible en el Arduino Uno es limitada, lo que exige un control riguroso de la memoria dinámica (heap) para evitar sobrecargas.

Consideraciones para la Solución Propuesta

Memoria dinámica limitada: Con solo 2 KB de SRAM en el Arduino Uno, es esencial restringir la cantidad de datos almacenados, implementando un sistema que controle el uso del heap.

Precisión en la medición: El algoritmo debe identificar correctamente los picos máximos y mínimos de la señal para determinar su amplitud, y registrar el tiempo entre picos para calcular la frecuencia.

Clasificación de la forma de la onda: Se añadieron reglas más robustas para identificar correctamente la forma de la señal, lo que permite una mayor precisión en la clasificación de senoidales, triangulares y cuadradas.

1. Optimización de recursos

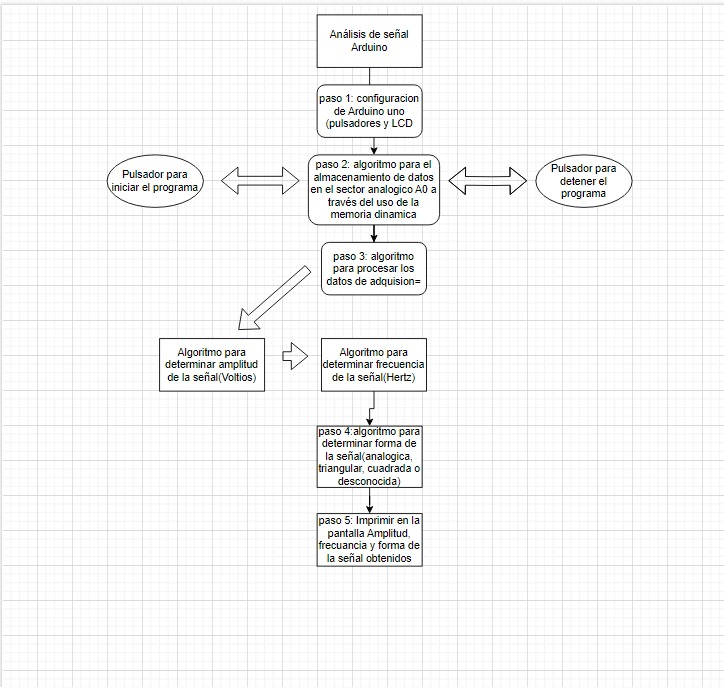
Uso de la memoria flash: Aprovechar la memoria flash del Arduino para almacenar datos invariables, liberando SRAM para el manejo de datos en tiempo real. Esto optimiza el uso de memoria, especialmente útil en aplicaciones con grandes volúmenes de datos.

Segmentación de funciones: Modularizar las funciones críticas para reducir el uso de SRAM durante la ejecución y optimizar el rendimiento del programa.

2. Control del heap y gestión de memoria dinámica

Umbral de memoria: Establecer un límite de 1000 bytes para evitar el agotamiento de la memoria dinámica, manteniendo la estabilidad del sistema durante la adquisición de datos.

Buffers circulares: Utilizar un buffer circular para sobrescribir los datos más antiguos una vez alcanzada la capacidad máxima, evitando el desbordamiento de memoria y manteniendo un flujo constante de adquisición de datos en tiempo real.

**Diagrama del análisis propuesto**

**Algoritmos Implementados**

Adquisición de Datos:

Se utiliza la función analogRead() para leer el valor de la señal analógica. Los datos se almacenan en un arreglo dinámico que se redimensiona cuando es necesario. También se captura el tiempo transcurrido con la función millis().

Detección de Picos

El algoritmo detecta los picos máximos y mínimos de la señal para calcular la amplitud:

int amplitud = maxVal - minVal;

Cálculo de Frecuencia:

Para calcular la frecuencia, se mide el tiempo entre dos picos máximos consecutivos. La fórmula usada es:

unsigned long periodo = time2 - time1;

float frecuencia = 1000.0 / periodo; // Frecuencia en Hz

Clasificación de la Forma de la Onda

El algoritmo clasifica la señal como senoidal, cuadrada, triangular o desconocida, según la suavidad o abrupto cambio en los valores leídos:

int determinarForma() {

// Comparación de cambios en los valores de la señal para determinar la forma

}

Control de la Memoria:

Se han implementado funciones para inicializar, redimensionar y liberar la memoria dinámica:

void redArr(int\*& arrDatos, unsigned long\*& tiempos, int& capacidad);

void liberarMemoria();

**Problemas de Desarrollo:**

Durante el desarrollo del proyecto, surgieron varios desafíos:

Rebotes en los botones: Se observó un comportamiento inconsistente debido a los rebotes. Para solucionarlo, se implementó una lógica de debounce, que filtra las pulsaciones accidentales de los botones.

Sobrecarga de memoria: La limitada memoria dinámica del Arduino resultó ser un obstáculo, ya que la adquisición continua de datos podía causar desbordamientos. Se resolvió utilizando un arreglo dinámico con redimensionamiento controlado y un límite en la cantidad de datos almacenados.

Precisión en la lectura de señales rápidas: En señales de alta frecuencia, la tasa de muestreo del Arduino no siempre es suficiente, lo que resultó en la pérdida de precisión. Se ajustó el algoritmo para optimizar el tiempo entre lecturas y minimizar los errores.

Evolución de la Solución

Inicialmente, el proyecto se enfocó únicamente en la adquisición y almacenamiento de datos. Sin embargo, con el tiempo se añadieron varias mejoras:

Manejo eficiente de la memoria: La memoria dinámica se administra mediante redimensionamiento controlado y liberación de memoria, permitiendo que el programa funcione sin sobrecargas.

Cálculo optimizado de amplitud y frecuencia: Se mejoró la precisión al detectar picos máximos y mínimos, ajustando los tiempos de muestreo para señales rápidas.

Clasificación de la forma de la onda: Se añadieron reglas más robustas para identificar correctamente la forma de la señal, lo que permite una mayor precisión en la clasificación de senoidales, triangulares y cuadradas.

En general, el proyecto ha evolucionado hacia una solución más robusta y precisa, considerando las limitaciones de hardware del Arduino Uno y optimizando el uso de sus recursos.