INFORME DESAFIO 1 INFORMATICA 2 – JUAN ANGEL OMAÑA MONTAÑEZ

a. Análisis del problema y consideraciones para la solución propuesta

El objetivo del proyecto era diseñar un sistema de adquisición y procesamiento de señales utilizando Arduino. Este sistema debía medir la frecuencia y la amplitud de una señal de entrada, así como identificar la forma de la onda entre opciones predefinidas: senoidal, cuadrada o triangular. Si la señal no correspondía a ninguna de estas formas, debía clasificarse como "desconocida".

Dada la limitada capacidad de memoria de Arduino y el entorno de simulación en Tinkercad, se propuso una solución basada en arreglos dinámicos, de modo que la cantidad de datos adquiridos no fuera un obstáculo en términos de memoria. Adicionalmente, se optó por una técnica de cálculo de la frecuencia basada en cruces por el punto medio de la señal en lugar de picos, evitando la complejidad matemática de técnicas más avanzadas, como la Transformada de Fourier, debido a su desconocimiento en la implementación. El algoritmo también incluye la capacidad de sobrescribir datos cuando el arreglo de almacenamiento se llena.

b. Esquema de tareas definidas en el desarrollo de los algoritmos

1. Inicialización del sistema:

   Se decidió configurar el entorno inicial para la adquisición de datos, comenzando por la pantalla LCD, los pines de botones para iniciar y detener la adquisición, y el pin analógico para leer la señal. Se utilizó la biblioteca Adafruit_LiquidCrystal para la gestión del LCD, y los pines de entrada digital con resistencia pull-up para los botones.

   Se asignaron variables para almacenar el estado de la adquisición de datos, los valores máximo y mínimo de la señal, el conteo de cruces por el punto medio, y las variables de frecuencia y amplitud.

2. Adquisición de datos:

   Esta fue una de las principales tareas, y su desarrollo se basó en la captura de la señal analógica y su almacenamiento en un arreglo dinámico. Se definió un intervalo de muestreo de 2 ms (equivalente a una frecuencia de 500 Hz), asegurando una tasa de muestreo adecuada para la mayoría de las señales.

   Gestión de memoria dinámica: Se implementó el uso de un arreglo dinámico (int16_t* arr) para almacenar los datos de la señal. Dado que Arduino tiene una

cantidad limitada de memoria, se optó por un sistema de sobrescritura en el arreglo cuando este llegaba a su capacidad máxima. Este enfoque permite una adquisición continua sin preocuparse por el desbordamiento de memoria.

3. Cálculo de la amplitud:

   Durante la adquisición, se mantenía un seguimiento constante de los valores máximo y mínimo de la señal. Esto facilitó el cálculo de la amplitud al finalizar la adquisición.

   Estrategia de cálculo: La amplitud se calcula tomando la diferencia entre el valor máximo y mínimo de la señal adquirida y escalando este valor al rango de 5V (el valor máximo para la señal analógica).

4. Cálculo de la frecuencia:

   La frecuencia se calculó utilizando los cruces por el punto medio de la señal. Esta tarea implicó identificar los momentos en que la señal cruzaba un valor promedio entre el máximo y el mínimo de la onda, utilizando este conteo de cruces para calcular la frecuencia.

   División de ciclos: Cada cruce por el punto medio representa la mitad de un ciclo completo de la onda, por lo que la frecuencia se obtiene dividiendo los cruces por 2 y luego ajustando según el tiempo total de adquisición.

5. Identificación del tipo de onda:

   Se estableció una serie de reglas heurísticas para identificar la forma de la onda. Las ondas posibles a identificar eran cuadradas, senoidales y triangulares, con la clasificación "Desconocida" en caso de no cumplir con las características esperadas.

   Detección de onda cuadrada: La forma cuadrada fue identificada mediante la existencia de solo dos niveles de voltaje (alto y bajo) durante todo el ciclo de la señal.

   Detección de onda senoidal: Se implementó un análisis de pendiente antes y después de los picos de la señal, buscando una variación gradual de la pendiente característica de una onda senoidal.

   Detección de onda triangular: En la onda triangular, las pendientes son constantes y simplemente cambian de signo. Se implementó un algoritmo que verificaba la igualdad en valor absoluto de las pendientes antes y después de cada pico.

6. Visualización de los resultados:

El sistema fue diseñado para mostrar los resultados en una pantalla LCD de 16x2, donde se presentaban los valores de amplitud y frecuencia, así como la identificación de la onda. Adicionalmente, se enviaron los mismos resultados al monitor serial de Arduino para su verificación.

c. Algoritmos implementados

1. iniciarAdquisicion()

   Esta función inicializa el proceso de adquisición de datos cuando se presiona el botón de inicio. Primero, libera la memoria del arreglo dinámico existente y luego reasigna un nuevo arreglo con la capacidad especificada. Se reinician las variables clave como el índice actual (`indiceActual`), los valores máximos y mínimos (`maxValue` y `minValue`), y el contador de cruces por el punto medio (`crucePorCero`). Se establece el estado de adquisición de datos a verdadero (`acquiringData = true`) y se guarda el tiempo de inicio utilizando `millis()` para llevar el control del tiempo de muestreo. Además, se actualiza el LCD para mostrar que la adquisición de datos ha comenzado.

2. detenerAdquisicion()

   Cuando se presiona el botón de detener, esta función finaliza la adquisición de datos. Cambia el estado de adquisición a falso, calcula el tiempo total transcurrido desde el inicio de la adquisición, y llama a las funciones que calculan la amplitud **calcularAmplitud()**, la frecuencia **calcularFrecuencia()** y el tipo de onda **identificarTipoOnda()**. Posteriormente, los resultados de la amplitud, frecuencia y tipo de onda se muestran en el LCD y se envían al monitor serial para análisis. Este módulo asegura que se detiene adecuadamente la captura de datos y se procesan para obtener información relevante de la señal.

3. adquirirDatos()

   Esta función es el núcleo de la adquisición continua de datos mientras el estado **acquiringData** es verdadero. Cada vez que se llama, lee el valor analógico desde el pin A0 y lo almacena en el arreglo **arr**. Luego, actualiza los valores máximos y mínimos para la medición de amplitud, y cuenta los cruces por el punto medio de la señal para la medición de la frecuencia. Si el arreglo alcanza

su capacidad, los valores nuevos sobrescriben a los más antiguos, lo que permite una captura continua sin desbordar la memoria. Un pequeño delay asegura una frecuencia de muestreo de 500 Hz, manteniendo la precisión en la adquisición de la señal.

4. calcularAmplitud()

   Esta función calcula la amplitud de la señal analógica utilizando la diferencia entre el valor máximo **maxValue** y el valor mínimo **minValue** almacenados durante la adquisición de datos. Luego, se convierte este rango de valores analógicos (de 0 a 1023) a voltios utilizando la relación de 5V/1023, que es la resolución del conversor analógico-digital del Arduino. Esto da como resultado la amplitud de la señal en voltios, que posteriormente es mostrada en el LCD y el monitor serial.

5. calcularFrecuencia()

   Este módulo calcula la frecuencia de la señal basándose en el número de cruces por el punto medio detectados durante la adquisición. Cada vez que la señal cruza el punto medio, el contador de cruces se incrementa. Para calcular la frecuencia, se dividen los cruces por dos (para obtener ciclos completos) y se divide por el tiempo total de adquisición en segundos, dando como resultado la frecuencia en Hertz. Si no se detectan suficientes cruces, la frecuencia se establece en cero.

6. identificarTipoOnda()

   Esta función analiza los datos adquiridos para determinar el tipo de onda (cuadrada, senoidal, triangular o desconocida). Primero verifica si hay suficientes datos para realizar la identificación. Luego, invoca subfunciones específicas como identificarOndaCuadrada(), identificarOndaSenoidal(), e identificarOndaTriangular(), que analizan las características de la señal, como los niveles y las pendientes de los picos, para determinar su forma. Si no

coincide con ninguna de las formas conocidas, el tipo de onda se marca como "Desconocida".

7. Subfunciones de identificación de onda

   Las subfunciones `identificarOndaCuadrada(), identificarOndaSenoidal() y identificarOndaTriangular() implementan la lógica para identificar cada tipo de onda. Para la onda cuadrada, verifica si la señal solo tiene dos niveles distintos, característicos de una señal cuadrada. Para la onda senoidal, revisa si las pendientes antes y después de los picos cambian de manera gradual, una característica de la señal senoidal. Para la onda triangular, verifica que las pendientes antes y después de los picos sean constantes, pero con signos opuestos, lo que indica una forma triangular. Si ninguna de estas condiciones se cumple, el tipo de onda se establece como desconocido.

d. Problemas de desarrollo afrontados

Durante el desarrollo se enfrentaron varios desafíos técnicos y de diseño:

1. Sobreescritura de valores en el arreglo: Inicialmente, el arreglo de almacenamiento se llenaba y generaba errores, lo que se resolvió implementando un sistema de sobrescritura cuando se alcanzaba la capacidad máxima.

2. Ruido en la señal analógica: El ruido presente en las lecturas dificultaba la identificación precisa de las formas de onda, especialmente en señales senoidales y triangulares. Esto llevó a realizar ajustes en los umbrales de detección de pendientes y cambios de nivel.

3. Frecuencia de muestreo: Se debía ajustar la frecuencia de muestreo para asegurar que las mediciones fueran representativas de las señales a diferentes frecuencias.

4. Limitaciones de memoria: Dado que Arduino tiene una capacidad limitada, fue necesario utilizar memoria dinámica para la adquisición de datos y evitar el desbordamiento de memoria.

5. Identificación de la onda senoidal y triangular: A ciertas frecuencias y amplitudes, la identificación entre senoidal y triangular se volvió imprecisa debido a las limitaciones del algoritmo de pendientes.

e. Evolución de la solución y consideraciones para la implementación

El proyecto pasó por varias iteraciones, desde la implementación de un arreglo de almacenamiento estático, cambiar el tipo de dato de entrada a `int16_t` para manejar mayor cantidad de datos, hasta la integración de memoria dinámica para evitar el desbordamiento. Se mejoró el algoritmo de detección de cruces por el punto medio para calcular la frecuencia, y se refinó el método de identificación de la onda.

La mayor dificultad fue la diferenciación entre señales senoidales y triangulares, que dependen de la precisión de las pendientes y del nivel de ruido presente en las lecturas. Para mejorar la robustez del sistema, se implementaron verificaciones adicionales para validar la suavidad de los cambios en la pendiente (en el caso de la onda senoidal) y la constancia de la misma (en la onda triangular). Además, se añadieron controles para sobrescribir datos cuando el arreglo alcanzaba su capacidad máxima, mejorando así la estabilidad del sistema.

Se concluye que el sistema cumple con los objetivos principales del proyecto, aunque la detección precisa de ondas senoidales y triangulares sigue siendo un desafío en frecuencias y amplitudes específicas.