DESAFIO I

INFORMATICA II

ALFONSO DIAZ LINA YOJANA  
RINCON SALDARRIAGA SEBASTIAN

GUERRA ANIBAL JOSE

UNIVERSIDAD DE ANTIOQUIA

2024-2

1. **Análisis del problema y consideraciones para la alternativa de solución propuesta.**

Para abordar el problema planteado, se requiere un análisis más detallado de las funcionalidades necesarias y una solución estructurada en términos de procesamiento de señales y programación en Arduino. A continuación, se presenta un esquema de la solución, sin utilizar código, pero detallando los pasos clave:

1. Montaje en Tinkercad

El sistema propuesto debe estar montado en la plataforma Tinkercad con los siguientes componentes:

1. Arduino UNO: Es el microcontrolador encargado de gestionar las entradas y salidas.
2. Generador de señales: Este generador emite señales analógicas que varían en frecuencia, amplitud y forma de onda. La señal se conectará a uno de los pines analógicos del Arduino para su lectura.
3. Pulsadores: Se utilizarán dos pulsadores:
   * El primero inicia la adquisición de datos. Cuando el usuario lo presiona, el programa comienza a capturar y almacenar los datos de la señal.
   * El segundo pulsador se utilizará para detener la adquisición de datos y visualizar los resultados en la pantalla LCD.
4. Pantalla LCD: Utilizando una pantalla LCD, se mostrará la frecuencia, amplitud y el tipo de señal detectada.

Conexiones:

* Los pulsadores se conectarán a pines digitales, con resistencias pull-down para evitar interferencias eléctricas.
* La pantalla LCD se conectará utilizando la librería adecuada, como Adafruit\_LiquidCrystal.h, que simplifica el manejo de la pantalla desde el código.
* El generador de señales se conectará a uno de los pines analógicos del Arduino para capturar los datos de la señal en tiempo real.

1. Registro de la señal en un arreglo de apuntadores

El programa debe capturar los datos de la señal en tiempo real y almacenarlos en un arreglo de apuntadores. Este método es eficiente porque permite gestionar de forma dinámica una gran cantidad de datos, almacenando solo los valores relevantes.

La captura de la señal debe realizarse en intervalos regulares de tiempo, lo que genera una serie de valores que representan la señal. Estos valores se almacenan en un arreglo que luego será utilizado para analizar las características de la señal, como la frecuencia, la amplitud y el tipo de onda.

Cada vez que la señal cambia, por ejemplo, en su frecuencia o amplitud, el programa debe almacenar estos cambios para analizarlos posteriormente.

1. Cálculo y amplitud de la señal

Frecuencia: Se determina calculando cuántas veces la señal cruza el valor cero, es decir, el punto donde la señal cambia de positivo a negativo o viceversa. Cada cruce por cero representa un ciclo de la señal. Para calcular la frecuencia:

1. Se detecta cada cruce por cero en los datos almacenados.
2. Se mide el tiempo entre dos cruces consecutivos.
3. La frecuencia se calcula como la inversa del tiempo entre esos cruces (es decir, el número de ciclos por segundo).

Este método es preciso porque toma en cuenta el comportamiento natural de una onda periódica, asegurando que cualquier tipo de señal, ya sea senoidal, cuadrada o triangular, se mida correctamente.

Amplitud: Se calcula tomando los valores máximos y mínimos de la señal durante el período de adquisición. La amplitud es simplemente la mitad de la diferencia entre el valor máximo y el valor mínimo de la señal. Esto es crucial para determinar la "altura" de la onda.

El proceso consiste en:

1. Encontrar el valor más alto y bajo de la señal capturada.
2. Restar el valor mínimo del valor máximo.
3. Dividir este resultado entre dos para obtener la amplitud.
4. Identificación del tipo de señal

Después de capturar los datos de la señal, el siguiente paso es determinar el tipo de onda. Existen varias formas de onda posibles: senoidal, cuadrada, triangular, etc. El algoritmo debe analizar la señal capturada para identificar su forma. Esto se puede hacer de la siguiente manera:

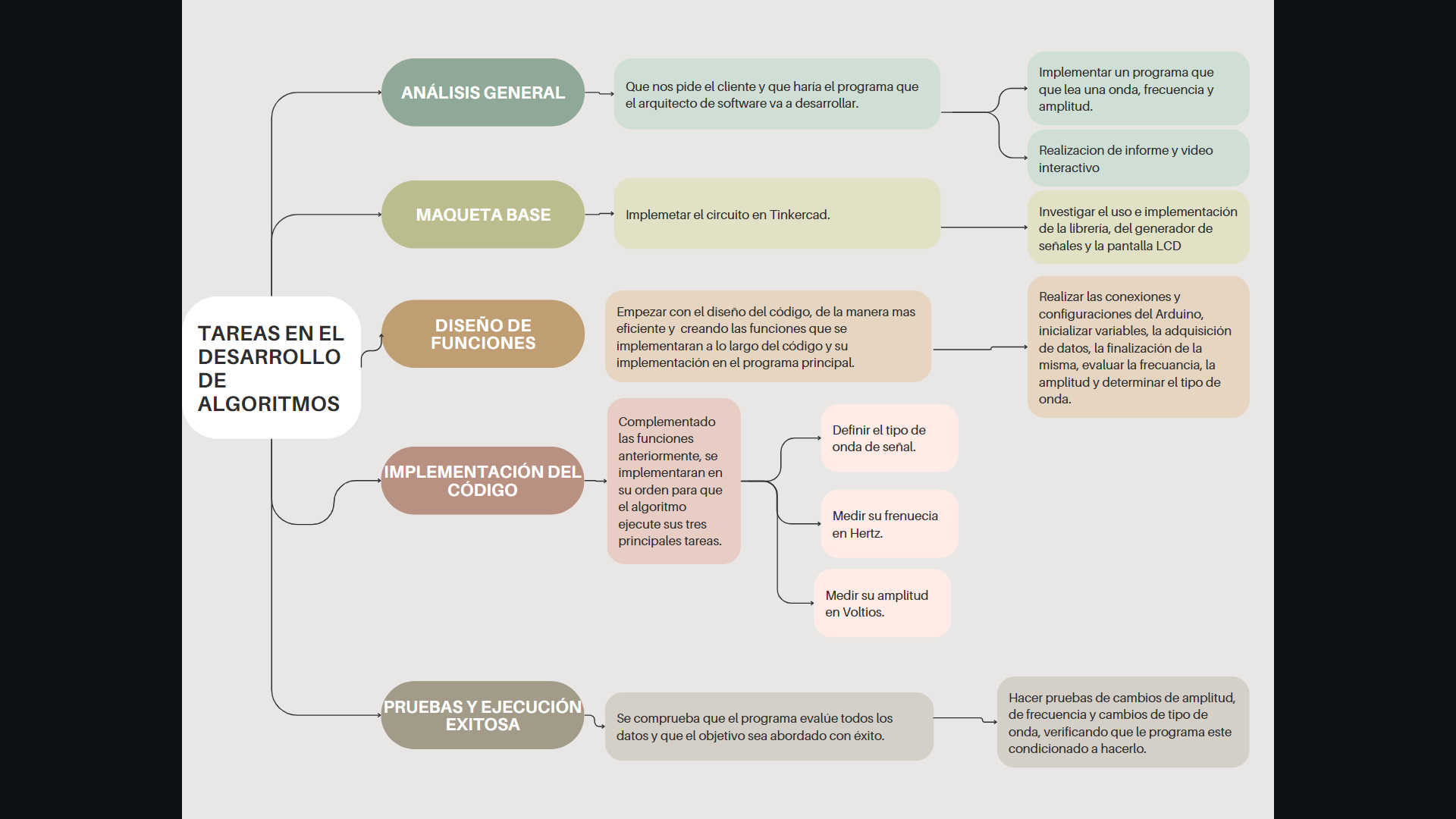
1. Onda senoidal: El análisis debe buscar patrones suaves en los valores capturados, donde no haya cambios bruscos entre muestras consecutivas.
2. Onda cuadrada: Para identificarla, el algoritmo buscará cambios abruptos en la señal, donde los valores suban o bajen drásticamente de un extremo al otro.
3. Onda triangular: Se identificará si la diferencia entre valores consecutivos es constante o lineal.
4. Señal desconocida: Si la señal no corresponde a ninguna de las formas estándar, se clasificará como desconocida. Esto puede ocurrir si la señal tiene un comportamiento errático o no cumple con los patrones esperados de una onda senoidal, cuadrada o triangular.

Flujo de la solución:

1. El usuario presiona el primer pulsador, lo que inicia la adquisición de datos de la señal.
2. El programa comienza a capturar los valores de la señal de entrada y a almacenarlos en un arreglo de apuntadores.
3. Una vez que el usuario presiona el segundo pulsador, la adquisición se detiene, y el programa procesa los datos capturados.
4. El algoritmo analiza los datos para determinar la frecuencia y amplitud de la señal.
5. Finalmente, se analiza la forma de la onda y se clasifica como senoidal, cuadrada, triangular o desconocida.
6. Los resultados (frecuencia, amplitud y tipo de señal) se muestran en la pantalla LCD.

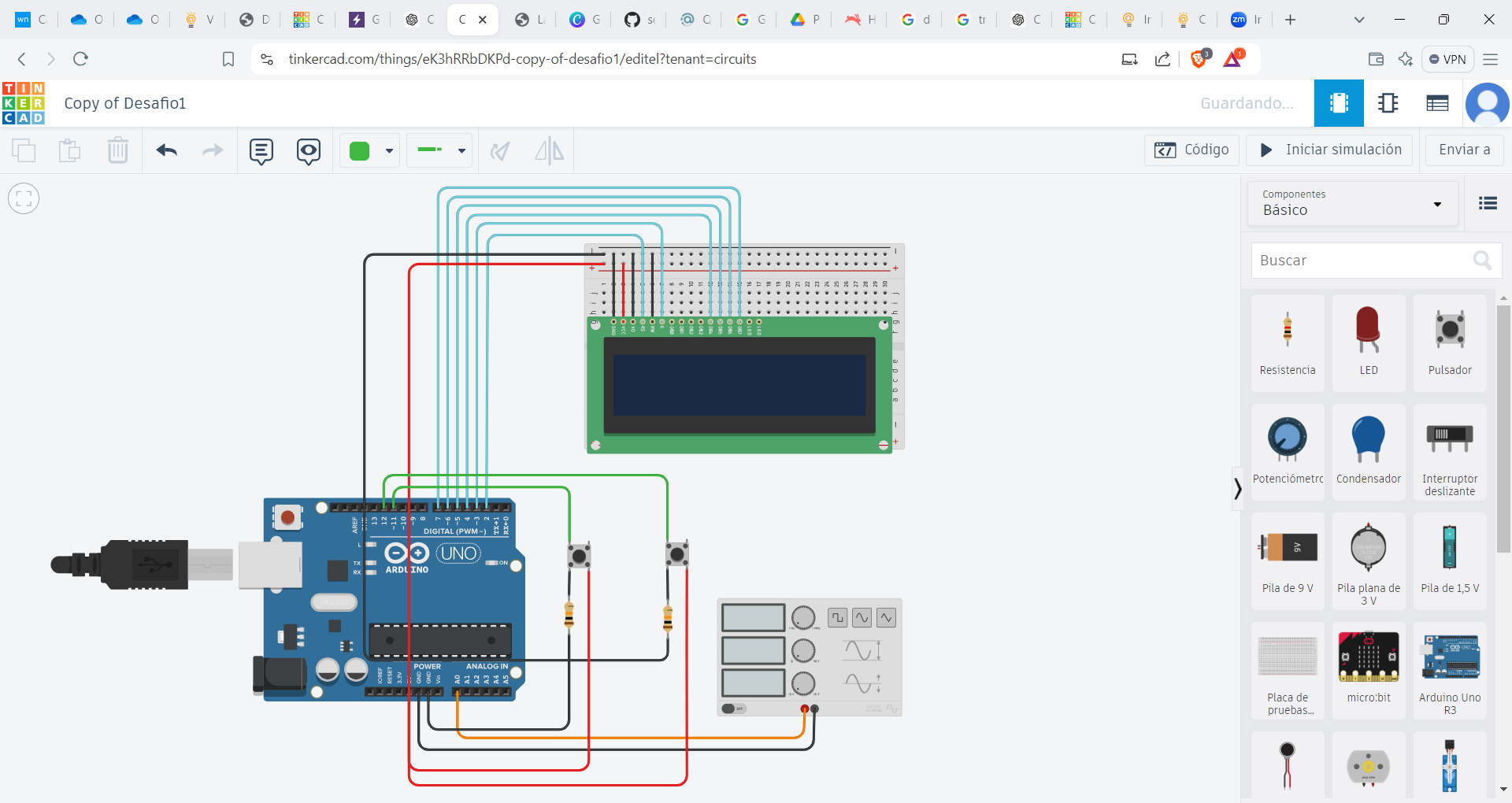
Con este enfoque estructurado, se garantiza que el sistema pueda identificar y procesar de manera eficiente las características principales de la señal analógica, cumpliendo con los requerimientos del proyecto.

1. **Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.**



1. **Algoritmos implementados.**

|  |  |
| --- | --- |
| **CONECTOR** | **CONEXION** |
| GND | Tierra |
| Entrada analógica | A0 |
| Entradas / salida digital | 2, 3, 4, 5, 6, 7 |
| Pulsador de inicio | 11 |
| Pulsador de fin | 12 |



|  |  |  |
| --- | --- | --- |
| NOM\_FUNCION | PARAMETROS | DESCRIPCION |
| calcularFrecuencia (float) | Recibe un arreglo (int\* datos), cantidad de datos (int cantidad), punto cero de la función (float puntoCero) y tiempo en que se realizó la adquisición (unsigned long duración). | Cuenta los cruces por el punto cero (promedio de los valores máximo y mínimo) en un arreglo de datos de la señal, determinando cuántas veces pasa de positivo a negativo. Con el número de cruces y el tiempo total de adquisición, calcula la frecuencia en Hertz (Hz) usando la fórmula f=N/T, donde N es el número de cruces y T es el tiempo de muestreo. |
| calcularAmplitud (float) | El valor máximo adquirido (int valorMax) y el valor mínimo adquirido (int valorMin) dentro de la señal generada. | Toma los valores máximo y mínimos medidos durante la adquisición de datos, calcula la amplitud restando el valor mínimo del valor máximo y luego convierte este rango a voltios, utilizando el voltaje de referencia del ADC para obtener una medida en un rango de voltaje. |
| redArr (void) | Recibe un arreglo de datos (int\*& arr) y la nueva capacidad que se va a sumar a un nuevo arreglo (int& capacidad). | Aumenta el tamaño de un arreglo dinámico de enteros, crea un nuevo arreglo más grande, copia los datos del arreglo original al nuevo, libera la memoria del arreglo antiguo y actualiza el puntero para que apunte al nuevo arreglo, permitiendo así almacenar más datos sin perder los existentes. |
| IdentificarTipoOnda (string) | Recibe el arreglo (int\* datos), la cantidad de valores (int cantidad), y el tiempo de duración de la adquisición (unsigned long duración). | identifica dos valores únicos para clasificar una señal como "Cuadrada". Si hay cambios en la pendiente al recorrer el arreglo, determina si la señal es "Triangular" (si el tiempo por punto es menor a 1 ms) o "Senoidal" (si es mayor o igual a 1 ms). Si no se cumplen estas condiciones, clasifica la señal como "Desconocida". |

* 1. **Problemas de desarrollo que afrontó.**

Durante el proceso de análisis y desarrollo del algoritmo, se enfrentaron varias dificultades clave. Estos problemas son relevantes tanto en el diseño del algoritmo como en la implementación práctica. A continuación, se describen los principales desafíos abordados:

1. Optimización del Análisis y Diseño del Algoritmo: La fase de análisis inicial tuvo que abordar cómo implementar la solución de manera eficiente dentro del tiempo disponible. Aunque el análisis preliminar fue útil, se descubrieron áreas de mejora durante la implementación. A medida que avanzábamos, identificamos consideraciones adicionales que optimizaron el rendimiento del programa. Estos ajustes mejoraron la eficiencia y la precisión del análisis, asegurando que la solución final se alineara mejor con los objetivos del proyecto.
2. Medición de Frecuencia y Manejo de Cruces por Cero: Uno de los desafíos principales fue la medición precisa de la frecuencia de la señal. Inicialmente, se asumió que la señal siempre cruzaría el punto cero, lo cual no siempre era el caso, especialmente con señales que tienen desfases o que no pasan por el punto cero. Para abordar esto, implementamos un método para calcular el punto cero dinámicamente en función de los valores máximos y mínimos detectados durante la adquisición de datos. Sin embargo, la alta frecuencia de la señal presentó problemas de resolución, ya que el Arduino no puede capturar todas las muestras a una velocidad infinita. Esto llevó a una limitación en la precisión de la frecuencia medida, pues el sistema no puede muestrear la señal con la resolución deseada a frecuencias muy altas.
3. Cálculo de Amplitud y Resolución de Datos: La medición de la amplitud presentó problemas relacionados con la variabilidad de la señal y la resolución del sistema. Cuando la amplitud de la señal es muy baja, la onda puede parecer casi una línea recta, lo que dificulta la medición precisa de su amplitud. Para mitigar este problema, se implementó un enfoque que divide la onda en ciclos y compara la amplitud de cada ciclo para determinar cuál es el valor más representativo. Esto permitió obtener una medida de amplitud más precisa, incluso cuando la señal tiene una baja resolución.
4. Identificación y Manejo del Tipo de Onda: La identificación del tipo de onda y su correcta representación en el sistema también representó un desafío. Diferentes tipos de ondas pueden presentar características diversas que afectan la medición de frecuencia y amplitud. La implementación del algoritmo tuvo que considerar las distintas formas de onda que podrían ser medidas y adaptarse a estas variaciones para proporcionar resultados precisos. La solución final incluyó una adaptación para manejar diferentes tipos de señales y asegurar que los cálculos de frecuencia y amplitud fueran adecuados para la forma de onda detectada.
5. Para clasificar una onda como triangular, se requiere un patrón específico de cambios de pendiente. Si los datos muestreados no presentan suficiente claridad o están contaminados por ruido, la función puede fallar en reconocer la onda triangular, una frecuencia de muestreo insuficiente podría resultar en la pérdida de detalles cruciales de la forma de onda, lo que impide una identificación precisa. La lógica actual de clasificación puede no ser robusta. Aunque se cuentan los cambios de pendiente, esto podría no ser suficiente para clasificar una señal como triangular. Si las condiciones para la identificación no están bien definidas, se corre el riesgo de obtener clasificaciones erróneas. La pantalla LCD tiene un espacio limitado, lo que puede dificultar la visualización clara de múltiples métricas (como frecuencia, amplitud y tipo de onda) al mismo tiempo. Si se actualiza la pantalla demasiado rápido, el usuario puede no tener tiempo suficiente para leer la información presentada. Cualquier problema en la adquisición de la señal analógica o en el proceso de muestreo puede resultar en valores incorrectos en la pantalla, afectando la interpretación de la señal.

Estos desafíos reflejan la complejidad de diseñar un sistema de medición de señales precisas utilizando un microcontrolador, y destacan la importancia de una planificación cuidadosa y ajustes iterativos durante el desarrollo.

* 1. **Evolución de la solución y consideraciones para tener en cuenta en la implementación.**

La solución para la medición de frecuencia y amplitud de señales analógicas ha evolucionado significativamente a través del proceso de desarrollo. A continuación, se describen las principales mejoras realizadas y las consideraciones importantes para futuras implementaciones:

* 1. Inicialmente, se asumió que el punto cero de la señal sería un valor constante. Se descubrió que calcular el punto cero dinámicamente a partir de los valores máximos y mínimos detectados proporcionaba resultados más precisos. Esta mejora abordó problemas de precisión en la medición de frecuencia, especialmente para señales que no cruzan un punto cero fijos.
  2. El problema de la resolución en altas frecuencias se abordó con una estrategia que ajusta el intervalo de muestreo y el cálculo de cruces por cero. Aunque el Arduino tiene limitaciones en la velocidad de muestreo, el algoritmo optimizado permite una mejor resolución y precisión en el cálculo de la frecuencia dentro de sus capacidades.
  3. Para mejorar la precisión en la medición de la amplitud, se implementó un enfoque que divide la señal en ciclos y evalúa la amplitud de cada ciclo individualmente. Esto ayuda a superar las limitaciones de resolución cuando la señal es muy baja, proporcionando una medida más representativa de la amplitud real de la señal.
  4. La solución final incluye capacidades para manejar diferentes tipos de ondas, adaptando el algoritmo a las características específicas de la señal medida. Esto asegura que la medición de frecuencia y amplitud sea adecuada para una variedad de formas de onda, mejorando la versatilidad y precisión del sistema.

Consideraciones para Futuras Implementaciones:

1. Mejoras en el Hardware: Para señales de alta frecuencia o amplitud muy baja, considerar el uso de hardware más avanzado o técnicas de muestreo más precisas puede mejorar la resolución y exactitud del sistema.
2. Optimización del Código: Continuar optimizando el código para reducir el tiempo de procesamiento y mejorar la eficiencia, especialmente en sistemas con limitaciones de recursos como el Arduino.
3. Adaptación a Diferentes Señales: Desarrollar mecanismos adicionales para identificar y adaptar el análisis a señales con características no estándar, garantizando que el sistema pueda manejar una amplia gama de tipos de onda, asegurando una frecuencia y amplitud exactas independientemente de los cambios a lo largo de la señal.

La evolución de la solución ha permitido abordar de manera efectiva los desafíos iniciales, mejorando la precisión y flexibilidad del sistema de medición de señales. Estas consideraciones son fundamentales para garantizar que el sistema sea en su totalidad eficiente y preciso en aplicaciones futuras.