Informe Desafío I

Juan Manuel Álvarez Cajiao

Jean Carlos Parra Serrano

Informática 2

Aníbal José Guerra Soler

Augusto Enrique Salazar Jiménez

Universidad de Antioquia

Facultad de Ingeniería

Medellín

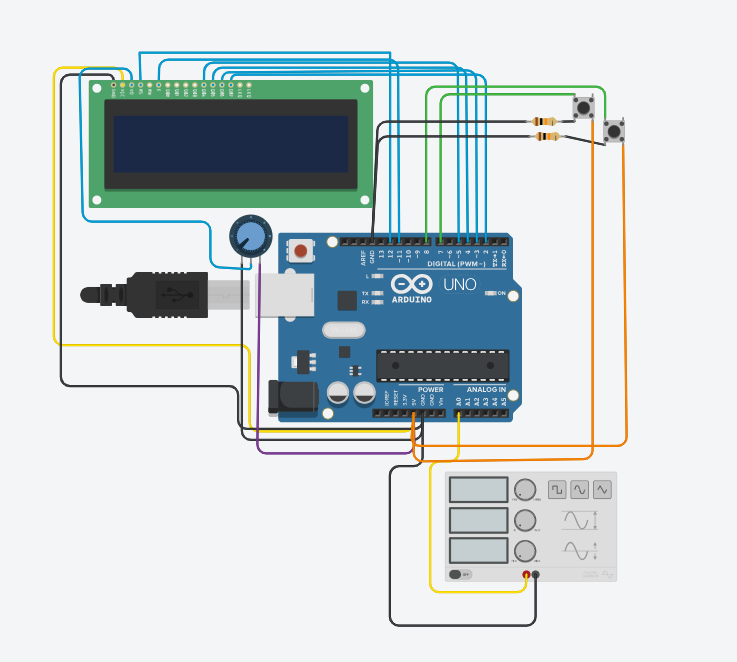
2024

**Análisis del problema y solución propuesta:**

El Desafío I plantea una serie de problemas, los cuales están relacionados entre sí. El primer problema que surge es la creación de un circuito, en el cual se basara el código para dar solución al resto de problemas planteados. Para lograr dar con los componentes adecuados del circuito, se debe analizar primero cuales son los otros problemas a resolver; previamente se pide utilizar un generador de señales, ya que con este se deberán hallar diferentes datos. Los datos que se deberán hallar son: tipo de señal que está produciendo el generador de señales (senoidal, cuadrada, triangular o no definida), la amplitud de esta en voltios y su frecuencia en Hertz; esto se debe imprimir en una pantalla LCD; sin embargo, los datos a analizar están definidos en un rango que lo determinara dos pulsadores, uno marcando el inicio y otro el final.

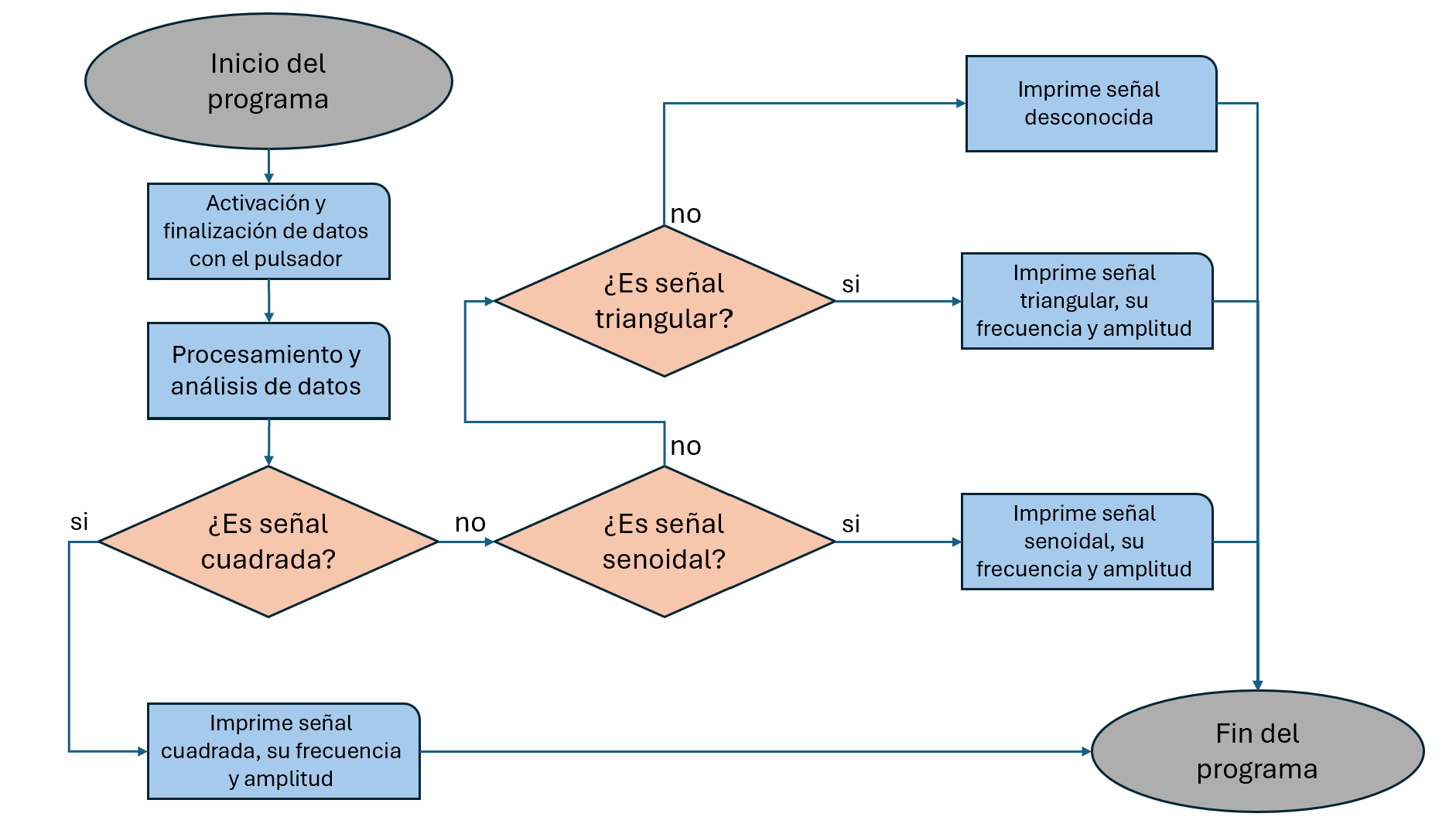
Los componentes del circuito en el que se planteó la solución fueron los siguientes: un Arduino uno R3, una LCD 16 x 2, un potenciómetro, un generador de señales, dos pulsadores y dos resistencias.

El diseño propuesto fue el siguiente:



La razón por la cual se propuso este circuito para dar solución al problema es porque cumple con los requisitos dados por el problema; se tuvo que implementar la presencia de resistencias y un potenciómetro para que este funcione correctamente. Al acomodar todas las partes del circuito se inicia la creación del código, el cuál es el elemento fundamental para la solución de la mayoría de los requisitos solicitados.

Para entender introducir un poco al análisis del código principal, se da el siguiente diagrama de flujo:



El programa consiste en lo siguiente: Cuando se inicia el programa, se da la recolección de datos al oprimir el pulsador y cuando se oprime el otro pulsador finaliza la recolección de datos, el tiempo entre dato y dato recolectado es aproximadamente de 0.1 segundos. Estos datos se guardan en un arreglo dinámico para hacer uso de la memoria dinámica, apuntadores y demás. Este arreglo sirve, entre otras cosas, para calcular la amplitud de la onda, donde se toma la diferencia entre el valor máximo y el valor mínimo, y se divide en dos; de esta forma se halla la amplitud en voltios.

Uno de los problemas fundamentales es saber qué tipo de onda es la señal. El análisis de la solución es que una señal es cuadrada cuando hay un cambio constante entre dos valores; si se determina que tiene más de dos valores, deja de ser cuadrada y pasaría a ser otro tipo de función. Para una señal senoidal, sus valores deben oscilar entre cierto rango específico, con su amplitud y frecuencia. Y, finalmente, para una señal triangular, debe tener una pendiente específica positiva y negativa. Si al recorrer todas estas funciones y no es ninguna de ellas, la señal será desconocida.

La mayoría del código consiste en varias funciones que recibirán el arreglo de datos guardados y su tamaño. Está la función para calcular la amplitud, función para calcular cada tipo de señal (cuadrada, senoidal, triangular). El código principal se lleva a cabo en la función *void loop*, la cual hace uso de todas estas funciones para su correcto uso, invocación y procesamiento.

En el desarrollo de los algoritmos surgieron varios problemas, en especial en las funciones triangulares y senoidales. Al tener una cantidad poca de datos a través del tiempo y con saltos no constantes; para la señal triangular, se debe hacer ciertas aproximaciones en el cálculo de las pendientes (positiva cuando crece y negativa cuando decrece). Se planteó la solución de que, al recorrer el arreglo, se calcule la pendiente y se compare las pendientes con un rango de error de 1.5 sobre la pendiente inicial. El rango de error también fue una estrategia que se implementó en la función para reconocer si los datos del arreglo pertenecen al comportamiento de una función senoidal, ya que la media de tiempo entre dato y dato es de 0.1 segundos. No se tenía una cantidad suficiente de datos para hallar un valor exacto. La estrategia usada para la confirmación de si es o no una función senoidal, fue un valor medio entre dato y dato, dado que la función senoidal es una función que no tiene cambios bruscos en su comportamiento, la “distancia” entre el valor 1 y el valor 2, debe ser la misma o aproximada entre el valor i y el valor i+1, siempre teniendo como referencia la primera “distancia”, esto con el fin de que no haya un error creciente.

El problema con el rango de error, por la insuficiencia de cantidad de datos, es que una función triangular puede cumplir en algunos casos los requisitos para ser función senoidal y viceversa, ya que el rango de error es bastante amplio, también puede suceder que las señales, a pesar de que sean triangulares o senoidales, no se les pueda identificar como señal definida, porque las operaciones para hacer la confirmación de a qué tipo de señal pertenecen superen este rango de error; esto puede suceder si la plataforma arroja un valor a destiempo.

Para hallar la frecuencia de una señal cuadrada debemos de recorrer el arreglo, y como son dos valores, tomamos el primer dato, luego debemos mirar cuántos valores hay desde ese primer dato recolectado hasta que el siguiente valor sea distinto al primer dato y determinar cuando vuelva a ser el primer dato, para ahí determinar que se completó un ciclo. Como el circuito arroja datos cada 0.1 segundos, multiplicamos cuántos datos hay desde que inicia el ciclo hasta que termine por 0.1; eso sería el periodo de la función y como la frecuencia es el inverso, se divide 1 entre el periodo.

Ahora bien, la frecuencia de las señales triangulares y senoidales, se pueden calcular de una forma simular entre ellas. Se toma el valor máximo de la señal y su valor mínimo, se determina cuántos datos hay entre es rango y se multiplica por 2, ya que el ciclo terminaría en la otra cresta. Lo anterior, se multiplica por 0.1 que son los segundos que hay entre cada dato y determinarías el periodo. Hallado el periodo, se hace la división 1 entre el periodo y obtenemos la frecuencia de la señal.

Algunas de las funciones que se usaron en el código son las siguientes.

|  |  |  |
| --- | --- | --- |
| Nombre función | Parámetros | Retorno |
| Bool Verificarcuadrada | (float a[ ], int c) | True or false |
| Bool Verificartriangular | (float a[ ], int c, float t) | True or false |
| Bool Verificarsenoidal | (float a[ ], int c, float t) | True or false |
| Int tiposenal | (float a[ ], int c, float t) | 0<=int<=5 |
| Float frecuencia\_st | (float a[ ], int c, float t) | Float>0 |

Las funciones de retorno booleano, cumplían la función de verificar si los valores que había en el arreglo de sus argumentos cumplían los requisitos para ser una función de uno de los 3 tipos definidos (senoidal, triangular o cuadrada). La función de retorno entero, solo cumple la función de retornar una bandera, esto para imprimir mas cómodamente por la pantalla LCD. Y la función de retorno float cumple con retornarme la frecuencia de una señal previamente definida como senoidal o triangular.

La realización de este circuito y su implementación crea grandes desafíos. A medida que se realiza el código, pueden surgir más y más problemas, más variables, más parámetros de medición, lo que complica el desarrollo del código. Al tener una plataforma con pocos recursos de muestra de datos y de obtención, además a medida que hacemos mayores las variables, las soluciones se vuelven tediosas, poco concluyentes e inciertas. Sin embargo, se plantean soluciones aproximadas y de un rango que sea aceptable dado los recursos.

Es claro que existen indeterminadas formas de resolver el desafío propuesto. Se pueden implementar códigos distintos, algoritmos diferentes, otro análisis pero que, entre ellos, llegan a una conclusión más precisa. Se dio esta solución, ya que era la más viable y que el equipo de trabajo determinó y llevó a cabo para dar la solución lo más precisa posible.

Uno de los cambios mas notorios que hubo fue agregar el tiempo el cual nos brinda la función “millis()” , este nos permite usar un “tiempo de espera” menor, obteniendo más datos por segundo y a su vez podemos calcular el promedio de datos por segundo dividiendo el numero total de datos que tiene el arreglo por el tiempo en que se demoro este en recolectarlos, esto mejoro mucho el reconocimiento de algunas funciones.