Informe del Desafío I de Informática II *Fecha: 17/09/24*

Implementación Arduino: Identificador de Ondas

Santiago Cortés Osorio **s.cortes@udea.edu.co**
Sara María Hincapié Hincapié **sara.hincapie1@udea.edu.co**

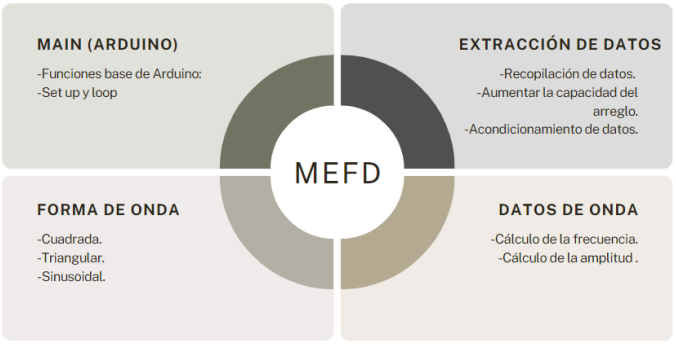
Resumen: Este proyecto se centra en el análisis de señales utilizando un Arduino y un generador de señales. Se requiere la adquisición y procesamiento de una señal para determinar su amplitud, frecuencia, y forma de onda. Esto se logra mediante la captura de datos, su análisis mediante algoritmos específicos, y la visualización de los resultados en una pantalla LCD.

1) Diseño

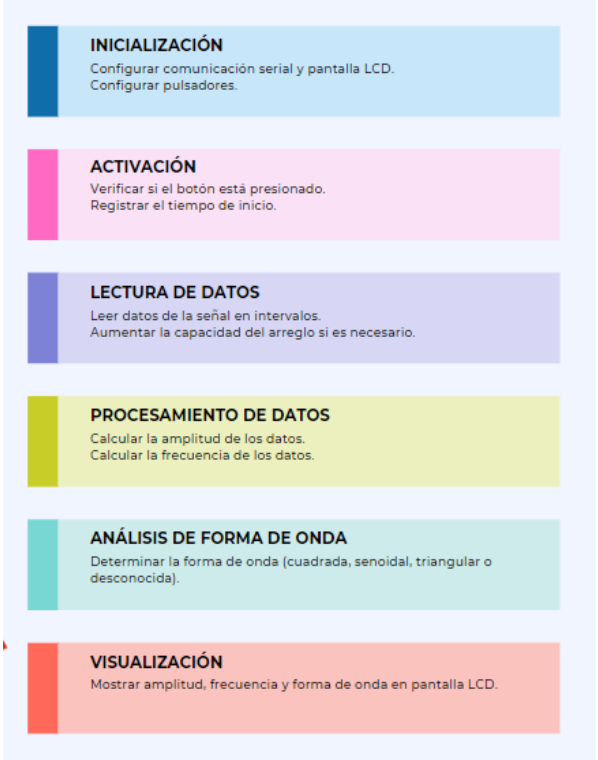
1.1 Modelo Estructural

Primero se desestimó la problematización subyacente, identificando conceptos clave como el tipo de onda, sus parámetros y la extracción de información relevante. Con base en esto, se delimitó el problema y se definieron indicadores para las funciones necesarias.

Adicionalmente, en la delimitación del programa, se estableció un conjunto de funciones específicas para el montaje en Arduino, abarcando la configuración del generador de señales, la adquisición de datos, y la visualización en la pantalla LCD.



2) Esquema de Procesos:



3)Análisis del Problema:

Este problema requiere de diversos análisis matemáticos que permitan modelar adecuadamente la onda a partir de los datos recolectados. Es crucial lograr un equilibrio entre la cantidad de datos recolectados: deben ser suficientes para lograr una reconstrucción adecuada de la onda, pero no tan excesivos para evitar una mala gestión de la memoria. De ahí podemos partir del dato de que Arduino posee la capacidad de tomar un dato aproximado cada 104 microsegundos, no obstante se tomó la decisión de limitar el muestreo cada 10 milisegundos para conseguir el balance esperado.

En cuanto a la idealización del modelo que logre identificar los 3 tipos de onda y en su defecto si es una desconocida, esto se puede identificar analizando los valores, seguido de someter estos a una verificación con base a 3 criterios, discriminando cada uno para cada tipo de onda. La onda de carácter cuadrado debe oscilar entre dos valores, la sinusoidal simulando una onda de seno, consecutivo a una comparación de valores entre ambas ondas, mientras que para la triangular esto puede conseguirse promediando la "pendiente" de los primeros 10 valores, sucesivo a una comparación de este valor junto al promedio general de toda la onda de " pendiente."

Se debe tener precaución en la frecuencia de muestreo para que la cantidad de datos a almacenar no sean excesivos (para administrar la memoria efectivamente) pero que sean lo suficientemente significativos para poder reconstruir la onda. Si bien Arduino tiene la capacidad de tomar un dato aproximadamente cada 104 microsegundos, se optó por limitarlo a una vez cada 10 milisegundos, para lograr este balance. Los 3 tipos de onda se pueden identificar analizando los valores, y sometiéndolos a una verificación en base a 3 criterios, uno para cada tipo de onda. La cuadrada si oscilan entre 2 valores los datos, la senoidal simulando una onda de este tipo y comparando los valores, y la triangular promediando la "pendiente" de los primeros 10 valores y comparando este valor con el promedio de pendiente de toda la onda.

En ese sentido, para la amplitud se puede realizar una suma entre el valor superior e inferior y dividirlo entre 200, puesto que hay que entregar el valor en voltios, mientras que para la frecuencia se realiza un conteo de la cantidad de cambios de signo, que posteriormente, se divide entre 4 (la división entre 2 se hace porque hay el doble de la cantidad de cambios de signo que el valor de la frecuencia, y de nuevo se divide entre 2 dado que se toman aproximadamente 200 datos, que corresponden a 2 segundos).

4) Algoritmos Implementados

A continuación se presenta más detalladamente la especificación de cada una de las funcionalidades planteadas en el programa.

4.1 Main:

Se completó la conexión del circuito utilizando un LCD I2C para minimizar el cableado. Además, se incorporó un osciloscopio para visualizar las ondas, reemplazando la necesidad del monitor serial para la depuración del código. Los pulsadores están configurados para manejar la adquisición de datos, asegurando que el sistema solo recoja información cuando se activa el pulsador correspondiente.
En el void loop, el sistema se activa al detectar la presión del pulsador. Se inicia la adquisición de datos y se calcula la información clave: la amplitud y la frecuencia de la señal, que será mostrada en el LCD. Además, el programa identifica el tipo de onda (cuadrada, senoidal, triangular) basándose en el patrón de datos recolectados.

4.2 Extracción de Datos:

Con el fin de manejar eficientemente la recolección de datos en una arquitectura de software con grandes limitaciones, se implementó una estrategia basada en la gestión de la memoria dinámica. Por eso se pautó una función que expande el arreglo en el cual se almacenan los datos de las señales; este va en aumento en bloques de 20 elementos cuando es necesario. Al mismo tiempo, se limitó la recolección de datos a uno cada 10 milisegundos, lo que equivale a un dato cada 104 microsegundos. Esto permite realizar un equilibrio entre la precisión de los datos y el uso de la memoria.

Otra funcionalidad que se añadió está relacionada con la forma de onda, específicamente para diferenciar si la señal es sinusoidal, para realizar el ajuste de la función en caso de que no comience en 0. Para esto se usaron dos funciones: una de ellas se encarga de alinear los datos de manera que la señal empiece en el punto correcto; posteriormente, agrega datos adicionales si es necesario para que la longitud de la señal sea adecuada. La otra función es la encargada de realizar el ajuste horizontal de la señal; está calcula cuánto necesita desplazarse la señal en el tiempo para que coincida con el valor esperado de la onda senoidal en ese punto.

4.3 Forma de la Onda:

Se planea que el programa pueda distinguir entre 3 tipos de onda; en caso de que no sea alguno de estos, se clasifica como onda desconocida.

Primeramente, se analiza cómo identificar si una onda es cuadrada; para esto se evalúan los valores de la señal, y se hace un conteo de la cantidad de niveles distintos. Para que una señal sea clasificada como cuadrada, debe alternar entre dos valores únicos.

Por otro lado, para que el programa identifique si una onda es triangular, se plantea comparar los valores que arroje la onda por medio de las diferencias, puesto que estas deberían seguir una secuencia constante. Esto quiere decir que esta onda tiende a tener rampas lineales ascendentes y descendentes; por lo tanto, las diferencias entre valores consecutivos deben ser consistentes en una secuencia que alterna entre aumentos y disminuciones lineales.

Finalmente, para determinar si una señal corresponde a una senoidal, se capturan los datos de la onda; en caso de que la onda no inicie en 0, se crea artificialmente un pedazo de onda para simular su inicio en 0. Posteriormente, se compara la señal ya ajustada con una señal senoidal; se busca comparar ambas señales y acumular el error. Por lo tanto, se espera que este error sea menor que al compararlo con otro tipo de ondas.

4.4 Datos de Onda:

Con el fin de obtener información específica de la onda, en este caso, la amplitud y la frecuencia, se realiza el cálculo de ambas en funciones distintas.

La frecuencia se define como cuántas veces la onda se repite en un periodo de tiempo determinado; en este contexto, se cuenta el número de transiciones entre valores positivos y negativos para identificar ciclos completos de la señal. Cada transición de un estado positivo a negativo o viceversa se cuenta como un cambio de estado. Al final del análisis, se divide el total de ciclos detectados por 4 para obtener la frecuencia en términos de ciclos por segundo.

Por otra parte, para la amplitud se identifican los valores máximos y mínimos de la onda; se ajusta el valor de la amplitud para que sea múltiplo de 100. Se realiza una sumatoria entre el valor mínimo y máximo y se divide entre dos para asegurar que se mantenga el rango de la distancia.

5. Desafíos Afrontados

A lo largo del desarrollo se afrontaron diversos retos, como el manejo de la memoria. En primer lugar, la limitación del tamaño del arreglo presentó desafíos a la hora de manejar el stack, debido que la función para verificar si la onda tiene forma senoidal requiere un "alto" poder computacional (alto para las características del Arduino) y se estaba sobrepasando el límite de memoria del stack, lo cual generaba un error que decía "invalid header file"; esto fue un gran reto porque el error no tenía ninguna descripción útil y la información de Internet no fue de gran ayuda, fue por medio de la técnica de ensayo y error que logramos encontrar la causa de este fallo para así darle solución, no obstante tomó un tiempo considerable.

También, teníamos problemas a la hora del debugging, antes de implementar la función de sobreescritura de datos, ya que si sobrepasábamos los 200 datos imprimía por el monitor serial una "ÿ", más tarde pudimos aprender que esto significaba que el arduino se había quedado sin memoria.

6. Consideraciones- Evolución del Desarrollo

6.1 Evolución del Desarrollo:

En primera instancia se realizó un análisis sobre la la resolución del problema. Seguidamente, se realizaron las conexiones correspondientes al Arduino.

Se empezaron a elaborar las funciones correspondientes a la extracción de datos, igualmente se introdujo la funcionalidad del aumento en la extracción de los datos generados por la onda, asimismo el programa logró identificar cuando una onda sea cuadrada.

Se realizó el cálculo de la frecuencia y amplitud de la onda, acompañado de la creación de dos funciones en relación al reconocimiento de tipo de onda; triangular y sinusoidal .

No obstante, la mejora de la función seno se continuó en varios momentos, con funciones que simulaban la onda. Los últimos días se hicieron correcciones en los tipos de variables para que estos se acoplen mejor y se realizará una gestión más responsable de la memoria.

6.2 Consideraciones:

En consideraciones, queremos resaltar la importancia del manejo de la memoria. Este problema no solo lo tuvimos que afrontar a la hora de crear el arreglo para almacenar los datos, sino también al invocar funciones, puesto que al ser muy ambiciosos con la verificación del seno, esta función estaba sobrepasando el límite del stack, lo que generaba que la plataforma Tinkercad mostrará un error con el mensaje “invalid header file”, sin ninguna descripción adicional, lo que dificultó en gran medida el desarrollo de la actividad visto que tuvimos que encontrar la causa del error por nuestra cuenta, debido a que en Internet no aparecía información útil para solucionar este error.

También, queremos destacar el valor de realizar el desafío a tiempo y no dejarlo para los últimos días, dado que siempre hay cosas para mejorar en el código y el tiempo adicional que se tiene de terminar a tiempo el desarrollo de un algoritmo funcional permite volverlo mucho más robusto y corregir errores que en caso contrario no se hubiesen notado.